0-Order Affordances through CAD-Model Recognition and 6DOF Pose Estimation

Aitor Aldoma and Markus Vincze
ACIN - Vienna University of Technology
aldoma@tuwien.ac.at

Radu Bogdan Rusu
Willow Garage

***Abstract*— In this paper, we propose to analyze the interaction opportunities between a robotic agent and the environment through object recognition and 6DOF pose estimation. We introduce the concept of 0-order affordance representing affordances that dependent solely on the object of interest and its configuration in the scene. For object recognition, we propose a semi-global 3D feature: the Clustered Viewpoint Feature Histogram (CVFH), that can be trained on CAD models and yet deliver excellent recognition results when recognizing objects from a single view point with a depth sensor like the Microsoft Kinect and together with the Camera's Roll Histogram (CRH) delivers a full 6-DOF pose estimation. Because the pose of the models in the scene is completely estimated, we are also able to perceive hidden 0-order affordances - affordances that are not directly perceivable in the current configuration - by using the stable poses of an object to decide if the affordance is usable or not. We present recognition results and pose estimation results on a preliminary set of objects showing better performance than state-of-the-art methods and demonstrate as well the applicability of stable poses to detect hidden 0-order affordances.**

I. Introduction and related work

From a robotic perspective being able to understand a scene and moreover, understanding which are the interaction possibilities that are provided in a specific environment, are a key capability for a task-guided robotic agent. What an environment affords depends strongly on two factors: the objects and their configuration in the scene and second, the interaction capabilities embodied on a specific agent. The combination of both factors is coined under the term affordance in the literature [1].

Directly perceiving affordances or interaction chances from a partial view or an image of a scene is an extremely difficult task. Several authors have already pursuit this goal by learning affordances directly on color images or depth images [2], [3]. The main disadvantage of such approaches are the underlying low level features on which the learning is based. 2D features or color information are not always adequate to learn geometrical attributes of objects that are in the end those voting for a specific affordance. Some years ago, other authors tried building functional 3D databases of objects to describe which are the functions provided by specific object / categories [4], however, the link between these databases and the real world was missing and the databases were independent of the agent.

Herein, we propose to handle the high complexity of the interaction opportunities between a scene and an agent by analyzing affordances at three different levels. We define 0-order affordances to be those that depend solely on the object. They can be understood as the intended functionalitiy of an object (e.g., a mug affords *liquid-containment* and a sphere affords rollable). 1st-order affordances are possible interactions between the objects and the capabilities of the robot (e.g., grasping an object if the agent embodiment has a gripper and the object is graspable by the specific embodiment). Finally, 2nd-order affordances describe those appearing when the agent embodiment has been extended by grasping an object in the scene (e.g., stacking an object onto another after the first object has been grasped).

In the scope of the paper, we focus on 0-order affordances. Although $0-$order affordances are independent of the agent, they do depend on the configuration or pose of the objects in a scene (e.g., a mug affords *liquid-containment* if standing upright but not otherwise). We call 0-order affordances that depend on the pose of an object, hidden 0-order affordances, as the specific object configuration might hide them to the agent.

Instead of perceiving affordances directly from the scene, we propose to perceive them through object recognition. Given a partial view of a scene, we recognize the objects in it against a CAD model database and estimate their 6DOF pose. In a training stage, 0-order affordances are labeled on the training CAD models. If an object presents hidden 0-order affordances, the stable poses of the objects are labeled as having the specific affordance hidden or not when found in that configuration. Upon successful recognition and pose estimation, the model affordances are retrieved. If hidden 0-order affordances are present, the object pose needs to be checked against the stable poses of the object to retrieve if the specific affordances are hidden or not for the current configuration.

Our main idea is to transform the problem of detecting affordances to an object recognition problem and use the stable poses of the recognized CAD model to retrieve hidden and non-hidden affordances. Moreover, by perceiving affordances through recognition of CAD models, affordances on new objects can be directly learnt at the CAD model level where the whole geometry is available and if desired other attributes like weight, material, *etc*. can also be included in this representation.

II. 0-ORDER AFFORDANCES

0-order affordances are independent of the agent and its capabilities and depend only on the object and its pose in the environment. While certain objects might fulfill certain affordances in all possible environment configurations, many others might only do so in a specific pose configuration. Note that for simplification, we consider the environment to be composed only by the object of interest and a planar surface where the object is positioned. For example, a 0-order affordance of a sphere is *rollable*. However, in a cluttered environment where the sphere is surrounded by other static objects, the sphere would not be able to roll. This kind of affordance perception would require a deeper analysis of the environment and represents future work.

A. Training stage

Given a set of object affordances $\mathcal{A}$ and an object set $\mathcal{O}_0$ we start by creating object - affordance relationships given by a human operator. We have a CAD model representation of each of the object in the starting set. An object $o \in \mathcal{O}_0$ is displayed to the operator together with the list of affordances $\mathcal{A}$ and the the operator decides which are afforded by the object. The operator decides also if any of the affordances might be hidden when the object is found in the environment of a robot. If yes, the operator is shown the object in different stable poses and for each of them, the operator decides if the affordance is hidden or not (see Fig. 1).

The set $\mathcal{A}$ of 0-order affordances in the scope of the paper is:

- *rollable*: the object can roll if pushed.
- *containment*: the object can contain other objects.
- *liquid-containment*: the object can contain liquids.
- *unstable*: the stability of the pose is compromised if pushed.
- *stackable-onto*: objects can be stacked onto the object.

Note that 0-order affordances might be pruned afterwards when higher orders affordances are analyzed depending on the embodiment of the agent or the specific task to be fulfilled. If the agent is unable to push: rollable or unstable 0-order affordances will be ignored when analyzing 1-order affordances. Affordances like containment or stackable-onto are analyzed when 2-order affordances are considered as they depend on the object being manipulated by the agent - the manipulated object must fit into the objects with the 0-order containment affordance. Thus, 0-order affordances represent an initial set of interaction opportunities and the amount of interaction opportunities decreases while going through the higher level affordances.

B. Computing stable poses

The stable poses of an object are those where the object pose is supposed to remain unchanged if not moved. As noticed in [5], the stable planes of a model are a subset of the tangent planes enclosing a model - the planar faces of the convex hull. The triangle faces of the convex hull can be grouped in planar faces by performing a hierarchical clustering [6]. The final planar faces represent the tangent planes Π that need to be further analyzed for stability.

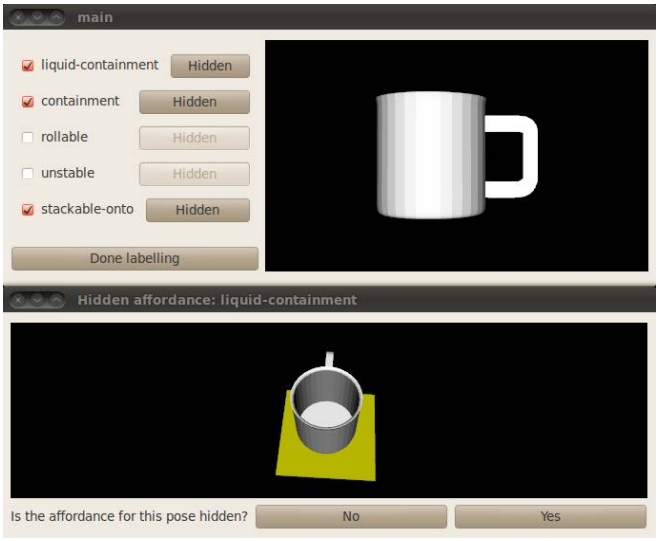


Fig. 1. Screenshot of the labeling tool. In the upper part, the user is given a CAD model to label which 0-order affordances are provided and when pressing hidden, the window in lower part pops up where the user decides for each stable pose if the affordance is hidden or not.

Consider $\pi \in \Pi$, the model is rotated in such a way that the normal of π matches the y-axis of the world coordinates and translated to stand on π. Let $cH(s)$ be the 2D convex hull of the points supporting the plane. π is a stable plane if the projection of the center of mass of the object lies inside $cH(s)$.

Highly symmetrical shapes like bottles, glasses or boxes, have many different stable planes, that do not provide additional information. In other words, the geometry of the objects when standing on those planes is exactly the same. To ease labeling, these non-informative stable planes are deleted and only one of them is included in the final set.

The model is projected onto π and the projected points are aligned using CPCA to the canonical axes. The 3D model is rotated accordingly around the plane's normal. Before π is added to the final set of stable planes, the model in the current pose is compared against the poses of the previous stable planes by checking how many points in the current pose do not have a neighbour within a threshold in previous computed poses. π is added to the final set if the same geometry is not found in the stable poses. By applying this simple check, the final number of stable planes is greatly reduced for these symmetrical shapes. For example, a cube ends having one single stable plane instead of 6, a cylinder 2 instead of $N >> 2$ (N depends on how fine the 3D model is discretized), etc.

III. RECOGNITION AND 6DOF POSE ESTIMATION

This section focuses on the recognition and pose estimation of the objects on a given scene. Because of its efficiency and good performance shown in [7], we decide to build on the VFH descriptor and modify it accordingly to obtain a semi-global descriptor (CVFH) that fits our needs of

allowing training on synthetic data and yet performing well on real data. The VFH descriptor is a compound histogram representing four different angular distributions of surface normals (see [7] for a complete description). Let p_c and n_c be the centroids of all surface points and their normals of a given object partial view in the camera coordinate system (with $||\mathsf{n_c}|| = 1$). Then $(\mathsf{u}_i, \mathsf{v}_i, \mathsf{w}_i)$ defines a Darboux coordinate frame for each point $\mathsf{p_i}$:

$$\begin{aligned} \mathsf{u}_i &= \mathsf{n_c} \\ \mathsf{v}_i &= \frac{\mathsf{p_i} - \mathsf{p_c}}{||\mathsf{p_i} - \mathsf{p_c}||} \times \mathsf{u}_i \\ \mathsf{w}_i &= \mathsf{u}_i \times \mathsf{v}_i \end{aligned} \tag{1}$$

The normal angular deviations $\cos(\alpha_i)$, $\cos(\beta_i)$, $\cos(\phi_i)$ and θ_i for each point $\mathsf{p_i}$ and its normal $\mathsf{n_i}$ are given by:

$$\begin{aligned} \cos(\alpha_i) &= \mathsf{v}_i \cdot \mathsf{n_i} \\ \cos(\beta_i) &= \mathsf{n_i} \cdot \frac{\mathsf{p_c}}{||\mathsf{p_c}||} \\ \cos(\phi_i) &= \mathsf{u}_i \cdot \frac{\mathsf{p_i} - \mathsf{p_c}}{||\mathsf{p_i} - \mathsf{p_c}||} \\ \theta_i &= \mathrm{atan2}(\mathsf{w}_i \cdot \mathsf{n_i}, \mathsf{u}_i \cdot \mathsf{n_i}) \end{aligned} \tag{2}$$

For $\cos(\alpha_i)$, $\cos(\phi_i)$ and θ_i histograms with 45 bins each are computed and a histogram of 128 bins for $\cos(\beta_i)$, thus the VFH descriptor has 263 dimensions. Using the centroid and average normals over the partial view ($\mathsf{p_c}$ and $\mathsf{n_c}$) to build the Darboux coordinate system, makes VFH sensitive to missing parts of the object caused by partial occlusions, segmentation or sensor artifacts (see Figure 2).

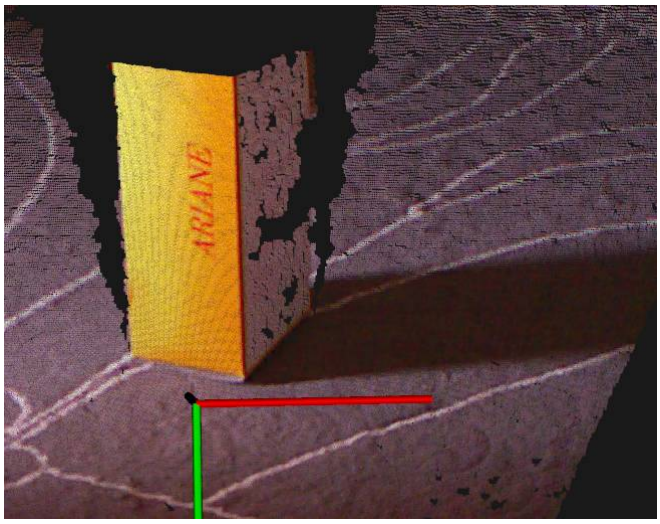


Fig. 2. Example of an incomplete surface due to limitations of the sensor. For instance, surfaces that are at a steep angle relative to the sensor as well as parts that are close to object borders contain more noise and even miss depth estimates.

These effects can result in unstable estimations of the object points and normals centroid (p_c and n_c from Eq. 1), thus affecting the resulting VFH and making it unsuitable to match against the corresponding synthetic view that will not present these artifacts.

A. The Clustered Viewpoint Feature Histogram

The main idea behind CVFH is to take advantage of the object parts that can be robustly estimated by the depth sensor and use them to build the Darboux coordinate system while still using the whole partial view to compute the descriptor.

Formally, we propose to describe a partial view of an object, represented by a set of points $\mathcal{P}$, as a set $\mathcal{H}$ of Clustered Viewpoint Feature Histograms. The cardinality of $\mathcal{H}$ is the same as the cardinality of $\mathcal{S}$, where $\mathcal{S}$ is the set of stable regions found on $\mathcal{P}$ using the procedure defined in the upcoming Section III-B.

Taking $\mathsf{s}_i \in \mathcal{S}$ with $\mathsf{s}_i \subseteq \mathcal{P}$, we can define a Darboux coordinate system $\mathcal{D} = (\mathsf{u}_i, \mathsf{v}_i, \mathsf{w}_i)$ like in Eq. 1 but in this case p_c and n_c represent the euclidean centroid and normal centroid of s_i and not of the whole partial view $\mathcal{P}$. Given $\mathcal{D}$ and using Eq. 2, the normal angular deviations for all points in $\mathcal{P}$ can be computed.

Let then $(\alpha, \phi, \theta, \beta)$ represent the normal angular deviations already binned in (45,45,45,128) bins, the CVFH histogram $\mathsf{h}_i \in \mathcal{H}$ is defined as the following concatenation:

$$(\alpha, \phi, \theta, \mathcal{SDC}, \beta) \tag{3}$$

where $\mathcal{SDC}$ represents the Shape Distribution Component of CVFH computed as follows:

$$\mathcal{SDC} = \frac{(\mathsf{p}_c - \mathsf{p}_i)^2}{\max((\mathsf{p}_c - \mathsf{p}_i)^2)} \tag{4}$$

The number of bins used for this component is again 45 thus making a total size of 308 for CVFH. This component allows to differentiate surfaces that have very similar normal distributions and sizes but their points are distributed differently.

To avoid scale invariance, each bin in CVFH counts the absolute number of points falling in that bin. To reduce ambiguities, we first construct a voxel grid over our point cloud data with a fixed voxel size ($5mm$ for all our experiments which reduces the number of points to be processed and yet does not smooth fine structures), and reduce the cloud to the set of voxel centroids. Because the actual size of the object is given by the 3D sensor, the amount of points for a given view will be the same no matter what the distance to the camera is. Avoiding the normalization step allows us to distinguish between objects of different size but identical shape. It also makes the descriptor more robust to missing parts of the object, as this will only influence local parts of the descriptor. Normalizing the histogram by the total number of points would increase the bin height under the presence of occlusion.

The advantages of CVFH are two-fold: (i) the coordinate system is more likely to resemble the one obtained from the synthetic view making the descriptor more stable and (ii) because the set of CVFHs represent a multivariate description of the partial view, we can better handle occlusions as long as at least one of the stable region is visible. Please note that the CVFH histograms in H are independent from each other

and not complementary as they describe the same geometry but encode them differently. To understand how CVFH is used for recognition, we refer the reader to Section IV-B.

B. Stable regions clustering

To overcome the instability caused by missing object parts and local noise artifacts, we first identify stable regions in a partial view obtained by the depth sensor. To do so, we apply a smooth region growing algorithm on the points obtained from a partial view of an object after removing points with high curvature (caused by noise, object edges or non-planar patches).

Each new region is initialized with a random point. A point p_i with normal n_i is added to a region C_k if the region contains a point p_j with normal n_j in the direct neigbourhood of p_i with a similar normal, i.e., the following constraint is fulfilled:

$$\exists p_j \in C_k : ||p_i - p_j|| < t_d \wedge n_i \cdot n_j > t_n \tag{5}$$

For all our experiments, t_d is set to three times the voxel grid size and t_n to $\cos(10^\circ)$. For each stable region, a CVFH descriptor is computed as outlined in the previous section. The number of stable regions for a specific partial view defines the cardinality of the descriptor set $\mathcal{H}$.

C. Camera roll histogram and 6DOF pose

Most descriptors based on views of an object like VFH, CVFH, CAP-SIFT [8] are unable to deliver a complete 6-DOF pose. Due to this invariance of CVFH with respect to rotations about the view direction of the camera (roll), the object and viewpoint recognition is determined up to an unknown rotation. To determine the correct orientation of the object, we introduce a new descriptor that is not invariant to the roll angle. To avoid a higher dimensionality in the overall descriptor by extending it, which would noticeably decrease the performance of the object/viewpoint recognition, we use a final optimization step to find the correct roll angle. Since the computation of the roll angle is only done for the best N candidates from the CVFH matching step and furthermore is efficient to calculate, the overall performance is hardly affected.

For each CVFH descriptor in $\mathcal{H}$, an additional histogram is computed - *the camera's roll histogram* (CRH). We project the normals at each point onto a plane that is orthogonal to the vector given by the camera center and the centroid of the stable region used to compute CVFH. For the projection, we compute a rotation-axis v and a rotation angle θ using the following equation:

$$\begin{aligned} v &= \frac{p_c \times z}{||p_c||} \\ \theta &= -\arcsin\left(||v||\right) \end{aligned} \tag{6}$$

that transforms the CVFH centroid $\mathsf{p_c}$ to coincide with the camera's z-axis. Since we use an orthographic projection, the projected normals are given by the first two components of the transformed normals.

The CRH is then computed by taking the angle of the projected normal relative to the up-view vector of the camera on the plane. The histogram contains 90 bins giving an angular resolution of 4 degrees. The number of bins for the CRH is selected from our empirical evaluations to provide a reasonable trade-off between efficiency and accuracy. Due to noise in the input data, we weight the projected normals by their magnitudes. This removes most of the equally distributed noise in the histogram, resulting from unstable projections of normals that are almost parallel to the roll axis of the camera.

In order to estimate the object's rotation around the roll axis, we need to find an orientation where the two roll histograms match best according to a metric. This can be considered a correlation maximization problem. Therefore, we apply a Discrete Fourier Transform for both histograms, and multiply the complex coefficients of the database view with the complex conjugate coefficients, and perform the inverse transform to compute the cross power spectrum R. The peaks of this spectrum appear at rotation angles that align the two histograms well.

There are cases where the power spectrum of two CRHs can have multiple peaks due to different kinds of symmetries. Also, partial occlusions or sensor noise might deteriorate the CRH, so it is generally not sufficient to rely solely on the maximal peak in R.

In order to select a set of orientations that can be pruned in a subsequent test, we select a minimum threshold t_p for peaks, and add peaks with higher magnitude to the set. We start with the highest peak, adding peaks if their corresponding rotation angles do not fall within a certain distance band t_b of any of the previously added peaks. This ensures that the set of orientations does not contain multiple entries for very similar alignments, but captures local maxima that are distributed over the whole set of rotations, if they indicate a good alignment.

In our experiments, we set $t_b = 12^\circ$ and chose a relatively high value for t_p in order to keep the size of the rotation set small. We found a value of $t_p = 0.9 * \max(R)$ to yield a low number of peaks - typically up to 5 peaks.

IV. Detecting 0-order affordances

This section focuses on two aspects: (i) the recognition module that matches the trained CAD models to a pointcloud obtained by a depth sensor like the Kinect and (ii) the detection of the 0-order affordances once the recognition and pose estimation has succeeded.

A. Object recognition - training

Each CAD model is rendered from different viewpoints to obtain a partial point cloud of the object seen from the viewpoint that is used to train the view-based descriptor (CVFH) that is used for recognition and pose estimation. We use 42 viewpoints around the object obtained by tesselating an icosahedron once.

Indistinguishable views that belong to the same object, like those obtained from symmetric objects such as cylinders or

bowls, are considered just once to avoid indistinguishable information in the database.

B. Object recognition + pose estimation - detection

The recognition stage runs on a raw point cloud from a depth sensor, which in our case is the Kinect. We proceed first with a segmentation of the scene using dominant plane extraction and Euclidean segmentation on the remaining points [9]. The segmented groups of points represent the objects to be recognized. Independently for each object in the scene:

1) Compute a set of CVFH descriptors (H) and CRHs. Please note, that each CVFH descriptors is paired with a camera roll histogram.
2) For each CVFH in H, a nearest neighbor (NN) search is performed to find the N closest CVFH descriptors in the training set, giving a set of views from the trained objects.
3) As we have performed as many NN-searches as elements in H, the best N candidates according to the metric given in Eq. (7) are selected.
4) For the resulting N view candidates the roll angle is determined using the CRH matching and 6DOF pose estimation (as detailed in Section III-C).
5) After aligning the views using the pose and roll information gathered so far, an additional ICP [10] step is used to refine the alignment.
6) Finally the N best view candidates are sorted using the number of inliers from the last iteration of ICP using a distance threshold of twice the voxel grid size.

Because of its efficiency, we use the FLANN library [11] to perform the nearest neighbor search.

We have performed different empirical experiments to determine which is the best metric for our needs. The major problem with metrics like L1 and L2 is the sensitivity to outliers. Dealing with partial occlusions implies that the histograms will have outliers due to missing parts of the objects even if the rest of the histogram is shaped correctly. Let A and B be two CVFH descriptors, we propose the following metric which favours overall likely shaped histograms and distributes the outliers weight over the whole histogram:

$$d(A,B) = 1 - \frac{1 + \sum_{i=1}^{308} \min(A_i, B_i)}{1 + \sum_{i=1}^{308} \max(A_i, B_i)}, \qquad (7)$$

Once the pose has been estimated and the best CAD model has been selected, we are already able to provide the non-hidden 0-order affordances for the specific object, e.g. the stackable-onto affordance for a box that is independent of its pose in the environment.

C. Hidden 0-order affordances

For those recognized objects that have hidden 0-order affordances we need to evaluate if their pose in the environment makes any hidden 0-order affordance usable. Let $\mathcal{M}_1$ represent the object in camera coordinates after being aligned using the procedure explained in Section IV-B and n_{dp} the normal of the dominant plane in the scene. Let $\mathcal{M}_2$ represent the same object in object coordinates together with the set of stable planes Π, where each $\pi \in \Pi$ has been labeled to have the specific 0-order affordance hidden or not hidden. The problem can be then posed in the following way: Find $\pi_i \in \Pi$ that best aligns $\mathcal{M}_2$ with $\mathcal{M}_1$ and check if the affordance at the stable pose based on π_i is hidden. We use the method presented in [5] to align $\mathcal{M}_2$ and $\mathcal{M}_1$ (assumed to stand on the plane with normal n_{dp}). Because the method is based on stable planes, the best alignment gives a certain π_i from $\mathcal{M}_2$ and by looking at the labeled information from π_i we know if in the current configuration the hidden 0-order affordance is indeed hidden or usable. Note that in our representation, a hidden 0-order affordance is a boolean variable and we do not consider poses where the object might partially fulfill the affordances. In the case that the pose retrieved by the procedure in Section IV-B does not represent a stable pose, the system will consider all pose dependant affordances to be hidden. Even so, the following section IV-D applies as well for this cases as long as the retrieved pose is correct.

D. Using hidden 0-order affordances to give manipulation hints

If the agent is looking for a specific affordance needed to fulfill a certain task that it is not available in the current configuration, we can give manipulation hints after detecting that the specific affordance is hidden (see Fig. 3). Let p_s be the current pose where the affordance is hidden and p_f any other stable pose where the specific affordance is not hidden. The manipulation hint consists of a constrained grasp that leaves the support surface of p_f free so that the object can be released in the desired pose and a path for the robot hand that brings the object from p_s to p_f while using the proposed constrained grasp.

V. Experimental evaluation

We performed a preliminary evaluation on a set of 6 objects containing a cylinder, a ball, a book, an opened box, a mug and a bowl together with the set of affordances presented before. There are two factors to evaluate:

1) Is the object correctly recognized and its pose correctly estimated?
2) If 1) succeeds, are we able to identify the stable pose properly so that hidden 0-order affordances are correctly retrieved?

To evaluate these factors, 8 different scenes (like the one in Fig. 3) are captured, each of them containing several objects from our training set and we visually inspect the recognition results together with pose estimation, which are summarized in Table I. The same scenes are also recognized using VFH and we show how CVFH outperforms VFH for recognition of CAD models on real scenes obtained with the Kinect. For both recognition methods, the same post-processing is done. Table I also shows recognition and pose estimation results according to the number of post-processed nearest

	Recognition and pose estimation rates			
	CVFH		VFH	
#NN processed	5	10	5	10
Scene 1	2/2	2/2	2/2	2/2
Scene 2	1/2	2/2	1/2	1/2
Scene 3	3/3	3/3	3/3	3/3
Scene 4	4/4	4/4	3/4	3/4
Scene 5	3/4	4/4	2/4	3/4
Scene 6	4/5	5/5	3/5	4/5
Scene 7	4/5	5/5	3/5	3/5
Scene 8	5/6	6/6	4/6	5/6
Total:	26/31	31/31	21/31	24/31
Percentage:	83.9 %	100 %	67.7 %	77.4 %

TABLE I

RECOGNITION AND POSE ESTIMATION RATES FOR DIFFERENT SCENES. #NN PROCESSED REPRESENTS THE NUMBER OF NEAREST NEIGHBORS THAT ARE POST-PROCESSED.

neighbours as detailed in Section IV-B. We perform a total 31 recognitions and Table I shows how CVFH outperforms VFH. It is interesting to note that the post-processing of the N (N being 5 and 10 for the experiments) nearest neighbours is always able to retrieve the good match if available.

The hidden 0-order affordances were always correctly retrieved for each positive recognition demonstrating the usability of the stable planes alignment method to deliver the correct stable pose.

VI. CONCLUSIONS AND FUTURE WORK

We have presented how hidden and non-hidden 0-order affordances can be perceived from a single viewpoint through object recognition, using stable object poses and labeled information on the training models. Moreover, we argue that abstracting affordances to a 3D CAD model representation opens the possibility to learn affordances on new objects using 3D features that are ported to the real world through recognition and should be more effective than current approaches that directly try to learn affordances using 2D features.

Future work includes dealing with a bigger set of 0-order affordances and objects, learning affordances using a supervised learning strategy on new CAD models and the integration of manipulation hints together with 1- and 2-order affordances in the grasping pipeline to solve specific manipulation tasks.

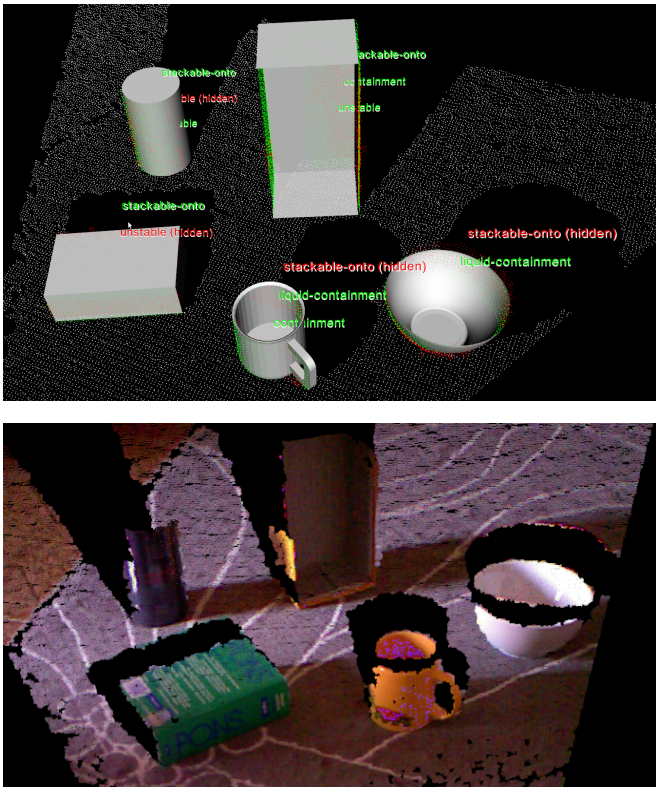


Fig. 3. Top: A scene recognized with CVFH. The CAD models are overlapped on the scene together with the 0-order affordances that each object provides. Hidden affordances are shown in red and usable affordances are shown in green. The green points represent the matching training view and the red points the points coming from the depth sensor. Best viewed in color. Bottom: Same scene from the Microsoft Kinect with color image overlayed. Note the difference between the CAD models used to trained CVFH and the data obtained from the Kinect. The recognition of the shoe box shows an example of how CVFH can deal with occlusions.

REFERENCES

[1] J. J. Gibson, "The Theory of Affordances," in *Perceiving, Acting, and Knowing - Toward an Ecological Psychology, Lawrence Erlbaum Ass., Hillsdale, New Jersey*, 1977, pp. 67–82.

[2] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor, "Learning Object Affordances: From Sensory–Motor Coordination to Imitation," *Robotics, IEEE Transactions on*, vol. 24, no. 1, pp. 15 –26, feb. 2008.

[3] S. Griffith and A. Stoytchev, "Interactive Categorization of Containers and Non-Containers by Unifying Categorizations Derived From Multiple Exploratory Behaviors," *Association for the Advancement of Artificial Intelligence (AAAI). Atlanta, Georgia.*, 2010.

[4] M. Sutton, L. Stark, and K. Bowyer, "Gruff-3: Generalizing the domain of a function-based recognition system," *Pattern Recognition*, vol. 27, pp. 1743–1766, 1994.

[5] A. Aldoma and M. Vincze, "Pose Alignment for 3D Models and Single View Stereo Point Clouds Based on Stable Planes," *International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, 2011.

[6] M. Attene, B. Falcidieno, and M. Spagnuolo, "Hierarchical mesh segmentation based on fitting primitives," *The Visual Computer*, vol. 22, pp. 181–193, 2006.

[7] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3D Recognition and Pose Using the Viewpoint Feature Histogram," in *Proceedings of the 23rd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, 10/2010 2010.

[8] Corey Goldfeder and Matei Ciocarlie and Jaime Peretzman and Hao Dang and Peter K. Allen, "Data-driven grasping with partial sensor data."

[9] R. B. Rusu, A. Holzbach, M. Beetz, and G. Bradski, "Detecting and Segmenting Objects for Mobile Manipulation," in *ICCV S3DV workshop*, 2009.

[10] Z. Zhang, "Iterative Point Matching for Registration of Free-form Curves," 1992.

[11] M. Muja and D. G. Lowe, "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration," in *International Conference on Computer Vision Theory and Application VISSAPP'09)*. INSTICC Press, 2009, pp. 331–340.

A Hybrid Approach For Object Reconstruction & Recognition

Nizar Sallem and Michel Devy

***Abstract*— Object reconstruction and recognition often constitute the underlying techniques employed in robotics manipulation. They are two major topics that keep motivating the computer vision community. Availability of RGB-D sensors at reasonable costs can achieve a milestone in providing a common framework for achieving both requests. The synchronized 3D and 2D stream enable for a new approach that produces a unified model suitable for reconstructing and recognizing *a priori* unknown objects to be manipulated later. In this paper we present an hybrid approach that takes benefits of 3D and 2D data to achieve faster reconstruction while providing strong keys for recognition.**

I. INTRODUCTION

While object reconstruction deals with retrieving object geometrical information such as volume and shape, object recognition deals with identification of an object in a scene and further determining its positioning. These topics are differently addressed according to the object representation. In the following we give a summary of object reconstruction using 2D data and 3D data.

A. Reconstruction from 2D data

2D reconstruction can be categorized in 3 groups:

- multiview geometry: the object is seen with one or several cameras from one or several positions. Then using matching techniques 3D positions of corresponding pixels are computed knowing the intrinsic and extrinsic parameters of the cameras ;
- setero reconstruction: the object is seen in two cameras or more and then applying some correlation the 3D points are recovered ;
- composite approach: the authors use multiview combined to stereo bench to achieve the reconstruction.

The multiview geometry has one major advantage : it only requires one camera and an initial guess of the intrinsics and extrinsics. The achieved reconstruction is sparse and suffer from the lack of features to match. The second issue can be addressed using some pattern projection. The stereo reconstruction achieves better reconstruction and the output model is a dense one. Still it requires at least two cameras. The composite approach allows for several point of view (**POV**) and thus the entire object can be reconstructed.

B. Reconstruction from 3D data

The object of interest is shot from several POVs or several devices are placed around it to take several shots at the same time and then a consolidation method - most likely Iterative Closest Point (**ICP**) - is applied either afterward or incrementally to obtain the reconstructed model. The output model is precise but ICP is an expensive algorithm. Another issue is the semantic poorness of the model. Indeed, the most common output models are point cloud model and meshed point cloud which don't support appearance data.

This work was supported by the ANR project ASSIST.

Nizar Sallem and Michel Devy are with LAAS-CNRS Université de Toulouse nksallem@laas.fr michel@laas.fr

C. The Proposed Approach

In this paper we propose an hybrid approach for reconstruction based on 2D features detection and matching in conjunction with 3D coordinates availability producing a geometrical model with 2D interest features when available. We then extend the PLY file format to handle both data. Finally we demonstrate the performances of this hybrid approach over basic pure 3D for object recognition.

II. BRIEF RGB-D SENSOR PRESENTATION

The **RGB-D** denomination in this article refers to a device producing synchronized depth image and color image stream such as $Microsoft^{®}$ kinect.

Fig. 1. RGB-D sample output of a scene

We are not focusing on implementation details, we just point the fact that RGB-D sensor produces color image and depth map such as $\forall$ *pixel* $p_i(u,v)$ *of the color image corresponds a 3D point* $M_i(X,Y,Z)$ *that is set to infinity*

out of the functioning range. Experimentally the functioning range for the Kinect is $[1m, 5m]$.

III. 2D RECONSTRUCTION AND RECOGNITION WITH NATURAL INTEREST FEATURES

Interest features could be defined as sufficiently discriminant pixels in an image to represent their close neighborhood. They were introduced by [5]. [4] presented HARRIS features corresponding to strong intensity variation on the edges and corners of an image. [7] introduced scale invariant features (SIFT). [9] developed scale invariant and rotation invariant features (SURF).

3D reconstruction can be achieved using natural 2D interesting features by the mean of multiview geometry which involves N perspective cameras or a single perspective camera at N positions.

- detect some features on an image
- match those features against the previous one(s)
- each pair of $< reference_{feature}, match_{feature} >$ is appended to a system solving $(u, v) = M(X, Y, Z)$ with M projection matrix.

Following are some detection result on the same image using different detectors.

Fig. 2. Detection result on the same image using SIFT, SURF and STAR

A. Features Matching

Almost all the modern interest features detectors assign a real number array to each point. The descriptor translates signaletic and, or, geometrical information computed on that point into numerical values. Comparing two interest features for matching resolves then to computing the euclidean distance that separates their descriptors. For a triplet $< f_0, f_1, f_2 >$ described by $< \delta_0, \delta_1, \delta_2 >, \delta_i \in \Re^D \forall i$, then f_1 is more likely to be matching f_0 than f_2 if $\|\delta_1 - \delta_0\| < \|\delta_2 - \delta_0\|$. Numerical tools like FLANN [1] and ANN [2] allow for such distance computations in reduced time by approximating the numerical representation. Still, euclidean distance computation doesn't prevent from false matching and outliers thus the matches are filtered more through some other constraint - geometrical for instance. [14] computes similarities between pairs and the one with the largest number of voters s_0 is said to be valid. All the pairs that fall in the range of s_0 are accounted as valid matches and the others are discarded. Another approach that gives similar result are the groups matching [3] where signaletic and geometrical constraints are combined to achieve robust pairing.

Our matching algorithm is similar to [14] except the accounted geometric transformation and the voting mechanism. Consider two images with two sets of detected features DR_1 and DR_2 then each DR_i is indexed in a k-d tree structure. $(f_{1,j}, \delta_{1,j}) \rightarrow (f_{2,k}, \delta_{2,k}), \|\delta_{1,j} - \delta_{2,k}\| = DR_2 argmax \|\delta_{1,j} - \delta_{2,k}\|$. f is further described by an angle θ and a scale σ. Rather than an overall loop we gather these two data 1. scale: $\frac{\sigma_{2,k}}{\sigma_{1,j}}$ and 2. rotation $\theta_{2,k} - \theta_{1,j}$ in a 2D histogram H to get the repartition of all affine transformations A_j. By thresholding H to its upper half as shown on Fig. 3 we keep only the most present transformations and through reprojection we get the valid pairs. On the Fig. 4 the initial matches are drawn in green and then overlayed in red for valid ones. This techniques shows to be robust for scale, rotation and arbitrary transformations changes.

Fig. 3. Built histogram of rotations and scales thresholds

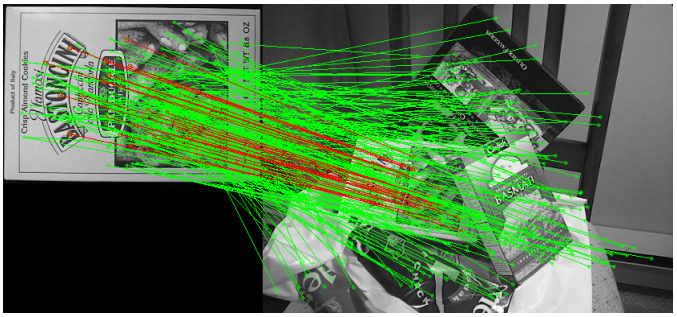


Fig. 4. Matching result: initial matches in green and valid matches in red

The set of pairs $< reference, match >$ detected respectively on $image_i$ and $image_j$ is called matching result and noted $MR_{i,j}$.

B. Multiview Geometry

The fundamental matrix of a perspective camera 1 ties a point $P(X, Y, Z)$ in world coordinates to a pixel $p(u, v)$ in the image.

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = M \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = IE \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad (1)$$

with I = $\begin{pmatrix} k_u f & 0 & u_0 & 0 \\ 0 & k_v f & v_0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ and E = $\begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix}$.

Then for unknown P that projects into $p_1, p_2, \ldots p_n$ we can write a system that returns (X, Y, Z) given the camera[s]

intrinsic and extrinsics. The sparse structure of the system allow serious improvements as in [6] and [8].

IV. 3D RECONSTRUCTION AND RECOGNITION

A. Reconstruction

3D reconstruction using 3D points use a totally distinct approach where the main challenge is to consolidate several views into one model. In [16], authors proposed a method for registering 3D shapes. The original algorithm is summarized in IV-A. With $C(X,P) = Y = \{y\}, y = d(\vec{p},X) = min\|\vec{x}-\vec{p}\|$ the closest points operator.

Require: point sets P with N_P points $\vec{p_i}$ from target and X with N_X points $\vec{x_i}$ from model, k_{max} maximum number of iterations allowed and τ the convergence tolerance
Ensure: $\vec{q}$ transformation from P to X
  $P_0 \leftarrow$P
  $q_0 \leftarrow [1,0,0,0,0,0,0]^t$
  $k \leftarrow 0$
  while $k < k_{max}$ **do**
    compute closest points $Y_k = C(P_k, X)$
    compute registration $(\vec{q_k}, d_k) = Q(P_0, Y_k)$
    apply registration $P_{k+1} = \vec{q_k}(P_0)$
    if $d_k - d_{k+1} < \tau$ **then**
    return $\vec{q_k}$
    end if
  end while

Original ICP algorithm

The original algorithm was further improved in different locations:

- P and X selection;
- corespondents calculus: point to plan distance instead of point to point, mahalanobis distance instead of euclidean,...;
- initial transform value: transform between $\hat{P}$ and $\hat{X}$ respectively means of P and X
- registration computations: quaternions, singular values decomposition, non linear methods,...;
- adding an outlier removing step using RANSAC, LMEDS, ...

In [17], authors depict a taxonomy of ICP variants and establish a comparison methodology noting the effect of each criterion on convergence speed. They conclude that point to plan metric is suitable for faster alignment and using $k-d$ tree to index point sets increases this speed.

B. Recognition

In [18], authors present Point Features Histograms (**PFH**) as robust descriptor for 3D points. PFH represent the angular variations between each pair of points normal in the vicinity k of a point p. PFH was then improved to reduce their complexity from Onk^2 to Onk in [15] and named Fast Point Feature Histograms (**FPFH**). FPFH translates the relationship between a point and its k neighbors and its neighborhood's neighborhood. Their experiments show their technique to be applicable for synthetic and real data and robust to noise. In [19], Normal Aligned Feature (**NARF**) is an interest feature extracted on depth maps. It corresponds to points detected on the borders representing the highest changes in surface at their local vicinity. Descriptor is formed after a normal aligned range patch at the point on which is overlayed a star pattern and each entry in the descriptor corresponds to the number of surface change along that star beam. These descriptors can be compared using the same tools in III-A to achieve recognition.

V. HYBRID RECONSTRUCTION METHOD

In a past work [10] we achieved 3D reconstruction using a camera fixed on a personal robot arm and the major drawback was the poorness of the resulting geometrical model because the natural features are not abundant in an image and if we relax the detection then we loose the robustness of those features and spend a lot of time in the matching step. On another level the availability of an appearance model allowed us to achieve recognition and localization tasks. When it comes to use the 3D map generated by a RGB-D sensor the registration was not always precise enough to produce a good model and we lost all the recognition abilities. Our challenge was to speed up reconstruction while producing an appearance model of the whole object.

A. Speed Up Reconstruction

After a quick analyze it was evident that the the slowness was due to the ICP. Indeed it is an iterative process that involves:

1) nearest neighbor search: $Onlog(p)$, n number of points, p number of neighbors
2) transformation estimation: On, n number of pairs

repeated N times or till convergence is reached. One way to reduce this cost is to "choose" the correspondence and thus eliminate the nearest neighbor computation step which is the most expensive. A good hint is given by the interest features matching. If we use only pairs computed through 2D matching then the process is at least $image_{width}ximage_{height}$ lighter since the matching operates on $2D$ and ICP operated on $3D$. This raises an issue: *what if number of pairs found is not sufficient to achieve transform computation?*, rather than falling to an overall ICP we chose to use only the "extreme" points. In fact, ICP can be seen as an averaging process and the points with extreme coordinates are the most affecting ones. Thus using the points that lie on the contour of the object would achieve same result as using the whole data. Now if for some reason none of the former succeed we can always fall back to using all the available 3D data. This behavior is summarized in V-A

Require: color image img_i, img_j ; 3D map map_i, map_j
Ensure: $\tau i,j$ 3D transform between img_i and img_j
  $DR_i \leftarrow$ detect features in img_i
  $DR_j \leftarrow$ detect features in img_j
  $MR_{i,j} \leftarrow$ match DR_i against DR_j
  $n \leftarrow$ number of pairs in $MR_{i,j}$
  if n $\geq threshold$ **then**
    $success \leftarrow$ compute $\tau_{i,j}$ using the pairs of $MR_{i,j}$

if not $success$ **then**
  $success \leftarrow$ run ICP on the contours of img_i and img_j
  if not $success$ **then**
    $success \leftarrow$ run ICP on map_i and map_j
  end if
else
  $success \leftarrow$ run ICP on the contours of img_i and img_j
  if not $success$ **then**
  $success \leftarrow$ run ICP on map_i and map_j
  end if
end if

3D transform recovery algorithm

B. Appearance Model

As shown in V-A the first step of our algorithm is to detect 2D features on input images thus keeping track of those computed features and saving them along with 3D data is sufficient to ensure appearance model construction. Still we needed an appropriate format to merge the data in. We chose to extend the **PLY** data format the way shown on V-B.

TABLE I
CLASSIC PLY AND EXTEND PLY FILE CONTENT

PLY	extended PLY
element camera	**element camera**
camera properties	camera properties
	element features
	feature properties
element vertex	**element vertex**
vertex properties	vertex properties + *features list*
element face	**element face**
face properties	face properties

C. Object Reconstruction

Achieving reconstruction in this hybrid approach require a 3D map and 2D image series taken from several POV not necessarily organized spatially. In order to integrate these inputs we incrementally build a graph G where nodes hold 2D detection data and 3D coordinates noted $\eta_i < image_{3D_i}, DR_i >$ and edges hold the computed 3D transform between images and the transformation error $A_{j,i} < \tau_{j,i}, \epsilon_{j,i}, MR_{j,i} >$ between η_i and η_j Given $image_{2D_i}$ and $image_{2D_j}, \forall j \neq i$ there exists 3 possibilities :

1) $\tau_{j,i}$ direct computation from pairs succeeds (blue edge on Fig:V-C) ;
2) $\tau_{j,i}$ is computed via ICP (red edge on Fig.V-C) ;
3) no connection can be established.

At each step we try matching the new input against all past ones and update G. There is no guaranty for G to be complete nor for the result to be precise enough thus a post processing need to be done to connect as much nodes as possible and enhance computed transform when necessary.

In this refinement step we traverse the nodes, if no edge between η_k and η_l or $\epsilon_{k,l} > \epsilon_{max}$ then we apply on

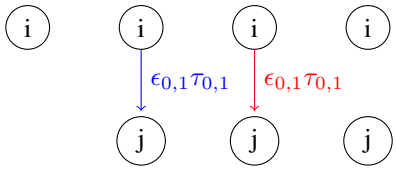


Fig. 5. 3 possibilities: direct computation, ICP, no success

$< image_{3D_k}$ and $image_{3D_l}$. If the computation of $\tau_{k,l}$ is successfully then we insert $A_{k,l}$.

Let us note G' the refined graph. Reconstruction is fulfilled if we find the path that relates the maximum of nodes while minimizing the global transformation error $= \sum_{j,i=0}^{N} \epsilon_{j,i}, j \neq i \in G'$. This turns to be Minimal Spanning Tree problem that can be solved using Kruskal algorithm [11].

VI. EXPERIMENTS AND RESULTS

For the experiment purpose we shoot a "Lego" box from several POV using $Microsoft^{®}$ kinect. We use an automated GrabCut [12] that operates the first selection based on foreground/background contrast. The image series is shown on Fig. VI.

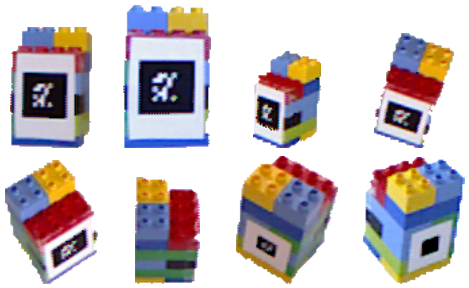


Fig. 6. Lego's box image series

The graph obtained when all images are added is shown on Fig. VI. G_{legos} shows three connected components where the best reconstruction that could be achieved covers 5 nodes from $8 < \eta_1, \eta_2, \eta_3, \eta_6, \eta_7 >$.

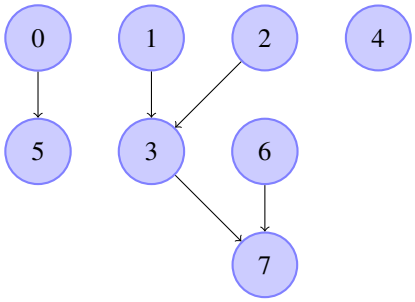


Fig. 7. G_{legos}: graph obtained incrementally adding "legos" box images

Refinement leads to Fig. VI. Several edges (drawn in blue whereas old ones are drawn in black) were added to the G_{legos} making it almost complete in 1 trial.

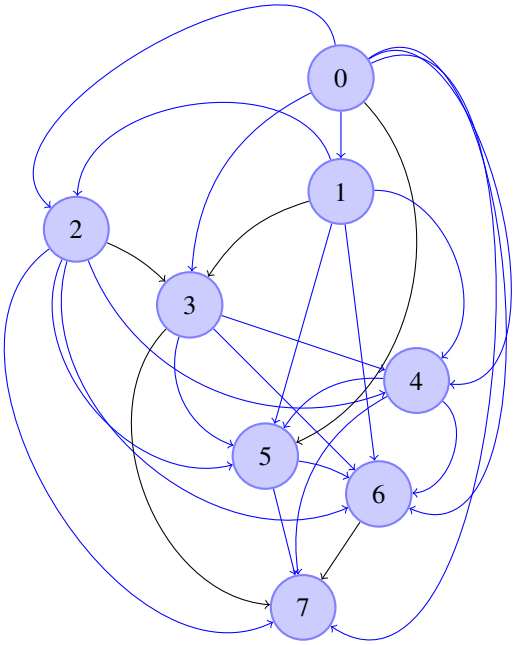


Fig. 8. G'_{legos}: refinement of G_{legos}

Running Kruskal on G'_{legos} results on Fig. VI. The weights on the edges represent $\epsilon j, i$ the average transform error in meters: $\epsilon j, i = \sum_{j,i=0}^{N} \|x_j - \tau_{j,i} x_i\|$ with N the number of samples retained for computing $\tau_{j,i}$.

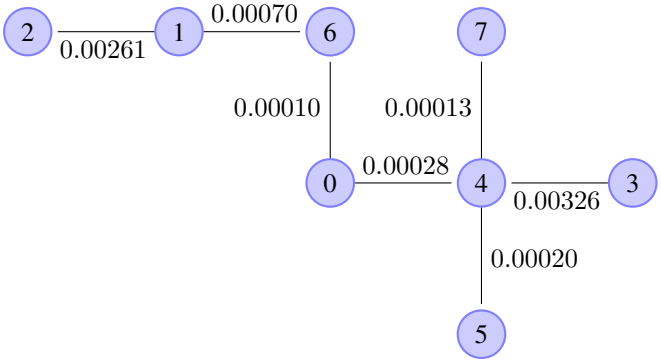


Fig. 9. G' MST as computed by Kruskal

The reconstruction is done selecting a start node η_{start} and following the path to an end node $_{end}$. When there is only one outgoing edge $A_{i-1,i}$, $image_{3Di} = image_{3Di} + \tau_{i-1,i} image_{3Di-1}$ else we need to compress all the outgoing edges : $\tau_{k,i-1} image_{3Dk} + image_{3Di-1}, \forall k \neq i, \exists A_{j,k}$. In Fig. VI, selecting η_2 as starting node and η_3 as end node, we need to compress η_5, η_4 and η_7 in $\eta_{4'}$. The new MST is shown on Fig. VI.

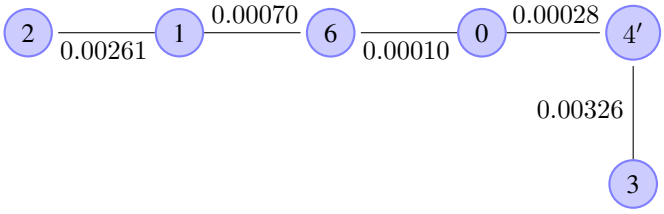


Fig. 10. MST after compression $\eta_{4'} = \eta_4 + \eta_5 + \eta 7$

VII. HYBRID REGISTRATION ALGORITHM

Our experiments shows that an hybrid approach is very convenient for RGB-D data registering. We integrate the transformation computation step in V-A into an iterative closest point standard algorithm to compute an initial guess of the transformation before further processing. This approach is quite similar to the SAC-IA (SAmple Consensus Initial Alignment) proposed in [15] where authors use local 3D keypoints Point Feature Histograms (PFH) matching to compute a first value for the desired transformation. The algorithm is named A Synthethized Methode for Alginement (ASMA) and is described in VII

Require: color image img_i, img_j ; 3D map map_i, map_j; 2d detector D; 2d matcher M
Ensure: $\tau i, j$ 3D transform between map_i and map_j
  $DR_i \leftarrow$ detect features in img_i
  $DR_j \leftarrow$ detect features in img_j
  $MR_{i,j} \leftarrow$ match DR_i against DR_j
  $n \leftarrow$ number of pairs in $MR_{i,j}$
  if n $\geq threshold$ **then**
    $success \leftarrow$ compute $\tau_{i,j}$ using the pairs of $MR_{i,j}$
    if not $success$ **then**
      $success \leftarrow$ run ICP on the contours of img_i and img_j
      if not $success$ **then**
        $success \leftarrow$ run ICP on map_i and map_j
      end if
    else
      $success \leftarrow$ run ICP on the contours of img_i and img_j
      if not $success$ **then**
      $success \leftarrow$ run ICP on map_i and map_j
      end if
    end if
    {apply RANSAC on the matched pairs}
    RANSAC

A Synthetized Methode for Alignement

VIII. CONCLUSIONS AND FUTURE WORKS

A. Reconstruction Performance

We first start comparing our hybrid approach to a basic ICP one. To do so we used the Lego's image series and the linear ICP class from PCL. In VIII-A we show average execution times for both approaches on the same data set

while selecting different image input order each time. Execution time are measured on an $Intel^{®}\ Core^{TM}$ 2 Duo CPU P8600 @ 2.40GHz.

TABLE II
COMPARISON BETWEEN HYBRID APPROACH AND 3D ICP

	hybrid	3D ICP
Adding Images		
time (ms)	1260	65217
Graph Analysis		
time (ms)	21804	X
MST Computing		
time (ms)	13140	88
Total		
	36204	427829

Our approach performs about six times better when building initial graph this is due to:

1) we use only 2D data for detection and matching
2) we try to use 3D ICP only on contours
3) at worst case we are behaving just equal to the basic approach with 3D ICP on full data.

The output graph of the 3D ICP method is shown on VIII-A

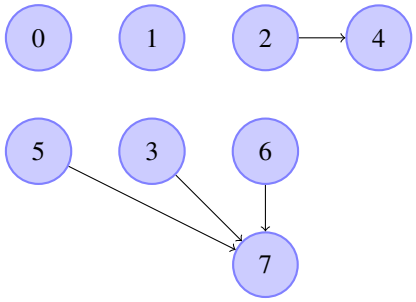


Fig. 11. G_{ICP}: graph obtained with pure 3D ICP

Graph analysis step is the most consuming part of the hybrid algorithm because it often falls to using 3D ICP to connect graph's components.

Finally the reconstruction purpose is better achieved with hybrid approaches which uses 4 images than basic ICP which achieves registration for only 3 images and this is due to the effort we made connecting the graph.

Even with the handicapping step of graph analysis, the hybrid approach performs 10 times faster than pure 3D approach with initial requirement better achieved: use as much input images as possible.

Reconstruction result is shown on VIII-A

B. Recognition Performance

To evaluate recognition performance a RGB-D output of the Lego's box was shot with the Kinect with an automated GrabCut and taken as learning image VIII-B. Then we took 3 others shots with:

1) the Lego's box is placed in front of a larger object VIII-B;

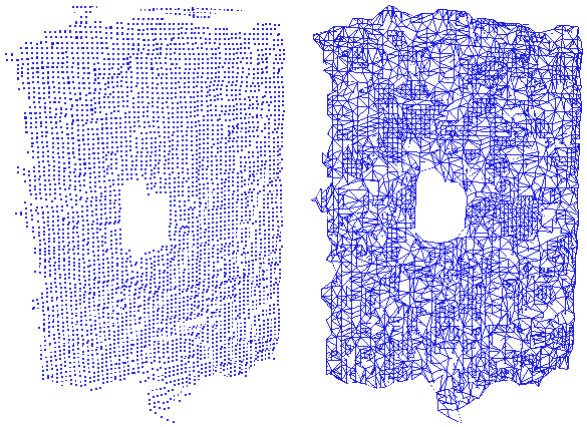


Fig. 12. reconstructed Lego's box point cloud and mesh

2) the Lego's box is placed among other objects VIII-B;
3) the Lego's box is twisted and placed alone VIII-B.

Fig. 13. recognition evaluation data set

VIII-B illustrates recognition process on the request dataset. The 2D features based recognition score ρ is measured $\rho = \frac{1}{N}\sum_{i=0}^{N}\frac{matched_{features}}{detected_{features}}$, N number of query images. In our measurements $\rho = 0.62$.

For the 3D approach evaluation we applied the PPF features detection and recognition [13] shipped with PCL with a single model image from the data set. Unfortunately, all our trials to recognize the Lego's box failed.

The hybrid approach outperforms 3D one for object recognition due to the richness of the signal information in the 2D images compared to the geometrical only information provided by the depth map.

C. Conclusions

RGB-D sensors offer more than 3D data and taking into account the richness of the images enhance drastically the reconstruction and recognition performances of a computer vision application. Hybrid approach shows to be faster and more adapted for object recognition than basic 3D approaches due to the lower dimensionality of its inputs and

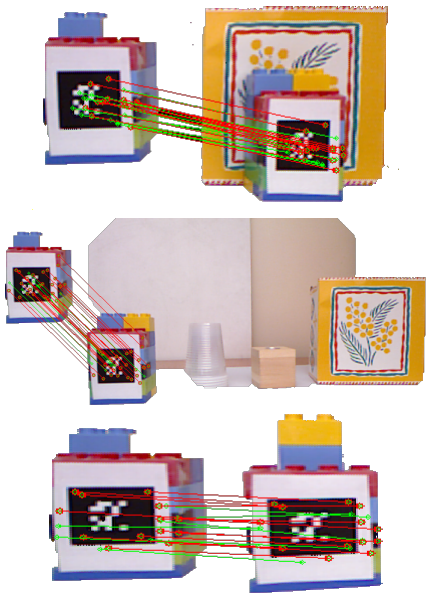


Fig. 14. recognition performance of SURF based 2D matcher

the accuracy of geometrical + signaletic 2D discrimination versus pure 3D geometric one.

D. Future Works

We are working at a full integration of our algorithm inside the PCL and improving the performance at the graph analysis step by selecting the most plausible pairs first and deleting the weak edges to reduce the MST computing time.

IX. ACKNOWLEDGMENTS

The authors gratefully acknowledge the contribution of National Research Organization and reviewers' comments.

REFERENCES

[1] M. Muja and D.G. Lowe, "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration", *in International Conference on Computer Vision Theory and Application*, INSTICC Press, 2009, pp. 331-340.

[2] S. Arya and D.M. Mount and N.S. Netanyahu and R. Silverman and A.Y. Wu, "An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions" *in ACM-SIAM SYMPOSIUM ON DISCRETE ALGORITHMS*, 1994, pp. 573-582.

[3] T. Lemaire, C. Berger, I-K. Jung, and S. Lacroix, "Vision-based SLAM: stereo and monocular approaches." *in International Journal on Computer Vision*, 2007, pp. 343-364.

[4] C. Harris and M. Stephens, "A combined corner and edge detection", *in Proceedings of The Fourth Alvey Vision Conference*, 1988, pp. 147-151.

[5] H.P. Moravec, "3D graphics and the wave theory" *in SIGGRAPH '81: Proceedings of the 8Th annual conference on Computer graphics and interactive techniques*, ACM, 1981, pp. 289-296.

[6] M.I.A. Lourakis and A.A. Argyros, "The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-Marquardt algorithm" *in Technical report, Institute of Computer Science*, 2004.

[7] D.G. Lowe, "Object recognition from local scale-invariant features", 1999, pp. 1150-1157 vol.2.

[8] M. Lhuillier and L. Quan. "A quasi-dense approach to surface reconstruction from uncalibrated images", *in IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005, vol 27, pp. 418-433.

[9] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features", 2006, pp. 404-417.

[10] N.K. Sallem and M. Devy, "Modélisation d'Objets 3D en vue de leur reconnaissance et leur manipulation par un robot personnel", *in ORASIS'09 - Congrès des jeunes chercheurs en vision par ordinateur*, 2009.

[11] J.B. Kruskal, "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem", *in Proceedings of the American Mathematical Society*, 1956, Vol.7, No 1, pp. 48-50

[12] C. Rother and V. Kolmogorov and A. Blake, ""GrabCut" : interactive foreground extraction using iterated graph cuts", *in ACM Trans. Graph.*, 2004, Vol.23, pp. 309-314.

[13] B. Drost and M. Ulrich and N. Navab and S. Ilic, "Model globally, match locally: Efficient and robust 3D object recognition.", *in IEEE CVPR*, 2010, pp. 998-1005.

[14] M.E. Munich and P. Pirjanian and E. Di Bernardo and L. Goncalves and N. Karlsson and D. Lowe, "SIFT-ing through features with ViPR", *in Robotics Automation Magazine, IEEE*, 2006, Vol.13, No 3, pp. 72-77

[15] R.B. Rusu and N. Blodow and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3D Registration", *in The IEEE International Conference on Robotics and Automation (ICRA)*, 2009

[16] P.J. Besl and N.D. McKay, "A Method for Registration of 3-D Shapes", *in IEEE Trans. Pattern Anal. Mach. Intell.*, 1992, Vol.14, pp. 239-256

[17] S. Rusinkiewicz and M. Levoy, "Efficient Variants of the ICP Algorithm", *in Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, 2001

[18] R.B. Rusu and Z.C. Morton and N. Blodow and M. Beetz, "Learning Informative Point Classes for the Acquisition of Object Model Maps", *in Proceedings of the 10th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2008

[19] B. Steder and R.B. Rusu and K. Konolige and W. Burgard, "NARF: 3D Range Image Features for Object Recognition", *in Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010

Can a 3D Classifier Be Trained Without Field Samples?

B. Douillard, A. Quadros, P. Morton, J.P. Underwood, M. De Deuge
The Australian Centre for Field Robotics, The University of Sydney, Australia

Abstract— **This paper presents a 3D classifier that is shown to maintain performance whether trained with real sensor data from the field or purely trained with 3D geometric models (for example, computer aided design (CAD) models downloaded from the internet). It is shown that the proposed 3D classifier outperforms Spin Image and Fast Point Feature Histogram (FPFH) based classifiers by up to 30%. The proposed classifier is a global 3D template matching technique which exploits the knowledge of the position of the ground for more accurate alignment of the objects above. The experimental results suggest that field samples may not be required in the training set of alignment-based 3D classifiers, which potentially has major implications on the way the training of 3D classifiers is approached.**

I. Introduction

This paper introduces an approach to 3D classification which does not require field samples for training but only 3D models downloaded from the Internet (as illustrated in Fig. 2). The approach bypasses feature extraction and directly compares 3D shapes via geometric alignment. It is equivalent to a 3D template matching process. Variants of the alignment process are compared and it is shown that constraining the alignment by using the knowledge of the location of the ground provided by 3D segmentation [3] improves the classification. A metric for shape comparison is proposed and used as a second step in the classification process once the alignment is performed. The alignment based classifiers are compared to standard feature based classifiers and are shown to provide significant improvements. Critically, it is shown that the template set used in conjunction with the alignment process can be entirely formed of 3D (CAD like) models downloaded from the Internet, while achieving performance on par with templates made of field samples (from sensors). These experimental results potentially have major implications on the way 3D classifiers are trained. It suggests that field samples may not be required for training, which is an intuitive result since the underlying 3D geometry is independent from the 3D sensing device (provided the sampling is sufficient to capture the main elements of the object's geometry). Prior 3D segmentation is assumed to be given and can be generated using the techniques developed in [3].

II. Related Work

3D classification techniques can be organised into four classes: (1) classification based on global features, as in [13] for instance; (2) classification based on local features, as in [8]; (3) classification based on Bag Of Features (BOF) [10]; (4) classification by combining local descriptions into a spatial structure representing the global topology of an object [5].

The 3D classifiers tested in this study belong to the first category: "Classification based on global features". A key difficulty encountered by local methods is that local features are not necessarily unique to one class of object, which causes ambiguity in matching. In contrast, by matching a full object scan to a full template scan, various scales of feature are implicitly considered at once. One drawback to this approach occurs when attempting to match occluded or partially observed objects and this is left for future work. Note that given the segmentation techniques recently proposed in [3], occlusions can be identified by fast ray-tracing in the datastructure underlying the segmentation. This provides a mechanism to assess whether the (segmented) objects to be classified were occluded in the scene and as a consequence whether the use of a global 3D classifier is appropriate.

The use of web (CAD like) data for 3D feature based classification has recently been investigated in [9]. An adaptation mechanism allows appropriate re-weighting of the web data with respect to actual field samples during training. Our approach does not consider adaptation techniques but experimentally shows that training sets entirely formed of web data can lead to improved performance while avoiding the otherwise critical steps of (1) acquiring training samples from the environment the classifier is to be tested in and (2) labelling these samples. Synthetic 3D models have also been used to train the body parts recognition system deployed for human gesture tracking in gaming applications involving the newly released Kinect sensor [14]. The authors showed that such training data provides accurate recognition on dense (Kinect generated) 3D data. The results presented in this paper lead to similar conclusions for sparser (Velodyne generated) 3D data.

The global matching of 3D models onto 3D models has received a lot of attention [15]. However, to the best of our knowledge, the alignment of sparse scans acquired in the field to 3D models had not been investigated in the context of 3D classification.

The approach proposed here builds on previous work [4] which already investigated shape alignment for 3D classification. This publication brings the following new elements to the study: (1) four different types of alignment strategies are tested and comapred against Spin Image and Fast Point Feature Histogram (FPFH) based classifiers, (2) it is experimentally demonstrated that template shapes obtained from 3D (CAD like) models found on the Internet can provide equivalent performance and even in some cases improved performance compared to using objects scanned in the field as templates. In the long term, this will enable a user to choose the classes to be identified by simply selecting 3D models (from an online database for instance) which will define a training set for the system. Also, this last result

suggests that field samples may not be required in training data to perform alignment-based 3D classification, which potentially has major implications on the way the training of 3D classifiers is approached.

III. Feature Based Classification

This section presents a set of global features whose classification performance is then compared to the proposed alignment based classifier.

A. Global PCA

Global PCA features were used as a benchmark since they are amongst the simplest features. They represent the overall three dimensional extent of an object point cloud. In the implementation used here, the first two dimensions contain the smallest and the largest eigenvalues of the point cloud in x and y. The third dimension contains the variance of the point cloud along the z axis. The z dimension of the sensor frame is used as opposed the third eigenvalue of the full object point cloud to capture the typical extrusion of objects in urban environments. The z direction is defined as the normal to the ground which is obtained from 3D segmentation [3]. Classification is based on K Nearest Neighbour (KNN).

B. Spin Images as Global Features

In previous work [4], the Spin Image feature [8] produced the best classification results among the tested local descriptors, on par with the alignment based classifiers. It is, as a consequence, used here again for comparison; the main difference being that the 3D point clouds are now sparser, since they are generated from Velodyne scans, as opposed to the dense Riegl scans in [4]. The Riegl scans used in previous work contained about 1.7 million points, while a Velodyne scan contains about 130,000 points for approximately similar coverage, implying that a Velodyne scan is about one order of magnitude sparser. To address the sparsity of the data spin images are computed using a larger support than in previous work (5m, see Sec. V-C). While Spin Images are local features, this deployment effectively makes them global features, as in [7]. A Spin Image is built by first creating a 3D surface mesh in which each point is a vertex. Matching two objects requires computing the description at each vertex of the associated mesh. In our implementation the surface mesh is obtained by exploiting the natural grid topology of the range image, as in [11]. In addition, to speed up the process, the description is computed only at a subset of randomly sampled vertices. Classification is based on KNN, as in [8].

C. FPFH

The Fast Point Feature Histogram (FPFH) [13] is a local feature designed for realtime applications, with an implementation in the ROS software library [6]. It requires surface normals, which are computed as in [11]. At each local region, a weighted histogram of surface normal variations are computed. The distance between histograms is computed with the histogram intersection kernel to find point correspondences [13]. Classification is by KNN as before.

IV. Classification by Template Alignment

The approach for 3D classification developed in this section was first introduced in previous work [4]. It is a 3D template matching process involving two main steps: (1) 3D alignment, (2) computation of a distance metric quantifying shape differences. These two steps are detailed in the following sections.

A drawback of global (object size) alignment is related to the potentially large number of templates required to obtain good matches as the number of object classes considered increases. Future work will consider non-rigid ICP techniques [2] which morph one object into another and as a consequence allow improved matching between any two objects. With such techniques, shape difference is quantified as the amount of morphing required to transform one object into another. This type of metric has the potential to lead to a reduction of the template set to a few samples per class, ideally *only one per class*, since good matches (prior to morphing) are no longer required.

A. Alignment Methods

Four different alignment methods are tested in Sec. V. The simplest one consists simply of shifting each point cloud to its mean. The other alignment methods are variants of ICP. In the first variant, the point clouds are shifted to their mean and a full 6 DOF ICP is run. This will be referred to as ICP 3D. In the second variant the point clouds are also shifted to their mean, but this time a 4 DOF ICP is run in which the x, y, z and yaw components of the rigid transformation are optimised. It will be referred to as ICP 2.5D. In the third and last variant, the x and y coordinates of the point clouds' centre of mass are shifted to 0 and the z component is shifted so that ground height is zero. The ground height is obtained from 3D segmentation [3]. A 3 DOF ICP is then run in which the x, y and yaw components are optimised. It will be referred to as ICP 2D. This last ICP variant leverages the knowledge of the ground surface given by the segmentation process as a way to constrain the ICP optimisation and avoid some of the local minima.

Many other variants of ICP have been proposed, varying aspects from point selection and matching to the minimisation strategy. As generalised by [12], most ICP implementations can be characterised by six aspects. Our design involves the following choices: (1) *selection*: nearest neighbour search in object clouds down-sampled by voxelisation; (2) *matching*: L_2 norm; (3) *weighting*: none; (4) *rejection*: none, since segmentation has been been performed prior to applying a classifier; (5) *error metric*: as in Eq. 1; (6) *minimisation*: non-linear least square optimisation (based on Python's Numpy function numpy.leastsq).

B. Shape Similarity Metrics

Once alignment is performed, the two different metrics presented below are used to quantify the difference in shape between the two objects aligned.

1) ICP Residual Error: The first metric considered is the value of the residual produced by ICP:

$$err = \sum_{i=1}^{N_{\text{Test}}} ||\mathbf{P}_i^{\text{Test}} - \mathbf{P}_{nearest}^{\text{Template}}|| \quad (1)$$

The subscript $nearest$ refers to the nearest neighbour of P_i^{Test} in the template point cloud.

2) Symmetric Residual: The formulation above may artificially take on large or small values: for instance in the case of a pole aligned to a wall, Eq. 1 will produce a small value when the test shape is the pole but a large value when the pole is the template. To avoid this artefact, Eq. 1 is extended into a symmetric measure:

$$\begin{aligned} err &= \frac{1}{N_{Test}} \sum_{i=1}^{N_{\text{Test}}} ||\mathbf{P}_i^{\text{Test}} - \mathbf{P}_{nearest}^{\text{Template}}|| \quad (2) \\ &+ \frac{1}{N_{Template}} \sum_{i=1}^{N_{\text{Template}}} ||\mathbf{P}_i^{\text{Template}} - \mathbf{P}_{nearest}^{\text{Test}}||, \end{aligned}$$

The formulation is similar to the one proposed in previous work [4], the difference being that each of the two terms is now normalised by point counts. This makes it insensitive to variations in sampling densities provided a number of samples large enough to capture the main elements of geometry in the shape[1].

V. Experiments

A. Annotated Velodyne Data

A visualisation of the manually labelled dataset is presented in Fig. 1. It was generated from a sequence of Velodyne scans by applying the segmentation techniques developed in [3]. When necessary, the segmentation was manually corrected and the resulting segments hand labelled into the seven following classes (the number of instances in each class being indicated in parenthesis): car (134), pedestrian (47), tree (51), building (13), trunk (32), pole (50), traffic sign (47); total number of labelled objects: 374. The sequence was divided into four non-overlapping subsets for four-fold cross-validation testing. All the results provided in this section are generated according to this set-up.

The classification problems considered here do not contain the class "other" which corresponds to the assumption that every test shape belongs to one of the seven considered classes. This assumption does not hold in real-world applications. A possible solution is to develop gating processes, as defined in the tracking and data fusion community. An instance of such process for classification was investigated in [4].

B. Scanned Objects as Training Data

In these first sets of results scan objects are used both training and testing. In Sec. V-C, the training sets are formed entirely of 3D models downloaded from the Internet.

[1]This minimum number of samples corresponds to the Nyquist-Shannon sampling criteria applied to the 3D surface forming the object.

truth \inferred	car	pedestrian	tree	building	trunk	pole	traffic sign
car	126	4	1	2	1	0	0
pedestrian	3	38	0	1	0	0	5
tree	11	0	34	4	0	2	0
building	4	2	7	0	0	0	0
trunk	1	0	1	0	10	7	13
pole	0	0	0	0	12	28	10
traffic sign	0	3	0	0	9	15	20

TABLE I Confusion matrix: Global PCA; accuracy = 0.68

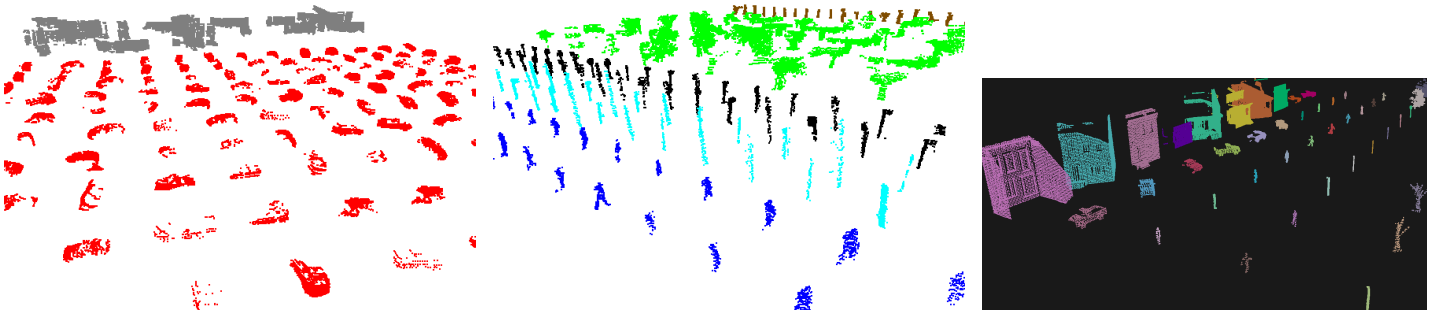
1) Global PCA: The Global PCA features described in Sec. III-A were classified with KNN and provided the results in Table I. KNN was chosen since it is similar to the template based approach which classifies by matching a test shape to the nearest template, closeness being evaluated with the metrics described in Sec. IV-B.

Table I presents the confusion matrix which corresponds to an accuracy of 0.68 with a standard deviation of 0.04 across the four-fold cross validation tests. The average computation time per cross-validation test is 0.03 secs for KNN classification (K=1) and 0.04 secs for the computation of all features.

The results in Table I correspond to high precision and recall values for the classes car and pedestrian: 0.94, 0.87 and 0.81, 0.81, respectively. Trees are partially confused with cars (precision: 0.67, recall: 0.79) due to canopy points dominating the PCA computation and leading to geometrical extents similar to cars. Buildings are mis-classified; they are confused with cars and trees; this is due to the extents of cars and of tree canopies being similar to the extent of (the small segmented sections of) buildings in this dataset. The classes trunk, pole, and traffic sign are not well separated due to their similar geometrical appearance; the precision and recall values for these three classes are: trunk (0.31, 0.31), pole (0.56, 0.54), traffic sign (0.43, 0.42). If these three classes were gathered into one class, the overall accuracy would increase to 0.86. This set of results will be used as benchmark since the combination of Global PCA and KNN represents one of the simplest approaches one can choose for the classification task at hand.

2) Spin Images: The Spin Image feature described in Sec. III-B is used to perform classification with a KNN classifier. At each region of each object in the training data, a spin image is computed and stored. To classify a region of a test object, a spin image is computed and compared with all training spin images, using the spin image similarity metric in [8]. To provide a final class for the whole object, the most common estimated class from all the local regions is chosen. More advanced methods of combining local information exist [8] [13], where local matches support the alignment of a global template.

To reduce redundant overlap, spin images were computed at points 30 cm apart. For the spin image parameters, a 16x16 grid spanning 5 m in width and height was used; this was found to provide the best results as opposed to smaller grids. This is due to the sparsity of the Velodyne data compared to the Riegl data used in previous work [4]. The large image size provides an almost global descriptor, but computed at many points and orientations, removing the need for global



(a) buildings, cars (b) pedestrians, poles, traffic signs, trees, trunks (c) Template Set

Fig. 1. (a) and (b), visualisation of the set of labelled 3D segments; one colour corresponds to one class. (c) The set of synthetic 3D templates generated using the process described in Fig. 2.

truth \inferred	car	pedestrian	tree	building	trunk	pole	traffic sign
car	126	0	0	0	0	0	0
pedestrian	31	10	0	0	0	1	1
tree	8	0	42	0	0	0	0
building	2	0	3	10	0	0	0
trunk	6	0	1	0	13	10	1
pole	15	1	0	0	3	22	1
traffic sign	29	2	0	1	0	5	10

TABLE II CONFUSION MATRIX: SPIN IMAGES; ACCURACY = 0.66

truth \inferred	car	pedestrian	tree	building	trunk	pole	traffic sign
car	104	1	14	4	1	0	2
pedestrian	2	22	4	0	0	5	11
tree	3	0	46	0	0	0	1
building	5	0	2	8	0	0	0
trunk	0	0	1	2	15	4	9
pole	1	2	1	2	6	17	13
traffic sign	3	1	2	1	12	7	21

TABLE III CONFUSION MATRIX: FPFH; ACCURACY = 0.66

alignment. The confusion matrix is given in Table II for K=3, which provided the best accuracy of 0.66. All classes have a high confusion with cars, in particular pedestrians and traffic signs. Cars may be crowding out an overlapping feature space, causing failure with KNN classification.

Table II presents the confusion matrix computed on a set of 26,000 spin images in total. Using a C++ implementation on a Core2Duo 3Ghz computer, these images took 14 secs in total to compute, and 242 secs to classify all.

3) FPFH: The FPFH feature was computed similarly, by selecting spherical regions of 5 m radius to compute the feature on, with points subsampled to 30 cm spacing. Using the C++ ROS implementation, it took 23 secs to compute all features and 68 secs to classify them. An accuracy of 0.66 was achieved for k=3, with the confusion matrix shown in Table II. Buildings are often identified as cars, potentially due to similar surface normal variations. As with the other features, trunks, poles and traffic signs are difficult to distinguish between.

4) Classification by Template Alignment: The classification accuracies for the four types of alignment developed in Sec. IV-A are presented in Table IV (the accuracies standard deviation across the cross validation tests is indicated in parenthesis) and the corresponding computation times are reported in Table V (including alignment and symmetric residual computation, Eq. IV-B.2, corresponding to the comparison to an average of 251 templates; Python implementation on a 2.6GHz Intel Core processor).

The results in Table IV show that the use of the symmetric residual (Eq. 3) for shape difference calculation consistently provides a significant improvement in classification accuracy (16% on average). Using constrained ICP (i.e. ICP 2D or

	ICP 3D	ICP 2.5D	ICP 2D	no ICP
ICP residual	0.51 (0.18)	0.56 (0.13)	0.57 (0.11)	x
symmetric residual	0.66 (0.07)	0.72 (0.08)	0.73 (0.07)	0.65 (0.08)

TABLE IV FOUR-FOLD CROSS VALIDATION ACCURACIES

	car	pedestrian	tree	building	trunk	pole	traffic sign
ICP 3D	4.0	2.5	8.4	22.2	1.7	2.0	1.7
ICP 2.5D	4.0	3.1	8.5	20.2	2.0	1.8	1.5
ICP 2D	3.6	3.0	8.5	17.6	1.9	1.7	1.7

TABLE V COMPUTATION TIMES (IN MINS)

2.5D) improves on a full ICP solution (ICP 3D) by 7% while being slightly faster (by 2% on average); see Table V for computation times. This experimentally confirms that constraining the ICP alignment using the knowledge of the location of the ground provides improved classification. ICP based methods improve by up to 8% on alignment without ICP (i.e., the means of the point clouds are shifted to zero without any further alignment), as indicated by the column "no ICP".

This first set of results suggests that the classifier providing the best performance corresponds to the combination ICP 2D/symmetric residual. The corresponding confusion matrix is detailed in Table VI. This confusion matrix shows that the classes car, pedestrian and tree are correctly classified; corresponding precision and recall values: class car (0.96, 0.88), class pedestrian (0.77,0.88), class tree (0.76,0.91). The class building is confused with cars (precision and recall: (0.22, 0.32)) due to our data set essentially containing small sections of building whose extent is similar the one of cars. The classes trunk, pole, traffic sign are confused due to their similar geometrical aspect, with precision and recall values of: class trunk (0.22, 0.32), class pole (0.86,049), class traffic sign (0.34, 0.55).

Compared to the classification results obtained with the Global PCA features, the classifier ICP 2D/symmetric residual provides an improvement of 5% (0.73 against 0.68). This is due to the Global PCA feature representing the overall 3D dimensional extent of an object point cloud as opposed to local shape variations, and as a consequence providing a

truth \inferred	car	pedestrian	tree	building	trunk	pole	traffic sign
car	129	2	2	1	0	0	0
pedestrian	1	36	0	0	0	0	0
tree	8	0	39	2	2	0	0
building	9	0	2	2	0	0	0
trunk	0	1	0	0	7	17	7
pole	0	1	0	0	2	43	4
traffic sign	0	1	0	0	11	19	16

TABLE VI CONFUSION MATRIX: ICP 2D, SYMMETRIC RESIDUAL; ACCURACY = 0.73

	car	pedestrian	tree	building	trunk	pole	traffic sign
# templates	8	16	2	8	3	2	4

TABLE VII NUMBER OF TEMPLATES GENERATED FROM 3D MODELS IN EACH CLASS; TOTAL = 43.

less fine comparison of the objects' shape. The spin image and the FPFH feature provided similar performance with an accuracy of 0.66 which is 7% lower than the alignment based classification.

C. 3D Models as Training Data

In this set of experiments, the use of templates generated from 3D models was investigated. Template generation is illustrated in Fig. 2. The resulting set of templates is shown in Fig. 1(c) and the final number of templates used per class is indicated in Table VII. While eight scans obtained from eight different angles are generated for each 3D model, not all these scans are kept in the final template set. For instance in the case of objects such as trees which are roughly symmetric with respect to their vertical axis, only one scan is kept. The selection of the final set of templates from the initial eight scans per object is at this stage manual. Future work will investigate the automation of this step.

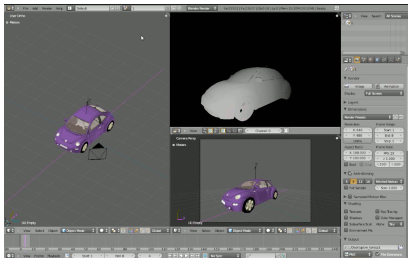


Fig. 2. Illustration of the generation of object templates from 3D models. A 3D model is loaded in Blender environment [1], a virtual range sensor is positioned at eight equally separated yaw angles around the car (left window), and the corresponding range image is computed (upper right window; the bottom right window shows the sensor view). The resulting range images are down-sampled and used as object templates.

1) Global PCA: The classification results provided by the Global PCA feature are detailed in Table VIII (KNN classification, K=1, using as training set the set of templates generated from 3D models; accuracy = 0.61 and the associated standard deviation is 0.03). It is 7% lower than the accuracy obtained with field data as training set.

truth \inferred	car	pedestrian	tree	building	trunk	pole	traffic sign
car	124	2	0	8	0	0	0
pedestrian	0	43	0	4	0	0	0
tree	8	0	**19**	**22**	7	0	0
building	5	0	0	**8**	0	0	0
trunk	2	1	0	1	15	10	3
pole	0	1	0	0	27	16	6
traffic sign	0	3	0	0	19	22	3

TABLE VIII CONFUSION MATRIX: GLOBAL PCA WITH 3D MODELS; ACCURACY = 0.61

2) Spin Images: Spin Images were generated on the 3D model scans and used as training data in KNN classification as before. Compared to the case where training was performed with field scans, a much lower accuracy of 0.46 was obtained (i.e. a decrease of 20%). The confusion matrix in Table IX indicates problems recalling trunks, poles and traffic signs, these being mistaken for pedestrians and buildings. The denser, cleaner web data may generate spin images which are less similar to those produced from real data.

3) FPFH: Likewise, FPFH features were used in this experimental setup, with a similarly low accuracy of 0.41, corresponding to a decrease of 25% with respect to the case where training was performed with field scans. The confusion matrix in Table X shows similar problems to the spin image confusion matrix.

truth \inferred	car	pedestrian	tree	building	trunk	pole	traffic sign
car	92	30	0	4	0	0	0
pedestrian	0	42	0	1	0	0	1
tree	4	0	32	14	0	0	0
building	3	0	1	11	0	0	0
trunk	1	18	0	10	0	2	0
pole	1	23	0	14	0	4	0
traffic sign	0	43	0	4	0	0	0

TABLE IX CONFUSION MATRIX: SPIN IMAGES WITH 3D MODELS; ACCURACY = 0.46

truth \inferred	car	pedestrian	tree	building	trunk	pole	traffic sign
car	82	33	4	4	3	0	0
pedestrian	0	42	1	1	0	0	0
tree	0	19	27	1	3	0	0
building	4	0	0	11	0	0	0
trunk	0	10	5	15	1	0	0
pole	1	9	1	27	4	0	0
traffic sign	0	29	1	14	3	0	0

TABLE X CONFUSION MATRIX: FPFH WITH 3D MODELS; ACCURACY = 0.41

4) Classification by Template Alignment: The same four test sets used in the previous experiments are re-used here and classified using the templates shown in Fig. 1(c). The corresponding classification results are given in Table XI (the accuracy standard deviation is indicated in parenthesis) and the computation times in Table XII (including alignment and symmetric residual computation for the comparison to 43 templates; Python implementation on 2.2GHz Intel Xeon processors). The confusion matrix for the combination ICP 2D/symmetric residual, which as in the previous experiments provides the best performance, is given in Table XIII.

The confusion matrix in Table XIII show high performance for the classes car, pedestrian, tree; precision and recall values: car (0.93, 095), pedestrian (0.98,0.73), tree (0.75, 0.86). This is consistent with the results in Table VI generated with field scans as templates. The performance with respect to the building class is worse; precision and recall: (0.54, 0.78); it is mostly confused with trees, due to the similar geometrical extents between the two classes in the datasets. The building class performed poorly as seen in the results in Table VI, which points to the fact that a global shape alignment mechanism may not be appropriate for larger scale objects (such as buildings). The local methods described in Sec. II may achieve better performance for those larger

	ICP 3D	ICP 2.5D	ICP 2D	no ICP
ICP residual	0.42 (0.07)	0.45 (0.07)	0.44 (0.13)	x
symmetric residual	0.61 (0.09)	0.71 (0.06)	0.73 (0.07)	0.61 (0.08)

TABLE XI CLASSIFICATION ACCURACY: 3D MODELS AS TEMPLATES

	car	pedestrian	tree	building	trunk	pole	traffic sign
ICP 3D	1.2	1.0	2.7	6.6	0.7	0.7	0.7
ICP 2.5D	0.8	0.7	2.0	4.6	0.5	0.5	0.4
ICP 2D	0.7	0.6	1.8	4.6	0.4	0.5	0.4

TABLE XII COMPUTATION TIMES (IN MINS)

truth \inferred	car	pedestrian	tree	building	trunk	pole	traffic sign
car	125	8	1	0	0	0	0
pedestrian	0	46	1	0	0	0	0
tree	4	0	38	2	7	0	0
building	2	0	4	7	0	0	0
trunk	0	4	0	0	12	7	9
pole	0	3	0	0	14	14	19
traffic sign	0	2	0	0	6	8	31

TABLE XIII CONFUSION MATRIX: ICP 2D, SYMMETRIC RESIDUAL, 3D MODELS AS TEMPLATES; ACCURACY = 0.73.

objects. This will be further investigated in future work. As in the case of the results in Table VI, the lower performance with respect to classes trunk, pole and traffic sign is due to the similar 3D geometrical appearance of these three classes of objects. When these three classes are gathered into one class, the overall accuracy increases to 0.90.

The confusion matrices in Table VI and XIII correspond the same accuracy (0.73) and to similar precision and recall values. This experimentally shows that the combination ICP 2D/symmetric residual can provide maintained performance while the training set is entirely replaced by templates generated from web data.

The results in Table XI show that this is not only the case for the combination ICP 2D/symmetric residual but also for the other three ICP variants tested in this study. In addition to maintaining the performance in terms of accuracy, the 3D model based classifier is significantly faster; as shown in Table XII. Averaged across the seven classes considered here, it reduces computation times by 77%[2], which is simply due to the smaller size of the template set: the number of templates is reduced from 251 to 43, i.e. a reduction of 82%.

This leads to the following more general remarks. *Classification of 3D scans of a given environment can be carried out without having previously scanned this environment at all* by using scans of 3D models as templates. This can be done while potentially achieving performance on par with a classifier trained with actual sensor samples from the field. A possible use case scenario of the alignment based classification system would involve the user choosing the objects to be identified by selecting 3D models found on the Internet and passing them on to the system.

The combination KNN/Global PCA (K=1) achieves an accuracy of 0.61, while as mentioned above, the 3D alignment based classifier achieves an accuracy of 0.73. Not considering the classes trunk, pole and traffic sign, the main difference between Table VIII and XIII lies in the performance with respect to the classes tree and building (as indicated in bold in Table VIII). Due to the number of points in the canopy of a tree, its extent is dominated by the extent of its canopy, which makes it closer, in the Global PCA feature space, to a building. (In our dataset, buildings are only observed as sections of wall). On the other hand, local variations of shapes are better captured by the symmetric error metric (Eq. 3), explaining the improved performance with respect to the class tree in Table XIII.

[2]The authors are aware that the computation times are evaluated on two different machines but assume that the difference between the two machines is small enough not to affect the overall trend showed by the computation times.

The alignment based classifier also improved on the Spin Image and FPFH based classifiers by 27% and 32%, respectively. By applying a global alignment constrained with the position of the ground, it is less sensitive to local shape differences and is able to maintain performance given non field samples as training data.

VI. CONCLUSION

There are two main outcomes to this work. First, the use of shape alignment techniques for classification in robotics applications involving sparse 3D sensing. This approach is able to leverage the representation of the ground in the segmentation to constrain the alignment of 3D segments to a 2D process. Second, experimental results indicating that field samples may not be required for the training of 3D classifiers, in particular when the classifier is implemented as a global alignment process. This has the potential to change the way the training of 3D classifiers is approached.

VII. ACKNOWLEDGEMENTS

This research was supported in part by the Australian Centre for Field Robotics (ACFR), by the Centre for Intelligent Mobile Systems (CIMS), funded by BAE Systems as part of an ongoing partnership with the University of Sydney and by the New South Wales State Government.

REFERENCES

[1] Blender open source 3d software. http://www.blender.org.
[2] B. Amberg, S. Romdhani, and T. Vetter. Optimal step nonrigid icp algorithms for surface registration. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.
[3] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel. On the Segmentation of 3D LIDAR Point Clouds. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2011.
[4] B. Douillard, J. Underwood, V. Vlaskine, A. Quadros, and S. Singh. A Pipeline for the Segmentation and Classication of 3D Point Clouds. In *Proc. of the International Symposium on Experimental Robotics (ISER)*, 2010.
[5] A. Dubrovina and R. Kimmel. Matching shapes by eigendecomposition of the Laplace-Beltrami operator. In *Proc. 3DPVT*, volume 2, 2010.
[6] W. Garage. Robotic Operating System (ROS). http://www.willowgarage.com/pages/software/ros-platform, 2011.
[7] A. Halma, F. ter Haar, E. Bovenkamp, P. Eendebak, and A. van Eekeren. Single spin image-ICP matching for efficient 3D object recognition. In *Proceedings of the ACM workshop on 3D object retrieval*, pages 21–26. ACM, 2010.
[8] A. Johnson. *Spin-Images: A Representation for 3-D Surface Matching*. PhD thesis, Carnegie Mellon University, 1997.
[9] K. Lai and D. Fox. Object Recognition in 3D Point Clouds Using Web Data and Domain Adaptation. *The International Journal of Robotics Research*, 2010.
[10] Y. Liu, H. Zha, and H. Qin. Shape topics: A compact representation and new algorithms for 3d partial shape retrieval. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference*, volume 2, pages 2025–2032. IEEE, 2006.
[11] F. Moosmann, O. Pink, and C. Stiller. Segmentation of 3D Lidar Data in non-flat Urban Environments using a Local Convexity Criterion. In *IEEE Intelligent Vehicles Symposium*, pages 215–220, 2009.
[12] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proc. 3DIM*, pages 145–152, 2001.
[13] R. Rusu, G. Bradski, R. Thibaux, J. Hsu, and W. Garage. Fast 3d recognition and pose using the viewpoint feature histogram. In *International Conference on Intelligent Robots and Systems*, 2010.
[14] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *Proc. of the International Conference on Computer Vision (ICCV)*, 2011. To appear.
[15] J. Tangelder and R. Veltkamp. A survey of content based 3D shape retrieval methods. *Multimedia Tools and Applications*, 39(3):441–471, 2008.

Detecting and Localising Multiple 3D Objects: A Fast and Scalable Approach

Dima Damen[1], Andrew Gee[1], Andrew Calway[1,2], Walterio Mayol-Cuevas[1,2]
[1] Department of Computer Science, University of Bristol, BS8 1UB, Bristol, UK
[2] Bristol Robotics Laboratory, BS16 1QD, Bristol, UK
{damen, gee, andrew, wmayol}@cs.bris.ac.uk

***Abstract*— Object detection in complex and cluttered environments is central to a number of robotic and cognitive computing tasks. This work presents a generic, scalable and fast framework for concurrently searching multiple rigid texture-minimal objects using 2D image edgelet constellations. The method is also extended to exploit depth information for better clutter removal. Scalability is achieved by using indexing of a database of edgelet configurations shared among objects, and speed efficiency is obtained through the use of fixed paths which make the search tractable. The technique can handle levels of clutter of up to 70% of the edge pixels when operating within a few tens of milliseconds, and can give good detection rates. By aligning our detection within 3D point clouds, segmentation and object pose estimation within a cluttered scene is possible.**

Results of experiments on the challenging case of multiple texture-minimal objects demonstrate good performance and scalability in the presence of partial occlusions and viewpoint changes.

I. INTRODUCTION

Detecting real objects under clutter has been one key problem underpinning a series of tasks in cognitive computing and robotics. One of the key complexities arises after clutter and object viewpoints are taken into account, and dealing with these has resulted in a series of techniques using both 2D and 3D information. A more serious competence is being able to learn extra objects as the system explores more about the world, calling for methods that allow for scalability without a severe penalty on processing time.

Consider the case shown in Figure 1 where multiple texture-minimal but known objects are to be detected, and their pose estimated within scenes of a complex cluttered nature. In some cases, as it is here, it is conceivable that the areas over which the process has to operate can benefit from some level of prior knowledge, and therefore background subtraction could be used. However, this still results in non perfectly segmented regions, multiple object occlusions and novel views to be dealt with.

This paper proposes a method that uses constellations of 2D edgelets as the representation. The method is generic, scalable and fast for concurrently searching for multiple rigid texture-minimal objects. Scalability is achieved by using indexing of a database of edgelet configurations shared among objects, and speed efficiency is obtained through the use of fixed paths which make the search tractable. By aligning our 2D edgelet configurations with a corresponding trained point cloud, object pose estimation within a cluttered scene is also possible.

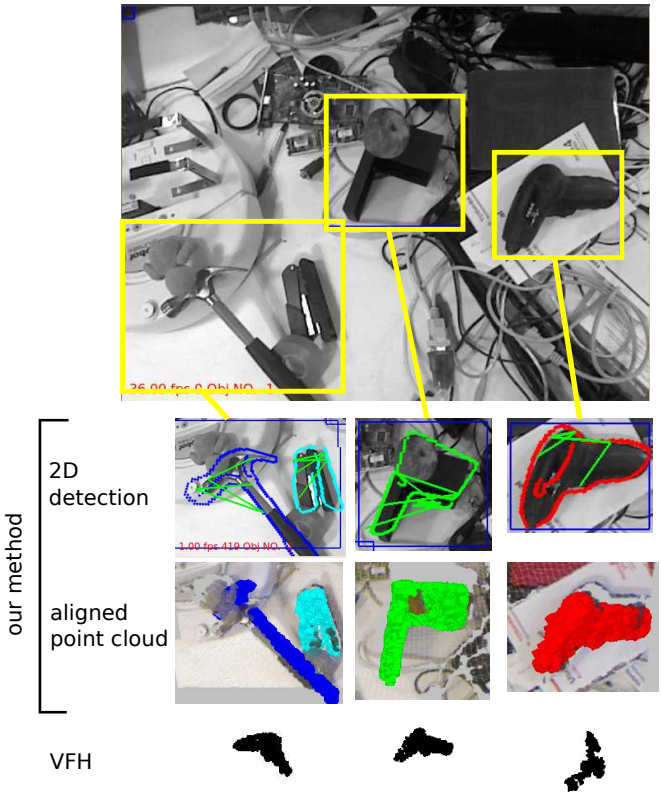


Fig. 1. Using depth background subtraction, three search regions are considered. Our 2D detector along with an aligned point cloud are compared to the best-matched training cluster using VFH object descriptor. When the cluster contains occluded or touching objects, our 2D detector is still able to detect one or more objects.

We have carried out a number of experiments with our approach for several key aspects that include detection performance, framerate expectation, scalability as the object library increases and clutter handling. We then contrast our approach to a state-of-the-art 3D object detector (VFH) [14] where we exemplify how our method can deal with larger levels of occlusion and clutter handling.

II. DETECTING TEXTURE-MINIMAL OBJECTS

There have been many previous approaches to shape matching for textureless/texture-minimal rigid objects and so we focus here only on some directly relevant works. In [9] chamfer distance matching is improved and made faster by

using 3D distance transforms and directional integral images. Detection time of 710ms with 300 reference views are reported. This work uses a search strategy where time increases linearly with more views or with more objects. Such methods are being used in robotics platforms (e.g. [10]).

In [7], image patches represented by histograms of dominant orientations are matched by efficient bitwise operations. This enables detection within 80ms of one object using a rich, 1600 reference views. However, the representation is not rotation or scale invariant (which can be indeed addressed by more reference views). Importantly, the complexity increases with multiple objects, with detection time increasing to 333ms for 3 objects. Wiedemann *et al*. [17] organise object views into hierarchies. Exhaustive search over discrete scales and rotations gives detection times within 300-900ms for a single object. Separate hierarchies are required for each object, giving linear increase in complexity for the case of multiple objects.

There are many techniques which make use of relationships between edge features either within local neighbourhoods to create features for classification [2], [11] or amongst constellations of edgelets located around the contour [3], [4], [5], [8], [12]. For example, Carmichael *et al*. [2] train weak classifiers using neighbourhood features at various radii for detecting wiry objects like chairs and ladders. In [4], consistent constellations of edge features over categories of objects are learnt from training views and used for recognition via exhaustive search over test images. Although these methods demonstrate impressive detection performance, they are not designed for the fast operation we aim to achieve, and are geared towards single object search, with complexity scaling linearly when multiple objects need to be detected.

Scalability to multiple objects was addressed in earlier works by the use of indexing and geometric hashing, similar in form to the library look up that we use in our method. Examples include early work by Lamdan and Wolfson [18] and Grimson [6], and later by others [13], [1]. Of particular note is the work of Beis and Lowe [1]. They use descriptors to represent the relative position and orientation of groups of adjacent line segments which are then searched using a kd-tree for recognising 3-D objects. Our method can be seen as building on these techniques, incorporating the discrimination benefits of constellations of edgelets and the novel use of fixed path configurations to give tractable library look up. Details of the method are given in the next section.

III. DETECTION USING EDGELET CONSTELLATIONS

Our detection of a 3D object is based on the object's shape when seen from different vantage points around the viewing sphere. We refer to the edge map as seen from one point on the viewing sphere as a **view** of that object. The views can be retrieved from 2D images around the viewing sphere or a CAD model. For simplicity, we do not distinguish between different views and different objects, as the aim is to jointly detect and localise the object. The next section explains the chosen descriptor for a constellation of edgelets extracted from the edge map, and Section III-B describes how two constellations are matched. Section III-C then discusses using fixed-paths for extracting similar constellations, and Section III-D explains the detection in cluttered environments.

A. Edgelet Constellations

From the view's 2D image, a set of edgelets $\{e_i\}$ are extracted where an **edgelet** is a short straight segment, represented by its centre point (pos) and orientation (ori). The edgelets are sampled from straight lines with a maximum length. A **constellation of edgelets** is an *n-tuple* of edgelets $c = (e_1, .., e_n)$. These can be adjacent, near-by or far apart (Figure 2). Constellations characterise both local parts and global shape by encoding the relative orientations and positions between segments of nearby and distant edges. The vector v_i connects the edgelet e_i with the consecutive edgelet e_{i+1} in the constellation's tuple. For an n-tuple, there are $n-1$ vectors and accordingly $n-2$ angles θ_i where $cos(\theta_i) = (v_i \cdot v_{i+1})/(|v_i||v_{i+1}|)$. The **base angle** θ_0 is the angle between the first edgelet and the first vector. The set of vectors are normalised to a unit vector and θ_0 is used to align with the vertical direction (Figure 2). These normalised and aligned vectors define the **path** connecting the edgelets in the constellation, and is represented by the angles $\Theta = (\theta_0, .., \theta_{n-2})$. In addition to the path, the constellation is described by $\Phi = (\phi_1, .., \phi_{n-1})$ and $\Delta = (\delta_1, .., \delta_{n-2})$ where $\phi_i = \widehat{e_i, e_{i+1}}$ is the relative orientation of consecutive edgelets, and $\delta_i = |v_{i+1}|/|v_i|$ is the relative length of consecutive vectors. The descriptor $f(c) = (\Theta, \Phi, \Delta)$ is translation-, rotation- and scale-invariant. It encodes the joint presence of n-edgelets with certain relative orientations and positions. Notice that the same set of edgelets could result in a different descriptor when the order of edgelets changes in the constellation's tuple. Though this might be perceived as an additional complexity, this complexity is managed by using fixed paths as will be shown in the Section III-C.

B. Matching Constellations

Given a test image, a constellation of test edgelets $c_t = (t_1, .., t_n)$ is represented by the same descriptor f. The test

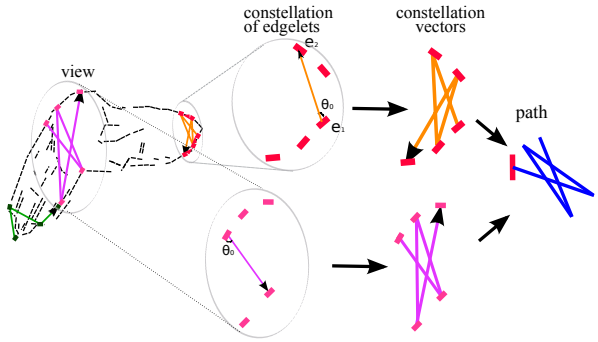


Fig. 2. Constellations of edgelets from one view of an electric screwdriver. The constellations can be local (part-based) or global. The path of the constellation is the normalised angular vectors connecting the constellation's edgelets. The figure shows two constellations of the same path.

constellation c_t matches a view constellation c_v if $\forall i; |\theta_i^v - \theta_i^t| < T_\theta|$, $|\phi_i^v - \phi_i^t| < T_\phi$ and $|\delta_i^v - \delta_j^t| < T_\delta$. When a match is found, an affine transformation H is estimated from the corresponding edgelets, and the remaining edgelets from the view ω are mapped to their corresponding positions and orientations in the test image. An affine homography can be estimated for any constellation for which $|c| \geq 4$ (considering positions only and assuming the edgelets are not co-linear). Iterative closest edgelet (ICE) can then be used to refine the alignment where the closest edgelet to the transformed e_i using the homography H is $\tau(e_i, H)$, i.e.

$$\tau(e_i, H) = \begin{cases} \arg\min_{t_j} d(H(e_i), t_j) & d(H(e_i), t_j) < \beta \\ null & otherwise \end{cases} \quad (1)$$

where a distance measure between two edgelets $d(e_i, e_j)$ assesses the similarity in orientation (using L-1 norm) and spatial closeness (using L-2 norm) [15]

$$d(e_i, e_j) = |e_i.pos - e_j.pos|_2 + \lambda |e_i.ori - e_j.ori| \quad (2)$$

In Equation 2, λ weights the orientation term. The cost of the detection E, using the refined homography H' after ICE convergence, is the scaled sum of the distance measures between corresponding edgelets.

$$E(\omega, H') = \frac{\sum_i \min(d(H'(e_i), \tau(e_i, H')), \beta)}{|\omega|} \frac{|\{\tau(e_i, H')\}|}{|\{e_i; \tau(e_i, H') \neq null\}|} \quad (3)$$

Here the distance measures are summed along with a penalty measure for missing correspondences β, and the scale is estimated by the number of edgelets in the test image to the number they correspond to from view edgelets. An object is then detected at edges $\{\tau(e_i, H')\}$ if $E(\omega, H') < \alpha$.

C. Edgelet Constellations over Fixed Paths

An exponential number of constellations is present in each training and test image. To manage the matching complexity, we adopt a fixed-path approach. All constellations over the same path Θ_k within a tolerance ϵ in the angles, and starting from each edgelet, are found in each view (see Figure 3). This is applied to all views and all objects. For a given Θ_k, the unique descriptor $f^{\Theta_k}(c_i) = (\Phi, \Delta)$ distinguishes different constellations with the same path. This forms an indexed library, where the hash key is the descriptor f^{Θ_k} of size $|(\Phi, \Delta)| = 2n - 3$, and the value is the corresponding view and the n-sized tuple of edgelets that generated the descriptor. Given the training objects, the search is for paths that can find *enough* constellations in the different views. We randomly select paths and rank them by the number of constellations found in all training views. In the experiments next, we use up to 6 paths from 100 randomly chosen path angles. We set the number of edgelets in each constellation to 5 based on preliminary tests. Shorter tuples have a higher chance of hallucinating detections while longer tuples decrease the recall. By keeping a comprehensive library of all constellations of path Θ_k, it is sufficient to extract one constellation of the same path from the object in the test image to produce a candidate detection that is verified using the rest of the view edgelets. A separate library is built for each chosen fixed path, and a quantised hash-table is created so a descriptor would directly lead to candidate matches.

D. Cluttered Environments

For each fixed-path Θ_k, all pairs of edgelets with a relative position that satisfies the base angle θ_0 in Θ_k (within the tolerance ϵ) are highlighted. A pair is chosen at random, and the search for edgelets that complete the fixed-path is carried out (Figure 4). This search is speeded up by pre-calculating relative measurements between all pairs of edgelets. As the tuple is appended, the descriptor f^{Θ_k} is incrementally calculated and compared to the corresponding library. When the partial descriptor cannot match any descriptor in the library, the search is prematurely stopped, and another pair is pursued. To speed up detection only one constellation is pursued from each starting pair of edgelets. Given that the library contains a comprehensive list of all possible view constellations of path Θ_k, the risk of skipping a pair before pursuing all the constellations starting with that pair is acceptable, and proves sufficient during the experiments. When a test constellation matches a view constellation with an error $E(\omega, H') < \alpha$ (Equation 3), the test edgelets $\{\tau(e_i, H')\}$ are explained edgelets based on this detection. These edgelets are removed from any further searches. If all pairs are tested, another fixed path Θ_2 is used. For k fixed paths, the worst case is $O(k \cdot p^2)$. A further modality of handling clutter exists when some prior scene knowledge is available where a depth sensor is used as explained in Section IV-D.

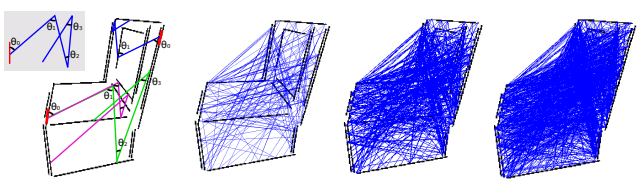


Fig. 3. For a given path (upper left), two constellations from the same starting edgelet, and one from a different starting edgelet are shown (left). All constellations found with $\epsilon = 0.01, 0.015, 0.02$ are also shown.

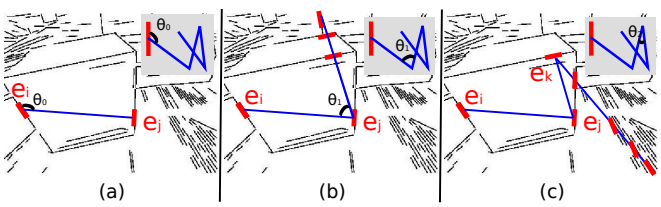


Fig. 4. For a pair of edgelets (e_i, e_j) that satisfy the base angle θ_0 (a), an edgelet e_k that completes the path is sought. When several edgelets are found (b), one is selected and the path is completed (c).

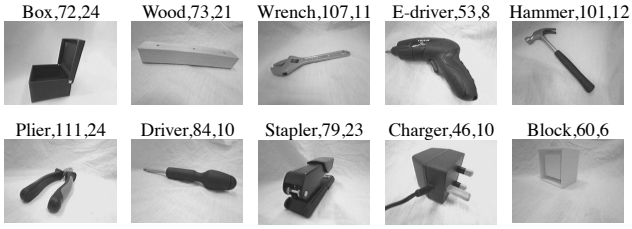


Fig. 5. One view from each of the 10 textureless objects in the dataset with (name, # of views, # of occurrences in test images)

IV. EXPERIMENTS AND RESULTS

We have tested the method on a dataset consisting of 10 real tools and components. Training images were coarsely sampled around the viewing hemisphere (Figure 5) To extract edgelets, we use the line-segment detector in [16]. The edgelet length is set to 10 pixels and ϵ is set to 0.02 (see Figure 3).

Note that due to the initial random order of pairs of test edgelets, and pursuing one path from each pair, detection performance can vary between runs. This is illustrated in Figure 6 which shows the results from 5 runs on a single image along with the edge map reflecting the complexity of the search. In the figure, the block was detected in 4 out of 5 runs while the stapler was detected in 3 out of 5.

A. Detection performance

We first compare the detection performance for the different tools (Figure 7), discarding the false-negative effect from the greedy removal of edgelets. This is evaluated using 100 ground-truthed test images with one to four objects per image. The PASCAL overlap criterion is used to evaluate the detections. The figure shows very low recall for the driver (yellow) and high false positive rate for the wood (red). This is because the wood's rectangular shape can often match other rectangular objects in the background. The e-driver, plier, wrench and stapler achieve the best detection results due to their distinguished shape. The box and the block are ambiguous which increased their FPPI.

B. Expected framerate

We ran the detector at multiple frames per second on a 300 frames video sequence containing 6 out of our 10 objects with surrounding clutter. In this case each frame is analysed from scratch, without considering the detections from previous frames. Figure 8 plots the recall and precision as the frame rate increases from 1 to 17fps. This is run using a non-optimised C code on a standard laptop machine (2.53Hz, 6GB DDR3 SDRAM). Detection can achieve above 50% recall at about 7fps while searching for the 10 textureless objects. The same performance can be achieved at 11fps when searching for a single object.

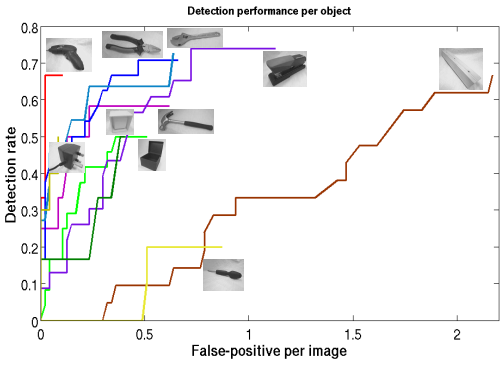


Fig. 7. FPPI versus Detection rate for the tools dataset

For a system that runs on a stream of live images, it is affordable if an object is missed in one frame as long as it can be detected in a few subsequent frames. Figure 9 reports a histogram of the number of frames between correct detections when running at 5fps. For each video, the object to be detected was present in all frames, either in full-view or partially occluded. This was tested for all the 10 objects, and an average of 65% of detections were found within 3 processed-frames from a previous correct detection. This goes up to 86% for the plier and drops to 33% for the driver.

C. Scalability

For the detector to be considered scalable, we expect that as the number of objects and views increases, the detection time scales gracefully. A single non-cluttered test image is selected for each object, and the test focuses on how the performance is affected as the library size increases. Figure 10 plots the average (and standard deviation) detection time from 100 runs compared to the library size as the number of objects in the library increases from 1 to 10. The

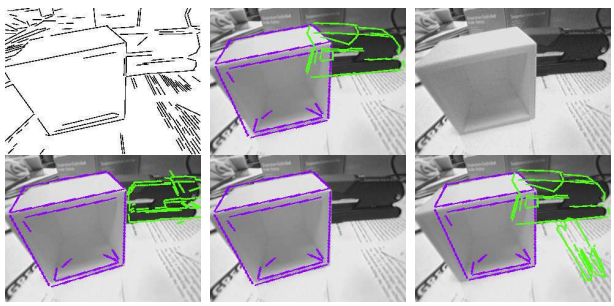


Fig. 6. Five runs on a single image (limited to 200ms). False negatives are caused by not selecting a constellation that belongs to the object.

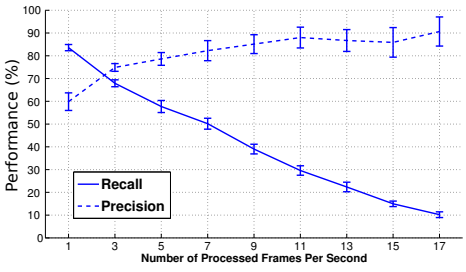


Fig. 8. As the maximum time limit decreases, the recall and precisions are plotted for a cluttered multi-object sequence.

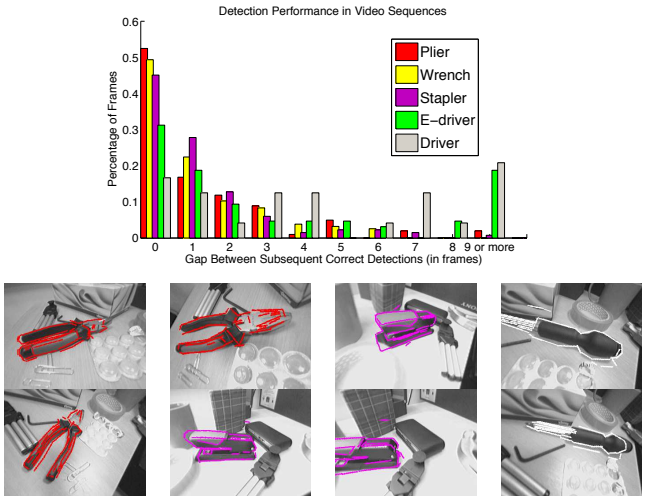


Fig. 9. Histogram of missed number of frames between subsequent detections for five objects, along with frames showing sample detections from the video sequences.

increase in detection time results from comparing to a larger number of descriptors within each indexed bin, as well as assessing ambiguous matches. From the figure, the driver has a nearly linear average time, while the increase in time for detecting the wrench is 2.7x as the library size increases by 10x.

D. Clutter handling and using a depth sensor

The other factor affecting the method's performance is the increase in clutter. Figure 11 is plotted from three video sequences, each starting with the object alone on the desktop then more objects are added. The detector is run at 5fps and the figure shows that the approach can tolerate clutter increased up to 70% of the edge pixels. For more complex environments with a higher percentage of clutter, other techniques can be used to retrieve search regions within which objects of interest can be found. One method to get such search regions is to use depth background subtraction. During training, the point cloud, as retrieved from an RGB-D sensor, is saved as background prior to the introduction of objects. At run-time, the segmented point cloud is used to highlight bounding boxes containing interesting objects. All edges within the bounding box are considered for possible edgelet constellations based on the fixed paths. In our tests the background segmentation is not perfect due to small

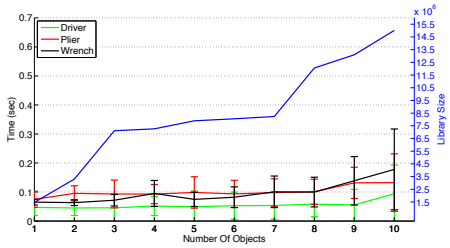


Fig. 10. As the library size increases by 10x (blue line), the average detection time increases by 1.7x-2.7x for 3 objects.

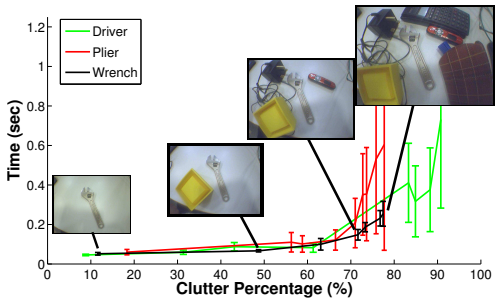


Fig. 11. The average detection time as the percentage of clutter increases for different objects. Sample images from the 'wrench' sequence are shown.

changes in the scene and also addition of non-class clutter objects. Figure 12 shows examples of detections in complex environments using depth background subtraction.

The 2D detections can also be used to align the object within the cloud point. During training, an RGB-D sensor is used to combine each view with a segmented point cloud (Figure 13). After the object is detected, the in-plane homography is used to rotate the corresponding model's 3D point cloud around its centre of mass. The model point cloud is then overlaid on the scene's point cloud with the depth estimated from that of matched scene's points.

We compare this with a 3D object descriptor; the viewpoint feature histogram (VFH) [14]. This descriptor is dependant on obtaining a separate cluster of points containing the object. For each cluster, a 308-D descriptor is calculated and compared to a previously calculated list of descriptors for all training models from various views. The histogram intersection distance is used to compute the mismatch between the descriptors, and the returned point cloud is that of the

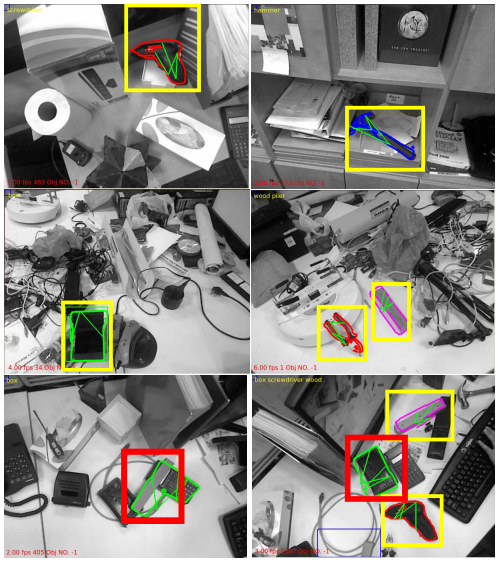


Fig. 12. Detection in complex scenes using depth background subtraction. Bounding boxes show a search region and all edges within the region are considered for the detection. Yellow indicates a correct detection while red indicates a false positive.

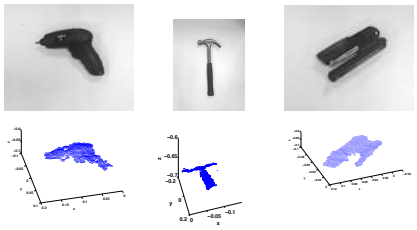


Fig. 13. Using the RGB-D sensor, views are sampled along the hemisphere. The 2D image (above) and the accompanying point cloud segmented from the background (bottom) are shown for one view of three objects.

lowest mismatch. The clustering - and thus the descriptor - is easily corrupted by the presence of occluding or touching objects. Figure 14 compares our detector to the retrieved best-matched model using VFH descriptors on 3 clusters. The VFH descriptors were calculated using the available implementation within Point Cloud Library (PCL). While VFH is able to correctly retrieve the object in the case of the occluded screwdriver, it is confused in the two other cases. Also in the first correctly retrieved point cloud, the mismatch score doubles as the point cloud is occluded by another object. Similar results are shown in Figure 1.

V. CONCLUSION AND FUTURE WORK

We have presented a framework for scalable detection of objects using constellations of edgelets. The constellations are tractable using fixed paths, and are described using translation-, scale- and rotation-invariant descriptors. By using edgelet constellations, the method is more robust to occlusions and camouflaged edges. The method was tested on a dataset of textureless real tools and objects. The technique allows concurrent detection of multiple objects each with views around the viewing sphere. A 50% chance of detection under clutter can be achieved within 140ms. The system can handle up to 70% visual clutter and scales gracefully as the number of objects increases.

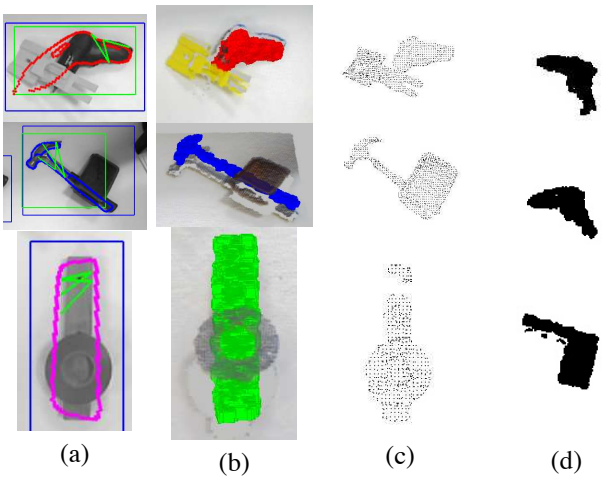


Fig. 14. Our detections (a) can correctly localise the model's point cloud (b) even when the clusters (c) contain occluding or touching objects. The best matched model using VFH descriptors (d) fails to correctly detect the object in two out of the three cases.

As the scene's complexity increases, bounding boxes representing search regions are used to search for objects of interest. In this work, depth background subtraction is used for retrieving such search regions. Other approaches can be alternatively used, like colour or point cloud priors, and these are left for future work.

As opposed to some current 3D object descriptors, this method does not require object clusters to be cleanly segmented. When the object is occluded or is touching neighbouring objects, the path-based detector can still detect and align the model's point cloud successfully.

Acknowledgement We would like to thank Pished Bunnun for his valuable input on the method and the C++ implementation. This work was supported by the EU Project COGNITO FP7-ICT-248290.

REFERENCES

[1] J. Beis and D. Lowe, "Indexing without invariants in 3D object recognition," *IEEE Transactions on Pattern And Machine Intelligence (PAMI)*, vol. 21, no. 10, 1999.

[2] O. Carmichael and M. Hebert, "Object recognition by a cascade of edge probes," in *British Machine Vision Conference (BMVC)*, 2002.

[3] A. Chia, S. Rahardja, D. Rajan, and M. Leung, "Object recognition by discriminative combinations of line segments and ellipses," in *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[4] O. Danielsson, S. Carlsson, and J. Sullivan, "Automatic learning and extraction of multi-local features," in *International Conference on Computer Vision (ICCV)*, 2009.

[5] V. Ferrari, F. Jurie, and C. Schmid, "From images to shape models for object detection," *International Journal of Computer Vision (IJCV)*, vol. 87, no. 3, 2009.

[6] W. Grimson and D. Huttenlocher, "On the sensitivity of geometric hashing," in *International Conference on Computer Vision (ICCV)*, 1990.

[7] S. Hinterstoisser, V. Lepetit, S. Ilic, P. Fua, and N. Navab, "Dominant orientation templates for real-time detection of texture-less objects," in *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[8] M. Leordeanu, M. Hebert, and R. Sukthankar, "Beyond local appearance: Category recognition from pairwise interactions of simple features," in *Computer Vision and Pattern Recognition (CVPR)*, 2007.

[9] M. Liu, O. Tuzel, A. Veeraraghavan, and R. Chellappa, "Fast directional chamfer matching," in *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[10] D. Meger, M. Muja, S. Helmer, A. Gupta, C. Gamroth, T. Hoffman, M. Baumann, T. Southey, P. Fazli, W. Wohlkinger, P. Viswanathan, J. Little, D. Lowe, and J. Orwell, "Curious George: An integrated visual search platform," in *Canadian Conference on Computer and Robot Vision (CRV)*, 2010.

[11] K. Mikolajczyk, A. Zisserman, and C. Schmid, "Shape recognition with edge-based features," in *British Machine Vision Conference (BMVC)*, 2003.

[12] A. Opelt, A. Pinz, and A. Zisserman, "A boundary-fragment-model for object detection," in *European Confernece on Computer Vision (ECCV)*, 2006.

[13] C. Rothwell, A. Zisserman, D. Forsyth, and J. Mundy, "Canonical frames for planar object recognition," in *European Conference on Computer Vision (ECCV)*, 1992.

[14] R. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3D recognition and pose using the viewpoint feature histogram," in *Intelligent Robots and Systems (IROS)*, 2010.

[15] J. Shotton, A. Blake, and R. Cipolla, "Contour-based learning for object detection," in *International Conference on Computer Vision (ICCV)*, 2005.

[16] R. G. von Gioi, J. Jakubowicz, J. Morel, and G. Randall, "LSD: A line segement detector," *IEEE Trans. on Pattern and Machine Intelligence (PAMI)*, vol. 32, no. 4, 2010.

[17] C. Wiedemann, M. Ulrich, and C. Steger, "Recognition and tracking of 3D objects," in *DAGM Symposium on Pattern Recognition*, 2008.

[18] Y.Lamdan and H.J.Wolfson, "Geometric hashing: A general and efficient model-based recognition scheme," in *International Conference on Computer Vision (ICCV)*, 1988.

Efficient Semantic mapping of Man-made environments using Kinect

Sven Olufs and Markus Vincze

***Abstract*— We propose a novel approach for efficient semantic mapping of Manhattan-like structure i.e. the frequently observed dominance of three mutually orthogonal vanishing directions in man-made environments. First, we estimate the Manhattan-like structure by using an MSAC variant that estimates the Manhattan system directly from the 3D data. In contrast to other methods we use only the normal vectors of each voxel rather than estimating it indirectly using plane estimation. The mapping is done using a geometric constrained ICP using a-priori knowledge on the estimated Manhattan system. The ICP registers only points within the same geometry to each other. We show that the geometric constrained ICP generate maps with a significant smaller angular drift than an unconstrained one. Octrees are used for map representation in combination with kd-Trees for ICP. We show the robustness of our Manhattan-estimation using real world data. In this paper we demonstrate our approach using a Microsoft Kinect, while the approach will work with all kind of 2.5D sensors.**

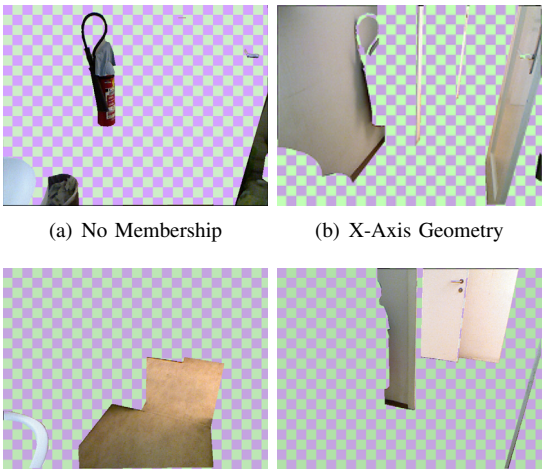
I. INTRODUCTION

The domain of service robotics has become an important and fast growing market in the last decade. While service robotics has become more and more common in the industry domain they are still rare in the home robotics domain. Recent demographic developments in Europe and Japan have shown that there is a need for service robotics in everybody's home due to the elderly society phenomenon. Such robots can be used for the remote surveillance of elderly in the case of an emergency or to help them in their daily life. With the home robotics domain we have usually a relative small area (e.g. $100m^2$ of the owners flat), clutter and visually weakly structured environments.

From the robotics point of view one of the key problems is to estimate semantics from the visually weakly structured environments. The semantics of the room structure is required for efficient user interaction, mapping and navigation and object recognition. Semantic information such as wall, ground, table surface, or door assist in all these tasks. We start from the observation that the structure of many rooms looks the same, e.g. they have a rectangular shape, due to the limited sensing capabilities of todays robotics.

To cope with these environments, the use of 2.5D sensors has become quite popular in the last decade, for instance the use of tilting 3D laser scanners or the Swissranger SR-3000. With the recent release of Microsoft's Kinect structured light sensor, the popularity of 2.5D sensors gained a boost. The Kinect sensor is suitable for the task for two reasons:

Sven Olufs (olufs@ieee.org) and Markus Vincze (vincze@acin.tuwien.ac.at) are with the Vienna University of Technology, Automation and Control Institute, Gusshausstrasse 25-29 / E376, A-1040 Vienna, Austria



(a) No Membership (b) X-Axis Geometry

(c) Y-Axis Geometry (d) Z-Axis Geometry

Fig. 1. Concept of semantic labels for Manhattan-like structures, using figure 2(a) as sample.

The sensors are cheaper than laser scanners and its offers a depth image at frame rate. The challenge with data from 2.5D data is to cope with noise and uncertainty due to the nature of the sensors. For instance, the quality of 2.5D data from the Kinect depends on the reflection properties of the observed surface and the angle of incidence (assuming Lambert surfaces). Since the sensor uses structured light in the infrared spectrum, the sensor is sensitive to sunlight.

Many approaches for room structure estimation use the concept of occupancy grids [1] or its extensions to 3D, e.g. [2]: The grid contains information on a primitive level if a grid cell corresponds to a wall or ground. At this level, there is no semantic information if certain parts of grid cells with the label "wall" are aligned to other "walls" or if the ground is parallel to other structures, e.g. a table top. This kind of constraints is referred to in the computer vision literature as the so-called *Manhattan world* assumption; The frequently observed dominance of three mutually orthogonal vanishing directions in man-made environments [3], [4], [5], [6]. Many indoor environments can be considered as Manhattan-like since most walls of a room are aligned orthogonally to the ground or quasi Manhattan-like if the walls are not aligned orthogonally to each other. In many cases, furniture is also aligned Manhattan-like to its environment, e.g. a couch or cupboard can be aligned with a wall. Here we emphasize that it is not necessary that the furniture is aligned to all three major axes i.e. even if a table is not aligned to a wall its table surface is usually parallel to the ground.

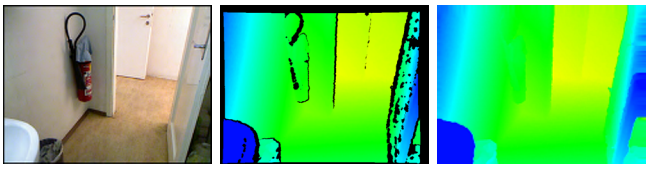
The use of the Manhattan world assumption is useful for home robotics for two reasons, see figure 1: First we can distinguish between Manhattan-like and non Manhattan-like structures. For instance Manhattan-like structures is useful for mapping while the non Manhattan-like structures can be used for object segmentation and classification. The second reason is we use it as description for potential places for objects. For instance objects useful for grasping are usually placed on a vertical structures.
The novelty of the paper is twofold: First an MSAC variant that directly estimates an Manhattan-system based on normal vectors. In this paper we use it to detect the dominant Manhattan system in the image. The second contribution is a geometric constrained ICP method to build maps from Manhattan-like structure if the Manhattan system is known. The method is efficient regarding computational time and memory usage with a complexity of $O(nlogn)$.

In this paper, we propose a novel approach for the robust room structure segmentation using Manhattan world assumption and geometric constrained ICP mapping. The approach estimates first the Manhattan system within the data using the well known RANSAC approach. The main difference to other approaches is that we estimate directly an valid Manhattan like system rather than indirect using individual planes. As a result we always obtain a valid Manhattan-system as model. The mapping using ICP is based on exploiting the geometric constrains within the Manhattan system. In a first step we segment the 2.5D according to its membership of the three axis of the Manhattan system or if it does not belong to it resulting in four labels. The idea is that we only register data to each other with the same label, e.g. "X Axis" to "X Axis" voxels, while we use only one ICP than four individual ones. In this paper we demonstrate our approach using a Microsoft Kinect, while the approach will work with all kind of 2.5D sensors.

The paper is organized as follows: After discussing the state of the art, we describe in section III the proposed approach in two major parts: The first part describes (III-A to III-B) the estimation of the Manhattan system within the 2.5D data. The second part (III-C) presents an geometric constrained ICP for mapping. Next we present experimental results. Finally, we give the conclusion in section V.

II. RELATED WORK

Mapping is a very well understood area in the field of robotics. A common approach is using registration methods to align point clouds to each other using either *Iterative Closest Point* [7], *Hough Transformation* [8], *expectation maximization* [9], *RANSAC* [10] or *split and merge* [11] techniques. Semantics can be obtained from the dataset 1) in a post processing step of the aligned data or 2) using segmentation techniques on the current dataset before registration. Rusu et al. [2] obtains semantics from a octree oversegmetion in a post processing for safe navigation on



(a) RGB Image (b) Raw Depth Image (c) Depth Diffusion

Fig. 2. Processing of depth data for normal vector estimation.

outside terrain. Triebel et al [9] use hierarchical expectation maximization method to extract planes as semantics from 3D laser range scans. Rusu et al. [12] extended the idea by using parametric descriptors for object primitives using geometric primitives. The primitives are obtained from the point cloud in a kitchen like environment. The semantics are given by the combination of objects within the scene. [10] Gallo et al. uses an hybrid approach of segmentation and post processing. First the planes are estimated in 3D using RANSAC, but uses an additional connected component analysis in 2D to extract coherent planes in the data i.e. coherent in 2D.
Semantics can also be obtained in a pre-processing step before registration of the point clouds. Methods based on segmentation uses either a 2.5D grid of the data or rely on split-and-merge plane strategies based on local homography [11] or by clustering normal vectors [13]. Murarka et al [14] use a parametric plane fit on segmented disparity patches to obtain the observer orientation. The segmentation is based on colour and local homogeneity in the image and on exhaustive graph search. The plane fit is merged to a plane hypothesis using graph cuts and energy minimization [3]. The use of the *Manhattan-World* assumption is quite popular in the computer vision literature, for instance, in the use of multi view-reconstruction [4], [5], [15]. Gallup et al. [4] use *Manhattan-World* assumption as prior for plane sweeping i.e. using only orthogonal planes. Furukawa et al. [15] use a similar approach for reconstructing piecewise planar patches and Markov random field formulation for exact planes. Sinha et al. [5] use a similar method, but with a less strict model. Gupta et al. [6] extend the idea by including not a-priory known kinematics on image structure. Another idea is used by Flint et al. [16] by dividing the Manhattan World assumption into local sub classes.

III. OUR APPROACH

Our approach consists of three steps: First we estimate the dominant Manhattan-room configuration i.e. the roll, pitch and yaw of the observer relative to the Manhattan system. We use normal vectors for each pixel in the 2.5D grid and assign every vector the most plausible Manhattan geometry. As next we remove the rotation of the three axis from the data and segment it according to their geometry membership. The map is generated using standard ICP method while only data within the same geometric are registered to each other.

A. Normal vector estimation

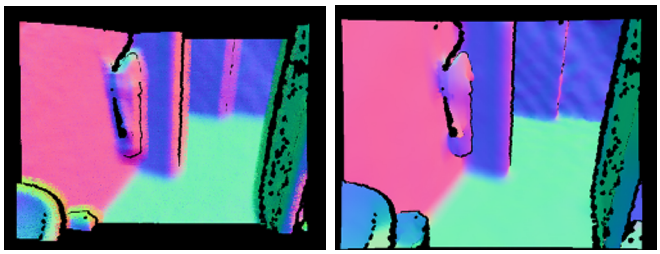
The use of normal vectors has become a defacto standard feature in 3D sensing over the last decades [17], [18], [13].

The usual approach is to calculate a normal vector for each corresponding voxel in the data set. Depending on the type of sensor this task can be computationally demanding, for instance if the data is not aligned in a grid-like structure or if the data is noisy. In the case of 2.5D data, the data is a aligned in an image-like grid structure which enables us to calculate the normal vector in $O(logn)$ runtime. We use a variant of Stefan Holzer method from the Point Cloud Library [18] to calculate normal vectors based on integral images i.e. the 2.5D depth is used as image. The use of integral images allows to calculate an average sum of any rectangular size area with $O(1)$. We extend the method by applying a depth interpolation on missing data voxels in the 2.5D grid, see [19] for details and figure 2(c). Depending on the type of sensor such holes can appear in dark or far surfaces or caused by the different viewpoint of the depth and RGB camera. Note that we use only the interpolated voxels for smooth normal vector estimation of the true (not missing) voxels, we do *not* use them elsewhere in the processing. Another way to calculate normal vectors in an efficient way is proposed by Badino et al. [17] using a constrained least square fit.

Figure 3 shows a comparison of our normal vector estimation and the standard kd-Tree approach using the Point Cloud Library. One can see that the integral image method shows adequate results within planes in respect to normal vectors. The "noise" within these areas is produced by the limited precision of 32bit float values in the integral image. The corners show artifacts because the estimation is done on a dynamic 2D window in the image and not in the 3D state space like with kd-Trees. Another effect is that the integral image method shows a margin around the outer boundaries. This is also caused by the used implementation of the normal vector estimation using integral images and the use of a 2D window (PCL Version 0.9.0).

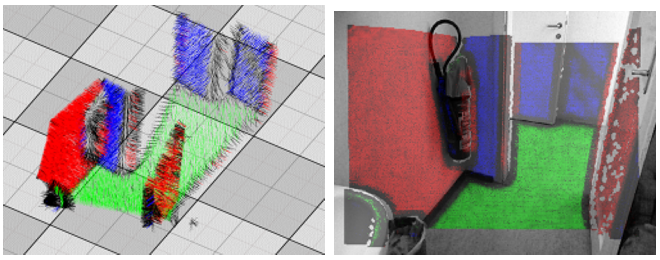
B. Manhattan System Estimation

We propose an alternative approach using a variant of the well known Random Sample Consensus (RANSAC) techniques. RANSAC based methods obtain their estimate by randomly selecting coefficients from a given dataset to a known model. The estimation is iterative, in each iteration



(a) Integral Image (PCL), 100ms (b) kd-Trees 20cm (PCL), 4500ms

Fig. 3. Comparison of different normal vector estimations methods on 2.5D data. While the integral Image method is 45x faster than the standard kd-Tree method, it shows quite some artifacts on depth edges



(a) Normal Vectors (b) Image

Fig. 4. Estimated Manhattan System of a sample scene. The colors indicate the membership to one of the three major axis

the number of inlier is counted. After a fixed number of iterations, the model with the most inlier is used as estimate. The idea is to describe the Manhattan-world as three normal vectors $\vec{N}_1, \vec{N}_2, \vec{N}_3$ one for each axis. We use the normal vectors to express the *orientation* to an axis i.e. the vector is virtually aimed in both directions of the axis. The normal vector of a voxel counts as an inlier, if the angle to one of the three axis normal vectors is within a certain threshold e.g. 5 degrees. The resulting angle is always between 0-90 degrees, since an axis does not have an orientation like a normal vector. The model is given as follows: Let $\vec{A}, \vec{B}_1, \vec{B}_2 \in V$ randomly selected voxels of the 2.5D grid and $\vec{a}, \vec{b}_1, \vec{b}_2$ its associated normal vectors. The three vectors are calculated with:

$$\begin{aligned} \vec{N}_1 &= \vec{a} \\ \vec{N}_2 &= \vec{B}_2 + \vec{a}((\vec{B}_1 - \vec{B}_2) \cdot \vec{a}) - \vec{B}_1 \\ \vec{N}_3 &= \vec{N}_1 \times \vec{N}_2 \end{aligned}$$

The entire concept is depicted in figure 5: The overall assumption is that A is a point on a Manhattan-like structure for instance on the "Z-Plane". B_1 and B_2 are both on corresponding different Manhattan-like structure for instance the "Y-Plane" or "X-Plane". Since the Manhattan system is redundant to one axis, we only need to calculate two axes e.g. in figure 5 the x axis is redundant. The first vector is given by the normal vector of A itself. The second vector is obtained by shifting the first vector to B_1 and using B_2 as "roll" component. The third vector is the cross product of both previous vectors. This approach generates always a valid Manhattan system using three vectors. Note that we do not check in advance if e.g. B_1 and B_2 are on the same plane, since we have no prior information about planes in the 2.5D data at this step. Such "miss configurations" usually generate a Manhattan system with a significant less inliers than a "proper configuration" like in figure 5.

In practice we use MSAC [20] for estimation, an M-Estimator RANSAC variant: Instead of counting inlier within a specific threshold, we accumulate the error of the model from the original data. The MSAC uses a threshold to specify a maximum error that a voxel belong to the model. Since we assume that we have one dominant Manhattan system we adapt this approach. The idea is to use an additional fixed

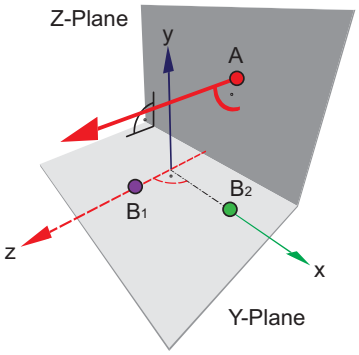


Fig. 5. Estimating the Manhattan configuration using three normal vectors and RANSAC methods: A is used as seed for the Manhattan system for the first axis. B_1, B_2 are used to calculate the "roll" of the second axis and the third axis is redundant. Here we assume that A and B_1, B_2 do not share the same orientation plane of the Manhattan system, but that B_1, B_2 does.

error if an error exceeds the maximum error. In practice the additional fixed error is significant greater than the measured maximum error according to the threshold. This will raise the probability that the MASC will favor the dominant system rather than a valid random one as long the dominant one is also the largest one in the data. In our setup we use 5 degrees as threshold and 10 times as max constant error. The conversion of the three normal vectors to roll, pitch and yaw is straightforward. Figure 4 shows an correct estimated Manhattan system.

C. Geometric Constrained ICP

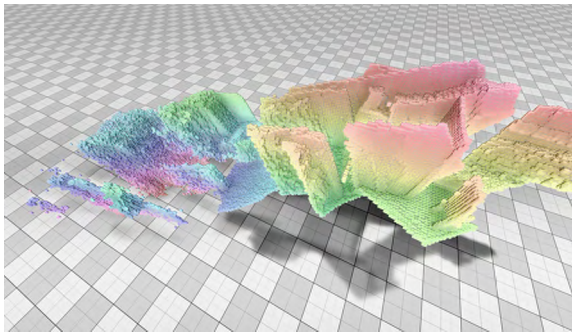
In a first step we remove the rotation of the three major axis from the data i.e. roll, pitch, yaw according to the estimated Manhattan system. Next we segment all voxels according to their membership to a Manhattan geometry, see figure 1. We use for labels i.e. "X Axis", "Y Axis", "Z Axis" and "None" membership. The membership to an axis is estimated using the normal vectors of each point. If a normal vector has almost the same orientation to a major axis within a specific threshold its assigned to that axis. We use a threshold of 5 degree, all normal vectors that have no membership to an axis are assigned with "None".

For the generation of the map we use an modified version of the well known iterative closed Point ICP algorithm. In a nutshell ICP consists of three steps to register a set of samples to a model. First the samples are associated to the model by the nearest neighbor criteria. Then estimate the transformation of the points to the model and finally the input points are transformed. The entire algorithm is repeated to a fixed number of iterations or the error converges. In our variant we associate only samples to the models with the same label. We estimate separate the transformation for translative and rotative part. The translative part is given using Least squares while the rotation is obtained using the Singular Value Decomposition approach. We also estimate the rotative transformation to compensate the drift that is caused by the inverse Manhattan transformation of the first step here. Usually the angular drift is within 1 degree.
One key problem with ICP is initialization of the samples to the model or in our case the map. Since mapping is a continuous process we use a the last transformation of the previous step for initialization for the next one. We do not use a motion model as used in many visual slam or visual odometry approaches.
The map is represented octrees with a fixed minimum resolution which is efficient regarding memory usage and computational power. Since the runtime of ICP depends on the number of input and map (or model) points or voxels its feasible to limit the size of the map instead of representing them in a kd-Tree like structure [7]. In practice we use the ICP with max 20 iterations and a octreemap with a resolution of 2.5cm. Due to we remove the rotation from the 2.5d data, ICP converges within 5-7 iterations in many cases. We use the octree map implementation proposed by Wurm et al. [21].

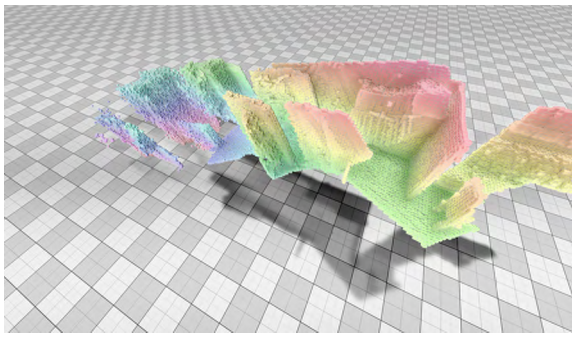
IV. EXPERIMENTAL RESULTS

For our experiments we use a non-holonomic mobile robot manufactured by Bluebotics with a first generation Microsoft Kinect Camera mounted in 120 cm height to the ground. Since the Kinect sensor has an inbuilt servo motor to tilt the camera, we use it to adjust the tilt dynamically according to the traveling speed of the robot. The tilt angle is adjusted such that the Kinect looks at the region the robot will be in 2 s according to the wheel speed. The average angle of the Kinect while moving is 10 degree, if the robot stands still or moves very slowly the camera has an average tilt of 35 degree. We use XSens MTi IMU for ground truth of roll and pitch of the Kinect i.e. its physically attached. Yaw for ground truth is obtained by hand for each frame rather than using the robot self-localization.
We choose a typical domestic environment (see fig. 2(a)) for data acquisition using the Kinect. The data of all sensors is recorded at 25 frames per second. We recorded a representative set of seven tours through our lab with a total length of approximately 180 meters. Three tours have a Manhattan like environment while the other ones represent a quasi Manhattan like environment. The environments were *not* cleaned before acquisition to have a more realistic environments. Since our lab is used for robotics it contains of many boxes which are randomly scattered in the lab i.e. they are not aligned to the dominant Manhattan system.

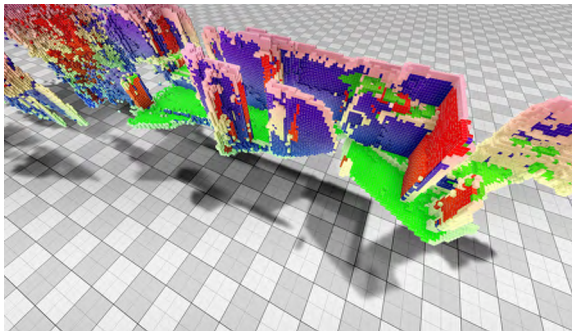
Figure 8 shows the average angle error of the estimation of dominant Manhattan system using a fixed number of iteration for RANSAC, MASC and our MSAC variant for all tours. All methods use the same normal vectors as prior for estimation. RANSAC needs here the most iterations before till error convergence. Our MSAC variant converges at 512 iterations in contrast to the normal MSAC. It is able to find the dominant Manhattan system in the case of multiple local Manhattan systems within the data i.e. a box that is not aligned with a wall (on purpose). The average error after 512 iterations is 0.77 degree for MSAC, 0.12 degree for our MSAC approach and 1.25 degree for



(a) standard ICP (PCL)



(b) standard ICP (PCL) with removed Roll, Pitch and Yaw



(c) our ICP variant

Fig. 6. Generated map after 30s (Frame 193) from figure 7 with 5cm plot cell size for better visibility, we use 2.5cm for processing. The unconstrained ICP shows an angular drift. Best Viewed in Color, height and geometric membership is shown in color-coding

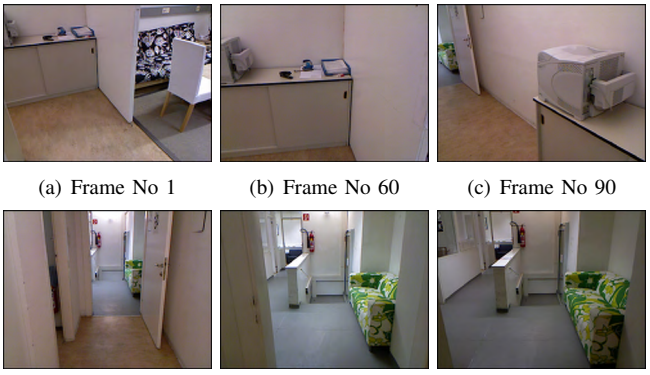
RANSAC. The traditional MSAC converges on the same level as our MSAC approach after 786 iterations, RANSAC after 1536. In general we consider an average error of 2 degree as reasonable for the segmentation which is reached after 192 iterations for our MSAC variant 256 iterations for the traditional one and 384 for RANSAC.

The generated map of a sample tour (after 32m) is shown in figure 6 using ICP with and without geometric constraints. The ICP without the geometric constrains generates maps with the typical angular drift. This is typical for a non-holonomic robots and sensors with a relative short sensing range. Holz et al. [22] states that the angular drift using ICP can be reduced with a large and wide field of view i.e. 360 degree sensing up to 80m. The ICP with geometric constraints shows almost rectangular maps since the Manhattan geometry estimation reduces the angular drift significant. One can see that non Manhattan-like structures are also mapped and segmented out.

The major drawback of our method the segmentation is that it depends on a proper detected Manhattan configuration. While its possible to detect an non-existing Manhattan system, it can not be always guaranteed that room structure is fully orthogonally to each other. The use of a relaxed Manhattan constrain is one possible solution i.e. that the major axis are almost 90 degree orthogonally to each other. Another issue that we encountered are open doors, or slightly open doors. Since we estimate only the dominant Manhattan structure right now, doors are not yet segmented as long they are not closed or aligned to one other major axis. Depending on the visible amount of the door in the data, e.g. more than 20 percent, the door will be segmented into small coherent areas.

Another drawback are artifacts in the map due to wrong classified areas. This is due to the use of normal vectors for segment ion. For instance a wall in a distance of 5m can produce bogus normal vectors due to limitations of the depth sensor resolution, see figure 3, while a wall in 2m shows reasonable normal vectors. Such artifacts does not influence the performance of the Geometric constrained ICP since to we use independent maps for each geometry. If the 5m away wall is closer in the image it will be correctly segmented. The artifacts can be removed from the map, since they are usually not cohered areas in the map. One issue are round objects. They seem to be aligned with to at least two geometries according to their normal vectors. Here the use of a local geometry analysis can reject this kind of artifacts. Another issue are corners and edges, see figure 4(b). They are usually marked with ”No Membership”, because we use normal vectors from an integral image.

Our code runs on 2.2GHz MacPro Book dual core with optimized multi threading code. Overall we achieve 1.7 frames per second with our system. The bottleneck of our approach is the detection of the Manhattan geometry with 305ms for 192 iterations using our MSAC variant. The registration using ICP for all data points of the Kinect needs



(a) Frame No 1 (b) Frame No 60 (c) Frame No 90

(d) Frame No 130 (e) Frame No 181 (f) Frame No 193

Fig. 7. Kinect Ego View of the Sample Tour

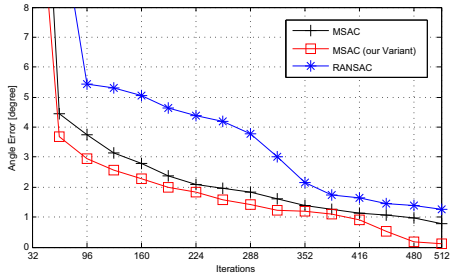


Fig. 8. Average Manhattan System estimation for a limited number of iterations on the entire data set.

250ms. We use an octree map with a resolution of 2.5cm. The update of the octree map is with 5ms relative fast.

V. CONCULSION

In this paper we presented a novel robust method for estimation of the dominant Manhattan system and a geometric constrained ICP for mapping. The proper estimation depends on the amount of visible structure of the Manhattan system within the 2.5D data. As long structure of two axis is visible a robust estimation is possible. As long as Manhattan structure of one axis is clearly visible in the image, its enough that the second axis is partially visible within the data i.e. at least 1 percent of the data. The use of normal vectors is efficient for Manhattan system estimation, but can be computational expensive. The use of integral image style normal vector estimation is a good trade off of runtime and quality and allows to use the entire data set rather than a sub sampled set. The use of a geometric constrained ICP is efficient in Manhattan-like environments regards runtime and map quality. The use of octrees for map representation is a good trade off in runtime and precision. The biggest advantage of this approach is that runtime scales almost constant over time and is memory efficient too.

Experiments have shown that many home and office environments contains Manhattan like structure with all three axis. We want to emphasize that the minimal Manhattan system for our approach are two axis, which is very likely in the most indoor environments. That is usually the ground and a random wall which will be aligned orthogonal in most indoor cases.

Our next steps is to relax the Manhattan constrains and to estimate multiple Manhattan configurations within one view and to use visual odometry.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Community's Sixth Framework Programme (FP7/2009-2012) under grant agreement no FP7-ICT-2009-4-248623 (TACO)

REFERENCES

[1] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge MA, first edition, 2005.

[2] R. B. Rusu, A. Sundaresan, B. Morisset, K. Hauser, M. Agrawal, J. Latombe, and M. Beetz. Leaving flatland: Efficient real-time three-dimensional perception and motion planning. *Journal of Field Robotics, Special Issue: Three-Dimensional Mapping*, 26(10):841–862, 2009.

[3] B. Micusik and J. Kosecka. Piecewise planar city modeling from street view panoramic sequences. In *IEEE CVPR*, 2009.

[4] D. Gallup, J. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys. Real-time plane-sweeping stereo with multiple sweeping directions. In *IEEE CVPR*, 2007.

[5] S. Sinha, D. Steedly, and R. Szeliski. Piecewise planar stereo for image-based rendering. In *IEEE ICCV*, 2009.

[6] A. Gupta, A. A. Efros, and M. Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *European Conference on Computer Vision*, 2010.

[7] Andreas NÃijchter, Hartmut Surmann, Kai Lingemann, Joachim Hertzberg, and Sebastian Thrun. 6d slam with application in autonomous mine mapping. In *Proceedings IEEE 2004 International Conference Robotics and Automation (ICRA 2004)*, New Orleans, USA, April 2004.

[8] Dorit Borrmann, Jan Elseberg, Kai Lingemann, and Andreas NÃijchter. The 3d hough transform for plane detection in point clouds - a review and a new accumulator design. *Journal 3D Research*, 2(2), March 2011.

[9] R. Triebel, W. Burgard, and F. Dellaert. Using hierarchical em to extract planes from 3d range scans. In *IEEE ICRA*, 2005.

[10] Orazio Gallo, Roberto Manduchi, and Abbas Rafii. CC-RANSAC: fitting planes in the presence of multiple surfaces in range data. *Pattern Recognition Letters*, 2011.

[11] Kaustubh Pathak, Andreas Birk, Narunas Vaskevicius, and Jann Poppinga. Fast registration based on noisy planes with unknown correspondences for 3d mapping. *IEEE Transactions on Robotics*, 26(3):424–441, 2010.

[12] R. Rusu, Z. Csaba Marton, N. Blodow, A. Holzbach, and M. Beetz. Model-based and Learned Semantic Object Labeling in 3D Point Cloud Maps of Kitchen Environments. In *IEEE/RSJ IROS*, 2009.

[13] Dirk Holz, Stefan Holzer, Radu Bogdan Rusu, and Sven Behnke. Real-time plane segmentation using rgb-d cameras. In *RoboCup Symposium*, 2011 2011.

[14] A. Murarka and B. Kuipers. A stereo vision based mapping algorithm for detecting inclines, drop-offs, and obstacles for safe local navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-09).*, 2009.

[15] Y. Furukawa, B. Curless, S. Seitz, and R. Szeliski. Manhattan-world stereo. In *IEEE CVPR*, 2009.

[16] A. Flint, C. Mei, D. W., Murray, and I. D. Reid. A dynamic programming approach to reconstructing building interiors. In *Proc European Conference on Computer Vision (ECCV 2010)*, Crete, Greece, 2010.

[17] Hernan Badino, Daniel Huber, Y. Park, and Takeo Kanade. Fast and accurate computation of surface normals from range images. In *International Conference on Robotics and Automation (ICRA)*, May 2011.

[18] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9–13 2011.

[19] K. Varadarajan and M. Vincze. Real-time depth diffusion for 3d surface reconstruction. In *IEEE Int. Conference on Image Processing (ICIP)*, 2010.

[20] R. I. Hartley A. and Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[21] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In *Proc. of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, Anchorage, AK, USA, May 2010.

[22] Dirk Holz and Sven Behnke. Sancta Simplicitas – On the efficiency and achievable results of SLAM using ICP-Based Incremental Registration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1380–1387, Anchorage, Alaska, USA, May 2010.

Emergent Task-Specific Object Semantics through Distributed Experience Networks

Arren J. Glover, and Gordon F. Wyeth, *Member, IEEE*

***Abstract*— Autonomous development of sensorimotor coordination enables a robot to adapt and change its action choices to interact with the world throughout its lifetime. The Experience Network is a structure that rapidly learns coordination between visual and haptic inputs and motor action. This paper presents methods which handle the high dimensionality of the network state-space which occurs due to the simultaneous detection of multiple sensory features. The methods provide no significant increase in the complexity of the underlying representations and also allow emergent, task-specific, semantic information to inform action selection. Experimental results show rapid learning in a real robot, beginning with no sensorimotor mappings, to a mobile robot capable of wall avoidance and target acquisition.**

I. Introduction

THE semantics of an object stem from its purpose or use. While high-level object semantics can come from multiple domains, initial understanding of object use arises from the action-outcome relationships which occur through interaction. Given a task, it is these semantics that enable a robot to select an object from those available in the environment with which to interact.

A robot's sensorimotor coordination (SMC) links perception, action and outcome [1]. The basic object semantics are therefore grounded in the SMC system of the robot. If a robot is to autonomously acquire understanding of object semantics over its lifetime it must be able to autonomously develop its SMC.

Object semantics become important when acting in an environment in which multiple objects are present. However, in such environments, the high dimensionality which occurs due to combinations of different objects being simultaneously detected in the sensory field makes the task of developing the SMC non-trivial, even for a low degree of freedom robot with the task of navigating to useful objects.

The proposed Experience Network (EN) is a type of Markov Network which continually develops the robot's SMC over its lifetime. The EN captures sensory experiences in the nodes of the network, and temporal and motor information in the inter-nodal links. While network dynamics are similar to that of typical reinforcement learning [2], the research focus is on how the continual stream of sensorimotor data is efficiently organised to produce the SMC representations which capture the required object semantics.

Previous work demonstrates the development of SMC representations which allow various goal states to be achieved, when interacting with only a single object [3]. The work is extended into a domain in which the robot simultaneously detects multiple features from both foreground and background entities, requiring the formation of basic semantics in order to achieve a goal state. Three problems are considered: (1) how unchecked growth can be minimised when state-space size increases exponentially with state dimensionality, (2) how emergent semantic information about feature relevance can be captured and used to better inform action, and (3) how learning speed can be boosted through inferring actions from of novel states.

This paper presents an alternative to developing a network which stores the entire sensory state within a single node. Instead, each node is created with only a single sensory feature, and thereby distributes the state across multiple nodes in the network. This has three benefits: (i) the state space size becomes O(N) with respect to the dimensions of the sensory features, rather than O(c^N), (ii) the probabilistic dynamics of the Markov network can perform pattern generalisation and separation to efficiently generate more semantically driven sensorimotor coordination, (iii) inference of action from nodes can be more easily calculated as there is no ambiguity in the credit assignment resulting from groupings of features.

The paper proceeds by outlining related work in section II, before describing the details of the EN in section III. Studies using a real mobile robot are presented in section IV. Object detection is colour-based; however features are extracted from both the foreground and background entities. Results and discussions are presented in Section V and VI respectively, and Section VII describes future work.

II. Background

Autonomously building object models is usually performed by predefining the foreground [4-6], or the background [7], so the semantically interesting information is available to the robot. The problem of autonomously learning the relevance of a feature (which is task contextual, and may not be static) is less often considered.

Learning appropriate actions to achieve the desired goal

Manuscript received February 16, 2011. This work was supported in part by the Australian Research Council under a Discovery Project Grant DP0987078.

A. J. Glover and G. F. Wyeth are with the Queensland University of Technology, QLD Australia. Phone: +61731389974 Email: {aj.glover, gordon.wyeth}@qut.edu.au.

is often performed by finding object, action, outcome relationships in a second phase, after object representations have been formed [5]. Outcome learning has been performed by statistically averaging the change in object state, after an applied action, over multiple runs [7],[8] More continuous methods generally apply an uphill learning algorithm to optimise the performance of a single task [4]. To acquire relevant semantics over the lifetime of the robot the learning must occur simultaneously to actions being performed. The representations learnt must also be flexible in allowing different goals to be achieved given a change in task.

While Markov networks have successfully been employed in robot navigation scenarios in the past [9], research in reinforcement learning has shown that network structures have issues with computational complexities when confronted with large state spaces and high dimensionality [2]. However, other recent trends in object/affordance learning have shown that the dimensionality can be reduced by employing Bayesian network learning to capture the conditional dependence relationships [5], allowing the full state distribution to be estimated by modelling only the most causal relationships.

III. The Experience Network

The Experience Network is a Markov network of sensory states which have been experienced by the agent and which are linked together through the actuation commands that were performed when the state changed. The sensory state at time t is referred to as an agent's experience e_t:

$$e_t = < V_t, H_t >$$

where V_t is a set of visual features and H_t is a set of haptic features. Each visual feature v_i in features V_t is extracted from the raw data and defined as:

$$v_i = < f_i, c_i >$$

where $f_i \in F_V$ is a description or label component of the feature and c_i is a component describing the position in the visual field. Similarly haptic features are defined by a label, selected from set F_h, and position in the haptic field.

The robot's actuation command is referred to as an action. The action a at time t is:

$$a_t \in$$

where the set A defines the set of all possible actions.

A. A Network of Experience and Action

The EN is realised using a graph structure in which the sensory experiences are stored within the nodes, N, and the actions are stored within the set of all links, L, that connect nodes (Fig. 1). The graph structure can then be exploited by forming an agent state from one or more nodes and following links towards a node with a desirable experience.

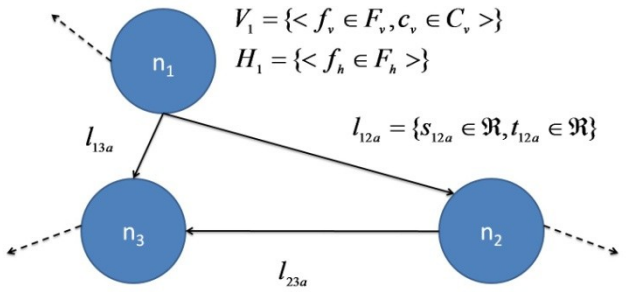


Fig. 1. The multi-dimensional EN projected in 2 dimensions. Each node stores sensory data while each link store transitional data. The dotted lines indicate links to experiences not shown.

Each node $n_i \in$ ´ is representative of an experience:

$$n_i = < V_i, H_i >$$

where V_i are the features which describe the node visually and H_i are the features which describe the node haptically. A link connecting node i to node j by performing action a is defined as:

$$l_{ija} = < s_{ija}, t_{ija} >$$

where s_{ija} is the strength or repeatability of the link, and t_{ija} is the time to traverse between nodes.

B. Measures of Node Similarity

Nodes are added to the network by considering the information currently stored in the network and comparing it to the experience at time t. To perform a comparison a measure of the similarity of two nodes is required. The probability that node n_i is the same as node n_j is calculated as:

$$P(n_i \mid n_j) = \text{?}(V_i \mid V_j) P(H_i \mid H_j) \quad (1)$$

where the probability based on the visual element is calculated as:

$$P(V_i \mid V_j) = \prod_{y=} \; {}_{\in_j} \text{x}[P(f_y \mid f_z) P(c_y \mid c_z)] \quad (2)$$

which makes the assumption that the feature label and position are independent. Independence is also assumed between feature labels; the label similarity is calculated as:

$$P(f_y \mid f_z) = \begin{cases} 0, f_y \neq f_z \\ 1, f_y = f_z \end{cases} \quad (3)$$

The probability due to position in sensory field is defined by a closeness measure which assumes that areas in the field are dependent on neighbouring areas:

$$P(c_y \mid c_z) = \frac{1}{1 + \; c_y - \; _z \mid} \quad (4)$$

The haptic component can be calculated in a similar fashion depending on the sensor arrangement. Due to simplistic haptic sensors of the current robotic setup, the haptic probability was simplified to:

$$P(H_i \mid H_j) = \begin{cases} 0, f_i \neq f_j \\ 1, f_i = f_j \end{cases} \quad (5)$$

C. Node Addition

The two methods for developing nodes and adding them to the EN, combination of features and distributed features, are now defined. In general both methods form the current robot state S_t, which is a subset of all nodes N, by selecting nodes already in the network or by adding new ones to it. New nodes are added when no current node similarity is above the threshold T_n.

Combining features in a single node is the direct extension to previous work, a single node is required to represent the current state and all features within the node must match to 'revisit' this node in the future.

Algorithm 1: Combination of Features

1. $S_t \leftarrow$; $n_p \leftarrow$; $n_{\max} \leftarrow \max_{n_i \in} P(n_i \mid n_p)$
2. **if** $P(n_{\max} \mid n_p) >$
3. $S_t \leftarrow$
4. **else**
5. $N \leftarrow$; $S_t \leftarrow$

The distributed method adds nodes to the network by forming individual nodes for each feature in the experience. Many nodes are then required to represent the current state; however a change in a single component feature does not shift the entire state to a new node, only the specific feature that changed.

Algorithm 2: Distributed Features

1. $S_t \leftarrow$
2. **for** $v_m \in$
3. $n_p \leftarrow < v_m, H_t >$; $n_{\max} \leftarrow \max_{n_i \in} P(n_i \mid n_p)$
4. **If** $P(n_{\max} \mid n_p) >$
5. $S_t \leftarrow$
6. **Else**
7. $N \leftarrow$; $S_t \leftarrow$

D. Link Addition

Links are added between nodes by considering the change in robot state from time t-1 to time t. For each node in S_{t-1} no longer present in S_t, a link is added to the most probable node in S_t.

Due to the closeness measure, $P(c_y|c_z)$, and the fact that the label probability, $P(f_y|f_z)$, is a binary measure, often the most probable node will be a nearby node with the same feature. This behaviour is designed to allow the modelling of feature motion through the visual field, however this is not an absolute 'rule' as noise in feature detection, as well as perceptual aliasing (the same feature can be in scene multiple times), will introduce uncertainty. Over time correct trends in motion should emerge. In cases in which no node has the same feature all other nodes become equally probable (at 0) and thus links are made to all nodes. Over time emergent behaviour, such as a feature changing label due to viewpoint change, can be captured.

Once links are chosen the current network links are updated by increasing the link strength, s_{ija}, by 1 and averaging the time component, t_{ija}, over all traversals.

E. Network Navigation

The graph representations formed in the EN are exploited to direct future agent action, closing the sensorimotor loop. Given the current agent state, S_t, the next action is selected so as to minimise the probable time it takes to arrive at a node that complies with a set goal criteria.

The probable time to a goal node given an action, $T(a_k)$, is calculated for every node in the network, given each action, using dynamic programming techniques [2]. Nodes which meet goal criteria are set to have a T value of 0 and every other node is updated according to:

$$T_i(a_k) = \sum_{n_j \in N} P_{ij}(a_k)(t_{ija} + \min_{a_r \in A}[T_j(a_r)]) \quad (6)$$

where $P_{ij}(a_k)$ is the probability of action a_k changing the state from node n_i to node n_j. $P_{ij}(a_k)$ is calculated as:

$$P_{ij}(a_k) = P(n_j \mid n_i, a_k, L) = \frac{s_{ija}}{\sum_{n_k \in N} s_{ika}} \quad (7)$$

If action a_k has never been performed from node n_i (i.e. the denominator of (7) is 0) the time to goal is set as:

$$T_i(a_k) = \quad (8)$$

which instigates the exploration of unperformed links to new nodes. Alternatively the time can be inferred from other nodes as described in the following sub-section.

In the distributed feature method the current state is formed by multiple nodes; a greedy method is used to perform action selection in order to minimise the probable time to goal given all possible actions and all nodes in S_t:

$$a_t = \min_{n_i \in} (\min_{a_k \in}[T_i(a_k)]) \quad (9)$$

F. Inference

Many objects in the world have similar SMC behaviours, and hence will have similar network connections within the EN. To remove the need for complete exploration of a previously un-encountered feature's state-space, the time to goal, $T_i(a_k)$, can be inferred from already established nodes. If a successful match, based on partial link similarity, can be made between two different features, the entire state-space of that feature can be inferred resulting in reduced learning times. Feature similarities are more causally calculated in a distributed network as the links from a given features only represent the change in the change in a single feature, as opposed to many.

Two algorithms are introduced to perform inference. To recognise when two features exhibit similar behaviour, the inference measure I between the features f_u and f_v is continuously calculated as the network is developed:

$$I_{uv} = \frac{\beta}{\alpha} \quad (10)$$

where the values of α and β are updated each time a link is added. Given a link between node n_i and n_j is added with

the action a_{t-1} the values of α and β (initially zero) are updated by finding similar links within the network based on the feature position c in pre and post nodes (Algorithm 3). The resulting cross-correlation matrix I defines the similarity in sensorimotor behaviour for all features in F.

Algorithm 3: Inference Calculation

1. **for** $n_g \in$
2. **if** $P(c_g \mid c_i) >$
3. $\alpha \leftarrow \alpha +$
4. **for** $n_h \in$
5. **if** $P_{gh}(a_{t-}) > .5$ **&** $f_h =$ f_g **&** $P(c_h \mid c_j) >$
6. $\beta \leftarrow \beta +$

The similarity between features is used when calculating probable times of each node for network navigation. If an appropriate inferred node can be found when action a has never been performed from node n_i the time is, instead, inferred.

Algorithm 4: Inference Usage

1. **for** $n_i \in$ $, a \in$
2. **if** $\sum_{n_i \in N} s_{ika} =$
3. **for** $n_j \in$
4. **if** $I_{ij} >$ **&** $P(c_j \mid c_i) >$
5. $T_i(a) \leftarrow$ (a)

IV. Experiments

A. Robot and Environment

Experiments were performed on a Pioneer3-DX mobile robot with a 2-DOF gripper including an IR break beam between the paddles for haptic sensing. A forward facing Logitech Webcam Pro 9000 was used as visual input (Figure 2).

Fig. 2. The robot in the experimental 3x3m walled environment. The red, and other (green) markers, can be detected using the gripper while the wall and floor cannot.

The robot's experience consisted of visual features from the camera and haptic data from the gripper. Colour-based image segmentation was employed as illustrated in Figure 3. Although colour segmentation is a simple way to extract useable features from the environment the experiments are only initial studies in which no distinction between foreground and background is made, the robot had to develop its own semantics about which features where important to achieve its goals.

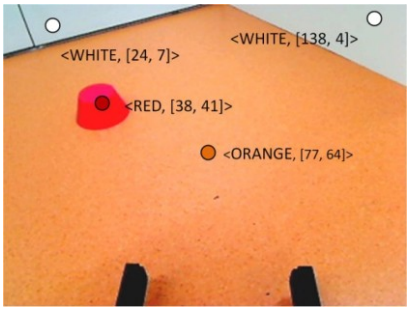


Fig. 3. The visual features extracted from a typical scene. The image is colour segmented allowing features to be generated by the walls, the floor, and the red and green markers. Each visual experience consists of the visual features f_v and the centroid c_v.

The elementary actions available to the robot were forward, left, and right and selected at a rate of 10 Hz. Laser range data was used to detect when robot actions would lead to a collision, any detection would stall the robot before collision occurred. The network was built with the node similarity threshold, T_n, set so nodes cover a region with a ~20 pixel radius (in a qqVGA image) and the inference threshold, T_I, was set to 30%.

B. Experimental Procedure

Each run was conducted with the robot in an initial position and the markers randomly placed within the robot's field of view. The robot began with an empty EN, hence no understanding of the sensory-motor mappings. The goal criterion was set to be nodes with the IR break beam triggered. Robot and marker positions were reset upon reaching a goal state. Two EN's were developed based on both Algorithm 1 and Algorithm 2, however only the network formed using Algorithm 2 was used to control the action of the robot.

Study 1: 20 runs were performed with a single red marker in the arena. The study aim was to demonstrate that the distributed EN could be successfully exploited to solve the target acquisition problem in the face of multiple different background/foreground sensory input.

Study 2: A further 20 runs were performed, continuing the use of the EN from Study 1. An additional red marker was added and the aim was to investigate performance and representation size when two distinctive (goal) features were present.

Study 3: A final 10 runs were performed with only a single green marker in the arena. The study aim was to investigate the utility of inferring novel feature behaviour from known feature representations.

V. Results

A. Node Addition Method Comparison

The distributed network (DN) led to a smaller network size than the combined features network (CFN) as can be seen in Figure 4. The CFN size grew significantly larger than the DN due to the state-space dimensionality; each feature in

each position was an orthogonal axis in the state space. It was from this evidence that the CFN was disregarded in further analysis; the time and physical exploration required to develop worthwhile representations was exponentially larger than the DN.

The DN growth supports the method's advantages: it slowed in the first study as a reasonable amount of the state space was physically explored, was not required to grow when a second marker was added, and only required new nodes to represent the green marker in the third study. The CFN continually increases in size over all three studies as the state-space is exponentially larger. It is only in the third study when growth slowed, due to the traversal of similar paths by the (action selecting) DN.

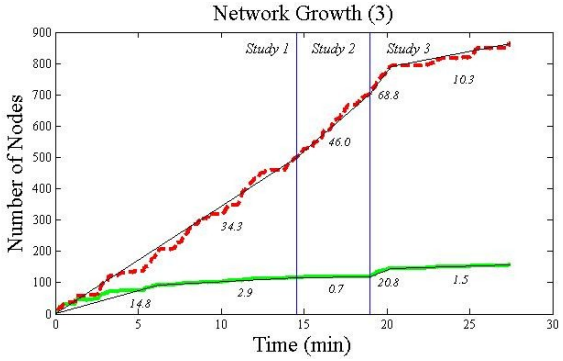


Fig. 4. The state-space size comparison between the CF network (dotted) and IF network (solid). The slope is reported in units of nodes/minute.

B. *Study 1: Grounding Multiple Sensory Features*

The initial trial took over 5 minutes as the robot filled out the empty EN with newly acquired sensorimotor experience. Subsequent runs, even with the marker in a random position, showed the effectiveness of the DN to acquire the task specific semantic information required to complete the task, as the average time dropped to 28s. It learnt that the marker feature is the most important feature to use given the haptic goal and in most cases the wall and floor features are not helpful.

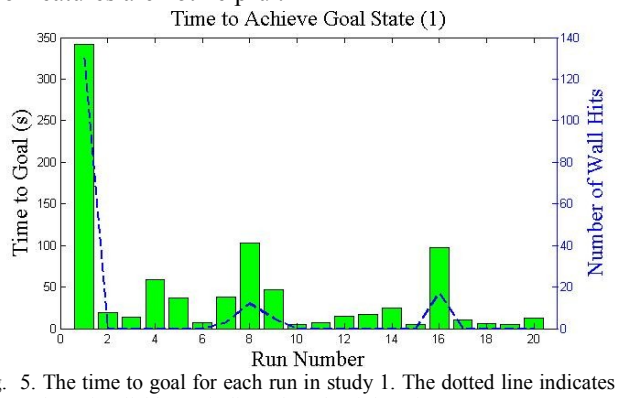


Fig. 5. The time to goal for each run in study 1. The dotted line indicates the number of wall 'hits' as indicated on the right axis.

When the robot cannot see the informative marker feature, goal directed behaviour is not possible. There is no egocentric information available to the robot to make a prediction about the relative location of the target unless it is in the field of view. However the robot would learn to turn rather than drive straight at the wall, although at times it would prevaricate between turning left and right without gaining sight of the target, as evidenced in runs 8 and 16. These runs also show an increased number of wall hits as the robot learnt more about the semantics of the wall as a feature. The ability to avoid walls is learnt as can be seen in the decreasing number of wall hits.

C. *Study 2: Navigating with Multiple Goal States*

The already grounded network was able to continue to achieve a goal state when a second marker was added to the environment (Figure 6). Very few new nodes were added to the EN in this study (Figure 4), hence the SMC state representations were already adequate for correct behaviour with links continually being reinforced. Due to the greedy algorithm the closest node could be selected as the goal target only further reducing the time to goal. In contrast the CFN network would have to develop new representations to represent the combination of 2 marker features.

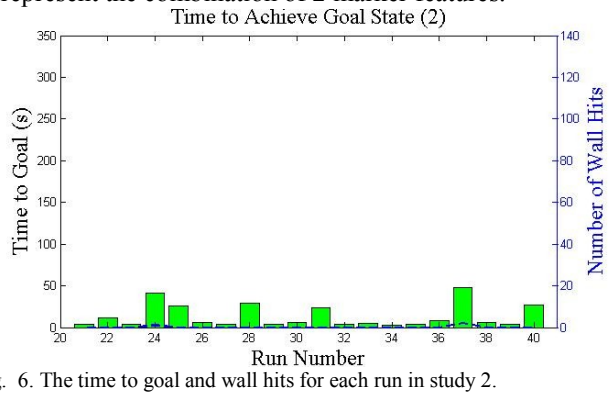


Fig. 6. The time to goal and wall hits for each run in study 2.

D. *Study 3: Inferring Novel Feature Utility*

While inference between features was being calculated and, if necessary, invoked throughout all studies it was only in the third study in which they came into effect. The walls floor and marker all had unique movement behaviour with respect to their centroids and hence no inference was performed. When the green marker was introduced (producing a previously unseen feature) it was closely matched with the already developed red marker nodes (Table 1), allowing correct actions to almost immediately be performed to achieve the goal state (Figure 7).

The long run duration in runs 45 and 48 were caused by the incorrect association of the green marker with the wall features. In both runs the marker was positioned towards the top of the visual image, where typical wall behaviour is to turn, rather than driving forwards. Action was then incorrectly inferred and hence direct motion to the marker was not performed. The red marker had a zero inference value from the green marker as the red marker was never reintroduced after the green marker was introduced.

TABLE 1
MOTION INFERENCE BETWEEN FEATURES

	Wall	Floor	Red Marker	Green Marker
Wall	48.45%	29.41%	17.31%	58.73%
Floor	4.35%	19.25%	10.20%	15.40%
Red Marker	14.81%	5.70%	47.26%	0.00%
Green Marker	32.79%	4.50%	33.21%	58.89%

The probability that a feature (rows) has the same sensorimotor behaviour as another (columns). Auto-inference is not 100% as the probability only increased when $P_{ij}(a) > 0.5$.

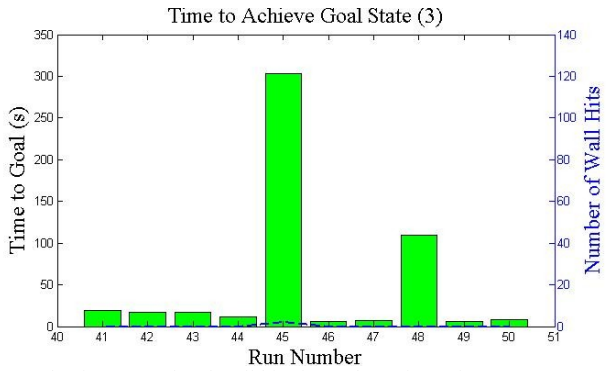


Fig. 7. The time to goal and wall hits for each run in study 3.

VI. DISCUSSION

The Distributed Experience Network algorithm differs from other methods of learning sensorimotor coordination in a number of important ways.

A. Learning is One-Shot and On-Line

While the EN develops an adaptable SMC from scratch, the network complexity is kept manageable by having each node deal only with a single sensory feature from an experience, representing the experience in a distributed fashion. The EN does not require separate learning and recall phases. All of the robot's interactions with the environment result in learning, allowing the robot to continually update its SMC over the lifetime of the robot

B. Attention is Intrinsic to the Network

Our experiments used colour segmentation to simplify the incoming visual information, but no specification was made as to which features were in the foreground, and which were in the background. There is no inherent attention operator to highlight features of interest – rather the EN develops task specific semantic information by noting which sensory changes occur consistently with motor action. Only the foreground features (i.e. from the object) are recognised as informative.

C. Learning can be Boot Strapped by Inference

Bootstrapping knowledge between features is important when using state-action representations especially in high dimensional space, as otherwise each state of each feature needs to be explored and grounded. The distributed network more easily allows for dependencies to be learnt which are then exploited to reduce the amount of learning the network has to do before appropriate actions are emergent.

VII. FUTURE WORK

Adapting these studies to a real-world environment requires the introduction of appearance-based features, such as SURF [10]. The large increase in the number of features which must be processed forms the key challenge of doing so. While the individual features can be picked out in the DN, a single object can produce multiple features in this scenario. It would then seem viable to allow small groupings of features to autonomously form in nodes thereby reducing the number of features to be processed, while also demonstrating emergent 'object recognition'.

Bayesian network theory could also be employed to increase the robustness of inference between variables in the system, such as in [5]. Not only does this speed learning but also allows reduction of the dimensionality of the system if new variables were added, such as global X, Y coordinates. This would allow the robot to then search for objects outside the immediate view, and also, in a similar method to visual features, develop task-specific spatial semantic information.

ACKNOWLEDGMENT

The authors would like to thank the other members of the Lingodroid group Dr Ruth Schulz and Prof. Janet Wiles.

REFERENCES

[1] R Pfeifer and C Scheier, "Sensory-motor coordination: The metaphor and beyond", Robotics and autonomous systems, 20(2-4):157-178, 1997

[2] RS Sutton and AG Barto, Reinforcement learning: An introduction. The MIT press, 1998

[3] A. Glover, R. Schulz, G. Wyeth and J. Wiles, "Grounding Action in Visuo-Haptic Space using Experience Networks", In The Australasian Conference on Robotics and Automation, Brisbane, Australia, 2010

[4] Christian Scheier and Dimitrios Lambrinos, "Categorization in a real-world agent using haptic exploration and active perception", Simulation of Adative Behaviour, pages 65-75, 1996

[5] L. Montesano, M. Lopes, A. Bernardino and J. Santos-Victor, "Learning Object Affordances: From Sensory--Motor Coordination to Imitation", IEEE Transactions on Robotics, 24(1):15-26, 2008

[6] J. Sun, J.L. Moore, A. Bobick and J.M. Rehg, "Learning visual object categories for robot affordance prediction", The International Journal of Robotics Research, 29(2-3):174, 2010

[7] J Modayil and B Kuipers, "The initial development of object knowledge by a learning robot", Robotics and autonomous systems, 56(11):879-890, 2008

[8] P Fitzpatrick and G Metta, "Grounding vision through experimental manipulation", Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, 361(1811):2165, 2003

[9] S. Koenig and R. Simmons, "Xavier: A robot navigation architecture based on partially observable markov decision process models", Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems, 91–122, 1998

[10] H. Bay, T. Tuytelaars and L. Van Gool, "Surf: Speeded up robust features", Computer Vision–ECCV 2006, 404-417, 2006

Fast and Robust Object Detection in Household Environments Using Vocabulary Trees with SIFT Descriptors

Dejan Pangercic, Vladimir Haltakov, Michael Beetz
{pangercic, haltakov, beetz}@cs.tum.edu
Technische Universität München, 85748 Munich, Germany

***Abstract*— In this paper we describe the *ODUfinder*, a novel perception system for autonomous service robots acting in human living environments. The perception system enables robots to detect and recognize large sets of textured objects of daily use. Efficiency, robustness, and a high detection rate are achieved through the combination of modern text retrieval methods that are successfully used for indexing huge sets of web pages and state-of-the-art robot vision methods for object recognition. The result is a robot object detection and recognition system that, with an accuracy rate of more than 80%, can recognize thousands of objects by learning and using vocabulary trees of SIFT descriptors.**

I. Introduction and Approach

A robot acting as an household assistant must be capable of recognizing the hundreds of objects of daily use that are present in its operating environment. It also has to be able to recognize new objects, for example, when emptying a shopping basket to put the purchased items where they belong. One way to equip robots with knowledge about the physical look of these various objects is to retrieve images of them from grocery webstores, such as `www.germandeli.com` (Germadeli), or image libraries, such as google images.

In this paper we report on the design and implementation of the Objects of Daily Use Finder (*ODUfinder*) perception system that can deal with some aspects of this challenge. The system can detect and recognize textured objects in typical kitchen scenes. The models for perceiving the objects to be detected and recognized can be acquired autonomously using the robot's camera as well as by loading large object catalogs such as the one by Germandeli into the system. In the system configuration described in this paper, the robot is equipped with an object model library containing approximately 3500 objects from Germandeli and more than 40 objects from the *Semantic3D* database[1]. The *ODUfinder* achieves an object detection rate of 10 FPS and recognizes objects reliably with an accuracy rate of over 80%. Object detection and recognition is fast enough so that it does not cause delays in the execution of the robot's tasks.

The *ODUfinder* system employs a state-of-the-art object perception technique Scale Invariant Feature (SIFT) [1] using a vocabulary tree [2], which we extend in two important ways. First, the comparison of object descriptions is done probabilistically instead of relying on the more error-prone original implementation with the accumulation of query sums. Second, the *ODUfinder* detects candidates for textured object parts by over-segmenting image regions and then combines the evidence of the detected candidate parts to infer the presence of the object. These extensions substantially increase the detection rate as well as the detection reliability, in particular in the case of partial obstruction and in certain lighting conditions like specular reflections on object parts. Another contribution is the mechanism realized to enable automatic acquisition of incomplete visual appearance templates, such as the ones from Germandeli. In a nutshell this paper provides the following main contributions:

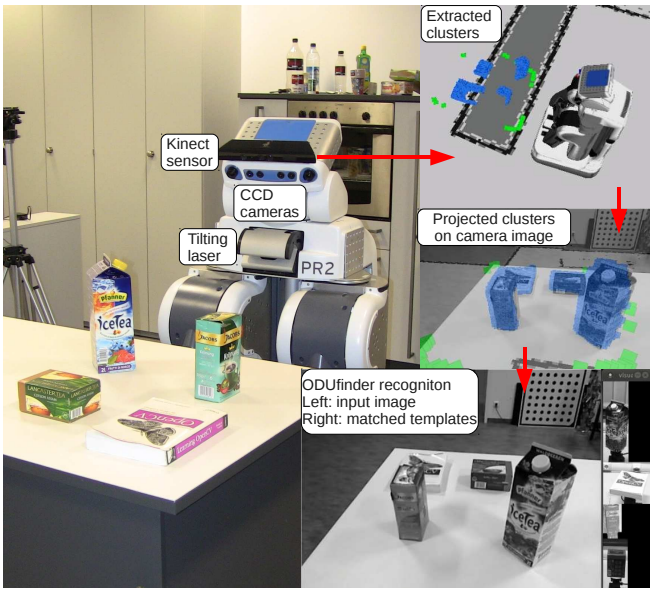


Fig. 1. TUM-PR2 robot recognizing objects lying on the tabletop using kinect sensor. Right column depicts extraction of clusters from point clouds (top), projection of clusters onto camera image and Region-Of-Interest extraction (middle) and, finally, *ODUfinder* recognizing objects (bottom).

- An application of a vocabulary tree matcher to real perception problems;
- A probabilistic comparison of objects' descriptors;
- An over-segmentation-based recognition of textured objects;
- A mechanism for automatic acquisition of incomplete visual appearance templates.

ODUfinder system is out-of-the-box and open-source

[1]http://ias.cs.tum.edu/download/semantic-3d

available as a ROS package [2] and can be easily deployed in any kind of robot equipped with a 3D sensor and a camera that are calibrated with respect to each other.

The remainder of this paper will proceed as follows: in the next section we discuss similar approaches, which is followed by a brief description of the system's architecture. SIFT based object recognition is explained in Section V, followed by Section VI focusing on the *ODUfinder's* capability to learn new objects. In Section VII we present the results of experiments and, finally, in the end we conclude and give suggestions for future research.

II. Related Work

Nakayama et al. [3] present the AI Goggles system, which is a wearable system capable of describing generic objects in the environment and of retrieving the memories of these objects by using visual information in real time without any external computation resources. The system is also capable of learning new objects or scenes taught by users. As the core of the system, a high-accuracy and high-speed image annotation and retrieval method supporting online learning are considered. The authors use color higher-order local auto-correlation (Color-HLAC) features and the Canonical Correlation Analysis (CCA) algorithm in order to learn the latent variables.

Arbeiter et al. [4] implemented a framework for 3D perception and modeling. The proposed algorithm can be used to reconstruct a 3D environment or learn models for object recognition on a mobile robot. Both color and time-of-flight cameras are used, and 2D features are extracted from color images and linked to 3D coordinates. Those coordinates then serve as input for a modified fastSLAM algorithm that is capable of rendering environment maps or object models.

A self-referenced 3D modeler is presented in [5] by Strobl et al., where the authors demonstrate that an ego-motion algorithm can simultaneously track natural, distinctive features and provide 3-D modeling of the scene. The use of stereo vision, an inertial measurement unit and robust cost functions for pose estimation further increased system's performance.

Incremental learning and recognition of objects is done in an unsupervised manner in [6], but Triebel et al. focus mainly on chairs, and it is not clear how well multiple objects could be reliably detected without any prior information. Moreover, scalability is hard to assess since only one view is analyzed at a time.

III. System Overview

The *ODUfinder's* mode of operation is depicted in Figure 2. The robot simultaneously takes a 3D scan and captures an image of the scene in front of it. The robot generates object hypotheses by detecting candidate point clusters in the 3D point cloud acquired by the depth sensor. These object hypotheses are then back-projected into the captured image as regions of interest and searched for detecting and recognizing objects (See section IV-A).

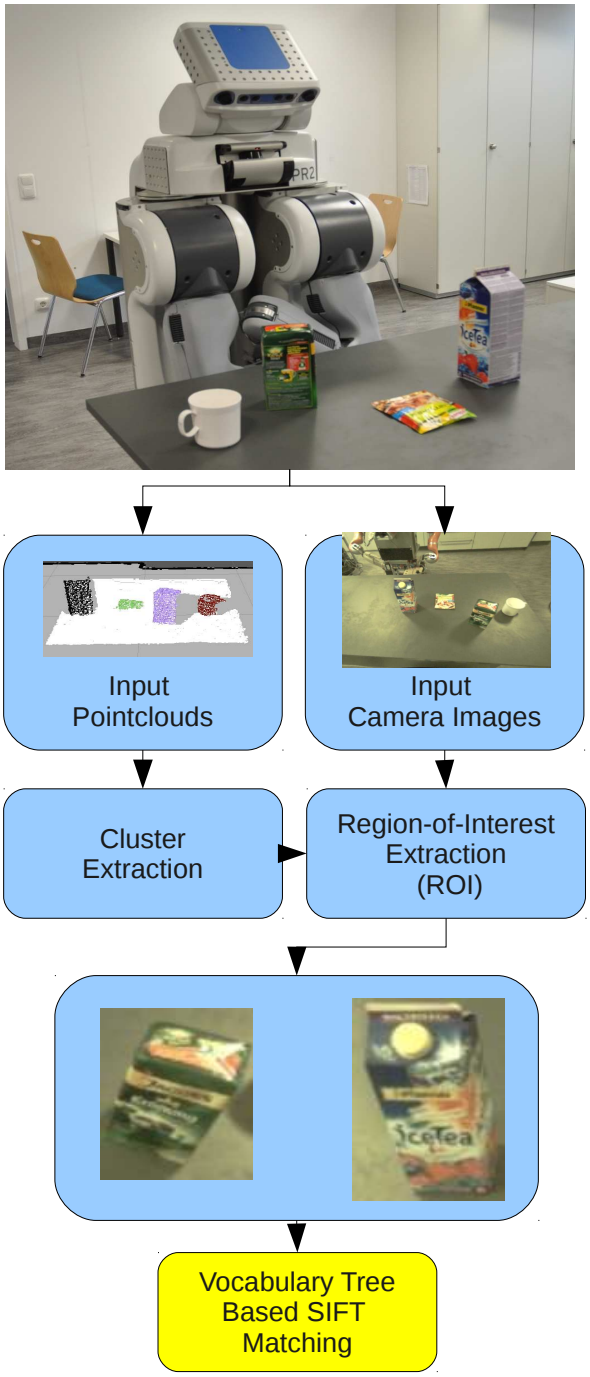


Fig. 2. System overview

For the SIFT-based detection we first determine segments by performing region growing on detected features in image space, which typically results in an over-segmentation of the region of interest (shown in Figure 3, right). Identifying the object and the image region it belongs to is then performed through methods transferred from document retrieval. In document retrieval tasks, for example in a web searches we

[2] http://www.ros.org/wiki/objects_of_daily_use_finder

look for the documents that best match a given query term. To do so the search engines compute frequency statistics for discriminative words or better word stems as a pre-processing step performed on all documents. Given a search term, fast indexing mechanisms quickly search for the documents that are, with respect to frequency, particularly relevant for the search term. The application of text retrieval technology for object matching is promising because it is very mature and the techniques allow for rapid functioning with high recall and precision rates.

The computational idea of textual document retrieval can be mapped to object description matching in the following way: the descriptors computed from the regions of interest that belong presumably to (or are partial views of) the same object are considered to be the search term. The object descriptors of the different views of the relevant objects are the documents of the document retrieval model. Word frequencies are replaced by the frequency of visual object features. Given a large set of objects represented by their object descriptors and the feature descriptor of an image region, we can then index the objects where the particular features are particularly prominent using the respective methods of document retrieval.

In this paper we apply vocabulary trees for TF-IDF (Term Frequency Inverse Document Frequency [7]) indexing, a method used in document retrieval to find documents that best fit a given textual user query. In this reformulation of object identification, vocabulary trees speed up the retrieval of the matching objects.

The methods for object descriptor matching do not only match a given region descriptor to the large set of object descriptors, they also learn new object descriptors to be put into the visual object library.

The subsequent sections describe the individual computational steps performed by the *ODUfinder* in greater detail.

IV. Theory of Region of Interest Extraction

In human living environments the objects of daily use are typically standing on horizontal, planar surfaces or, as physics-based image interpretation states it, they must be in stable force-dynamic states. The scenes they are part of can be cluttered or the objects are more or less isolated. To account for these conditions, the *ODUfinder* employs two alternative methods for region extraction: first, the combined 2D-3D extraction for objects standing more or less isolated on planar surfaces, and, second, the region-growing based extraction for cluttered scenes, such as the objects standing in a cupboard. These two methods are described below.

A. Combined 2D-3D Object Candidate Detection

The combined 2D-3D object detection takes a 3D point cloud acquired through a tilting laser scanner or a kinect sensor and a camera image of the same scene. Figure 3 (left) shows how the *ODUfinder* detects major horizontal, planar surfaces within the point cloud and point clusters that are supported by the planes [3]. The identified point clusters in the point cloud are then back-projected into the captured image to form the region of interest that corresponds to the object candidate. An accurate back-projection is possible thanks to the accurate robot calibration, as described by Pradeep et al. [9]. The sensors are calibrated using a non-linear bundle-adjustment-like optimization to estimate various parameters of the TUM-PR2 Robot.

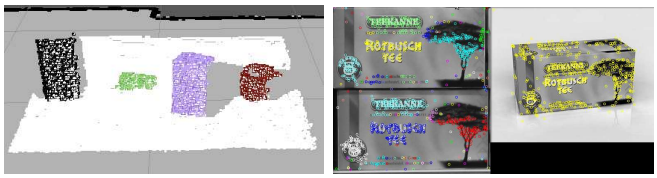


Fig. 3. Left: Region of Interest Extraction using back projection of 3D points, Right: Over-segmentation using a region-growing based approach.

B. Over-Segmentation-Based Object Candidate Detection

The second method for identifying image regions that might correspond to objects is the computation of clusters of visually distinctive pixels in the image space. This method exploits the fact that many objects of daily use have distinct textures.

In our case we determine the visually distinctive pixels using SIFT features and apply region growing algorithms to determine the clusters. Region growing starts from a point that does not belong to any clusters and incrementally adds points that are in a predefined radius *r* around the original point. The process is repeated for all newly added points. This results in clusters that represent the strongest texture "islands" in the image.

For our application, the quality of the segmentation results heavily depends on the appropriate setting of the radius parameter *r*. In order to improve performance, we adaptively chose the radius length in relation to the level of texturedness of the camera image using a scaled and shifted logistic sigmoid function:

$$r^2(x) = (r^2_{max} - r^2_{min})(K(1 - logsig(x - A))) + r^2_{min} \quad (1)$$

where logsig is defined as:

$$logsig(x) = \frac{1}{1 + e^{-x}}, \quad (2)$$

which tends to work well for input images of the same size.

In the equations above the argument x is the number of keypoints in the image. The parameters r_{min} and r_{max} denote the maximum and the minimum values of the radius. The parameter A denotes the value of x, where the value of the function is the average of the minimum and maximum value of the radius. The constant K denotes the speed at which the function approaches its minimum and maximum values. These 4 parameters are determined empirically and

[3]The implementation details of these steps have already been described in [8] and fall outside the scope of this paper.

are valid for images of roughly similar sizes. In the experiments below we use the following values: A = 800, K = 0.02, r_{min} = 200, r_{max} = 600.

This approach allows for bigger distances in images containing fewer features, thereby forming better shaped clusters and, respectively, it allows for small radiuses for images with a lot of features, thereby avoiding the use of extreme radius values.

V. Recognition of Textured Objects - Implementation

The *ODUfinder* performs object recognition of textured objects by computing the set of SIFT descriptors for all distinctive pixels in any given region of interest and then determines the object model in the library that best explains the set of SIFT descriptors of the region of interest. Each object view contains the set of SIFT descriptors of the distinctive pixels.

Unfortunately, comparing a region of interest with every object view in the object model library is prohibitively expensive. To this end, as proposed by Sivic and Zisserman [10], we consider object recognition as a document retrieval problem, which enables us to use fast data structures and retrieval algorithms and apply them to object recognition problems for large libraries of object models.

In this paper we employ vocabulary trees that were developed by Nister and Stewenius [2] for retrieving similar images in very large image libraries. In this section we show how we have specialized this technique for the purpose of object recognition in the context of robot perception. Our principal aim was to improve the capability of the proposed method for identifying objects in real scenes, which required taking different lighting conditions, obstruction and clutter, and the uncertainty and noise associated with physical sensors acting in the real world, into consideration.

A. Vocabulary Tree

The vocabulary tree of branching factor K and depth L is a tree data structure where the nodes in the tree represent a set of SIFT descriptors. The root node of the vocabulary tree represents the SIFT descriptors of all views of all object models in the library. If a node *n* in the vocabulary tree represents the set of SIFT descriptors $\mathcal{N}$ then its children nodes represent the partitioning of $\mathcal{N}$ into *k* subsets represented by the children nodes $cn_1 \ldots cn_k$, where the SIFT descriptors within a children nodes are similar and the ones of different children nodes dissimilar.

Thus, by taking a SIFT descriptor *sd* and classifying it hierarchically through the vocabulary tree using the defined distance measure on the SIFT descriptors we quickly find the set of SIFT descriptors that are most similar in the object model database as the leaf nodes, whose representative SIFT descriptors have the smallest distances to *sd*. For efficiency, *sd* is not compared to all features in a given node, but to the centroid of its features.

The SIFT descriptors in the vocabulary tree also have a reference to the object model in which they occur. Thus, when *sd* matches a leaf node it votes for the object models that the SIFT descriptors of the identified leaf belong to.

The children nodes $cn_1 \ldots cn_k$ of $\mathcal{N}$ are computed by applying k-means clustering to the SIFT descriptors of node *n*. Since the TF-IDF algorithm works on words (the equivalent of leaf nodes), we use a vocabulary tree to convert the keypoint descriptors into words, where each word is an integer value corresponding to the number of the leaf node.

B. Building the database

In our approach we use a similar database (object model library) as described in [2]. In order to be able to detect objects the database only stores the quantized SIFT features of the images, but not the images themselves.

1) Extracting SIFT features: In order to extract the visual SIFT features from the images we use an open-source implementation [11] of the standard SIFT algorithm as initially described by [1]. Each SIFT feature is characterized by a 128 dimensional descriptor vector, 2 image coordinates, a scale and an orientation value. In the current implementation we only use the descriptor vectors for the detection process and the image coordinates for visualization.

2) Generating database documents: After we have the vocabulary tree, we quantize feature descriptors to single words. For every image, we take all SIFT features, we quantize them with the vocabulary tree and we group the resulting words into one document for every image. In this way each document is composed of a list of all quantized features corresponding to a single image.

3) Populating and training the database: After generating all image documents, we insert them into a specialized database as proposed in [2]. The database is then trained with the TF-IDF [7] algorithm. After this training the database can be queried with documents generated from input camera images in order to find the best database matches between objects in the image and objects in the database. The database documents, along with specific database information, can be stored in a binary format in order to allow for fast loading of the database. Additional information, like image file names, textures and feature coordinates, is also saved for visualization purposes.

The whole detection process is implemented as a single ROS node, which receives an image coming from the camera and outputs the most probable N matches from the database.

C. Retrieving Object Models

In order to find an object in the received image we have to generate a database document in the same way as described above. We first extract the SIFT features from the received image and we quantize the descriptor vectors to words with the vocabulary tree. A single document is formed from all words of the input image and we can query the database with it. The database returns the best N matches with their respective scores (between 0 and 2, where 0 is best and 2 is worst).

This approach performs well so long as there is only one object in the image, i.e., if we were able to nicely segment

out clusters as described in Section IV-A. If two or more objects are visible in the input image, and especially if more than one of them is also loaded in the database, the performance decreases. This happens because the database retrieval mechanism tries to find an image containing all of the objects together and, although the objects can still be detected, their scores are low and very similar. This makes it very difficult to tell which match truly corresponds to the object in the image.

In order to improve recognition performance in such cases we thus present a power-horse idea of this paper, namely, a clustering of features of the input image in 2D space (the position of the feature in the image). In this way we can find rich-textured sub-regions in the object candidate image. It is difficult to make the clustering algorithms find the exact regions of the objects, but our experiments show, that this is indeed not necessary. If we adjust the clustering algorithm to over-segment, we get several clusters per object. These clusters correspond to the strongest textures of the objects and are, in most cases, enough to identify the whole object (see Figure 4).

Fig. 4. Detection of objects by partial textures. Left part shows that only a "Jacobs" sign is sufficient, while the right part implies the same for a "Kronung" sign.

The next step is to generate a document for every cluster size greater than the predefined size $S_{cluster}$ and query the database with those documents. Typical values for the $S_{cluster}$ are between 20 and 30, because smaller clusters are unlikely to produce meaningful results. Thus, every cluster has its own ranking of the most probable matches and we need to merge the results. In order to combine the results from every cluster into one final list of matches, we sum the scores ($clusters_{scores}$) which result from matching of every cluster against every image in the database. In this way, if several clusters vote with a high score for a specific image in the database, we understand that it is very likely that we have found the right object in the image. Note that if we had two objects in one input image, which also have respective entries in the database, then we will get more than two clusters from the input image (thanks to the over-segmentation) and the database retrieval mechanism will not search for the documents containing both objects, but rather only for parts of the objects, which will result in far more distinctive scores.

The final consensus is that, as our segmentation method tends to over-segment, the *ODUfinder* considers the image regions that could spatially lie on the same objects as multiple evidence for the respective objects and combines the evidences provided by the individual regions. Obviously the visual region-based object model appearance is particularly appropriate to handle partly obstructed objects and those which might have parts that cause reflections.

VI. Incremental Build-Up of Incomplete Models - Implementation

The *ODUfinder's* primary mode of operation provides basic functionality for online learning of new appearances of objects and mechanisms for storing and reloading them. We consider this feature to be very important for the continuous operation of service robots.

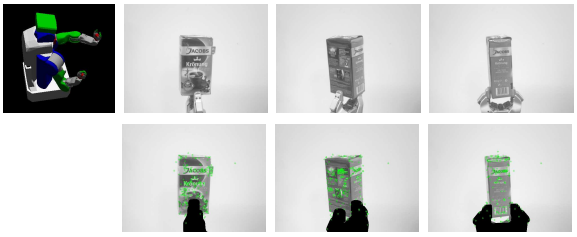


Fig. 5. **Top row:** Robot (left-most image) is manipulating an object in front of the camera **Bottom Row:** Extraction of keypoints and masking of robot's parts.

In this *ODUfinder's* mode of operation two cases may emerge: i) either the objects' appearances have been learned a priori and they only have to be located in the perceived scene, or ii) the robot encounters unknown objects (or unknown views of objects) and the new views have to be learned incrementally. While in the first case just a direct query for each appearance of the object candidate in the database is performed, in the second case we have to a) verify whether we have a partial template model of the object in question and, if so, b) the missing templates have to be acquired, features extracted and quantized with an existing vocabulary tree, and added to the existing database. In order to acquire missing object templates we implemented an in-hand object articulation and modelling process which is best explained through the following steps:

- classify object as unknown if the sum of clusters' scores $clusters_{scores} < 0.5$,
- calculate object grasp points on object's cluster [12],
- grasp the object, bring it in the frustum of the camera and set it upright,
- rotate the object around the up-right (z) axis to a viewpoint where you verify that it matches a template (from e.g. Germandeli),
- mask out parts of the robot and extract keypoints and region of interest,
- build documents from the keypoints, quantize them with the existing vocabulary tree and add them to the database,
- repeat above three steps until object has been rotated for $2\pi rad$ (note that our TUM-PR2 robot is equipped with the continuous revolute wrist joint). Also see Figure 5.

An important aspect of the learning of new models is that the vocabulary tree does not need to be computed again. The addition of a new document in a large database does not change the distribution of the words in the database substantially and therefore the existing quantization provided by the vocabulary tree is still adequate. Regeneration of the tree (and consequently of the database) is only needed if lots of new documents are added to a relatively small database, but this could be done later in an offline phase. A demonstration of the incremental build-up of incomplete models are available in the accompanying video submission [4] and Figure 5.

VII. Evaluation

A. Database Training

To evaluate our approach we have trained two vocabulary trees and built two databases with textured objects. In the first case we parsed the Germandeli website, downloaded product descriptions (semantic data, such classes of objects as well as appearances) and in the second case we generated a database out of the *Semantic3D* initiative which consists of over 40 household objects (see Figure 6) as described in [13]. The latter database was enriched with 10 more objects from the Germandeli website in order to demonstrate incremental build-up of additional models. While K,L parameters for the structure of vocabulary trees were 6,6 and 5,5 respectively, the rest of the properties of the databases are given in Table I.

	nr. images	**nr. features**	**training time**	**cluster query time**
Germandeli	3500	2500000	1h	90ms
Semantic3D	170	65000	1min	50ms

TABLE I

Technical data for the generated databases of objects.

Fig. 6. A subset of the collection of objects from Semantic3D database.

B. Recognition Results with Over-Segmentation

Used test images were taken with the hand-held camera in a German grocery store and encompass a wide variety of grocery products. We carried out recognition tests against the Germandeli database and present and discuss the results in Figure 7. We show the segmentation in feature space, where the circles denote SIFT keypoints and adjacent points with the same color belonging to the same cluster. The left part of every box in the Figure is the image received from the camera and the right image is the first match from the database.

In the first three rows of Figure 7 we see examples of the detection of 3 different objects. In only one case is the correct image not the first match found (row 3, column 3) and we attribute this to the test image's lack of the resolution.

The fourth row presents an interesting case. The camera image contains a strawberry juice, but the best match in the database is a juice with similar packaging, but of a different flavor. If we take a look in the top 10 matches for the test images in this row, we see that the first 6 matches are juices of the same make with very similar packaging, which differ only in respect to the small text in the middle of the package and in the flavor drawing on the bottom. This case is especially difficult because the strong texture from the Rauch and Happy Day logos and the upper part of the packaging are identical in all templates. This is why the correct flavor is not always first place, but in the top 5 matches. Thus, our system can find the right class of an object, in this case Rauch juice, but fails to find the right instance (in this case, flavor).

Fig. 7. Evaluation of recognition of objects found in German supermarkets with over-segmentation

C. Recognition Results with Combined 2D-3D Object Candidate Detection

We ran this test in our kitchen laboratory (see left column of Figure 8). The test was carried out against the SemanticDB database on a total number of 12 objects located at 4 different scenes (denoted with Scene 1 ... Scene 4 and depicted in top-down order in the right column of Figure 8). The robot was programmed to navigate to each of the scenes and capture

[4] http://youtu.be/Hjwj0YN2z5w

point clouds and images from several different views by traversing along the free paths around the scenes. The partial and total results of the evaluation are given in Table II.

Scene	#Views	#Failures	Success Rate [%]
Scene 1	52	10	80.7
Scene 2	11	5	54.5
Scene 3	24	2	91.6
Scene 4	12	0	100
Total	99	17	82.8

TABLE II

RECOGNITION OF OBJECTS USING SIFT WITH VOCABULARY TREES FROM COMBINED 2D-3D OBJECT CANDIDATE DETECTION.

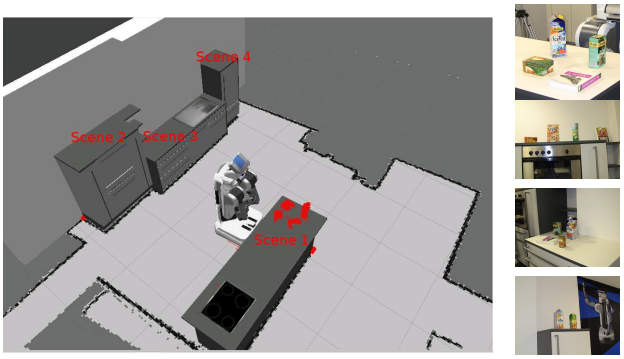


Fig. 8. Left column: We performed the final evaluation test on a total number of 12 objects located at 4 different scenes in our kitchen lab (denoted with Scene 1 ... Scene 4 and depicted in top-down order in the right column). Left column: The robot was programmed to navigate to each of the scenes and capture point clouds and images from several different views by traversing along the free paths around the scenes. Results of this test are presented in the Table II.

1) Novel Object Case: To demonstrate the capability of our system to acquire new object models on the fly we set up Scene 1 with 1 unknown object (green coffee box), which generated all 10 false positive measurements reported in the first row of Table II. Since setting the score value of the database retrieval mechanism to the experimentally determined value of *0.5* enables us to classify all measurements that exceed this value as unknown, we can introduce image templates generating this score as new object models. The assumption we are making here is that the scene remains static and that the image templates have consistent association with the cluster cloud with fixed 3D position in the world coordinate frame.

Scene	#Views	#Failures	Success Rate [%]
Scene 1	52	2	96.0

TABLE III

IMPROVED RECOGNITION RATE FOR SCENE 1 FROM FIGURE 8 AFTER THE FEATURES FOR GREEN COFFEE BOX WERE ADDED TO THE DATABASE.

After this we performed another test run on Scene 1 with the the updated database of SIFT descriptors and were able to reduce the number of false positives down to 2, as shown in Table III. Please also refer to the accompanying video submission.

VIII. CONCLUSIONS AND FUTURE WORK

This paper has presented a perception system for autonomous service robots acting in human living environments, coined the *ODUfinder*. The perception system enables robots to detect and recognize textured objects of daily use, it ensures real-time and robust operation and is modular with respect to the integration of new components (e.g. detection of texture-less or translucent objects). On the theoretic part, we consider an over-segmentation of image regions and the combination of the evidences of the detected candidate parts to infer the presence of the object, to be a major contribution herein.

In the future we plan to improve the segmentation of cluttered scenes using graph-based methods [14] and interactive perception approaches [15]. Furthermore, we plan to include more recognition routines (e.g. Dominant Orientation Templates [16], Transparent Object Detection [17]) and thus convert the *ODUfinder* into a bag-of-experts system. En route to ensure autonomous, continuous operation of the robot over large spans of time, we plan to look into i) sharing of model libraries between different robots and ii) inferring of semantic types of objects using barcodes.

IX. ACKNOWLEDGMENTS

We would like to thank our colleagues Thomas Rühr and Monica Simona Opris for their valuable support with our experiments. This work is supported in part within the DFG excellence initiative research cluster *Cognition for Technical Systems – CoTeSys*, see also `www.cotesys.org`.

REFERENCES

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[2] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 2161–2168.

[3] H. Nakayama, T. Harada, and Y. Kuniyoshi, "Ai goggles: Real-time description and retrieval in the real world with online learning," *Computer and Robot Vision, Canadian Conference*, vol. 0, pp. 184–191, 2009.

[4] J. F. Georg Arbeiter and A. Verl, "3d perception and modeling for manipulation on care-o-bot 3," in *In Proceedings of the ICRA 2010 Workshop: Best Practice in 3D Perception and Modeling for Mobile Manipulation*, Anchorage, USA, 2010.

[5] K. H. Strobl, E. Mair, T. Bodenmüller, S. Kielhöfer, W. Sepp, M. Suppa, D. Burschka, and G. Hirzinger, "The Self-Referenced DLR 3D-Modeler," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, MO, USA, October 2009, pp. 21–28, best paper finalist.

[6] R. Triebel, J. Shin, and R. Siegwart, "Segmentation and unsupervised part-based discovery of repetitive objects," in *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain, June 2010.

[7] S. Robertson, "Understanding inverse document frequency: On theoretical arguments for idf," *Journal of Documentation*, vol. 60, p. 2004, 2004.

[8] R. B. Rusu, I. A. Sucan, B. Gerkey, S. Chitta, M. Beetz, and L. E. Kavraki, "Real-time Perception-Guided Motion Planning for a Personal Robot," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, MO, USA, October 11-15 2009, pp. 4245–4252.

[9] V. Pradeep, K. Konolige, and E. Berger, "Calibrating a multi-arm multi-sensor robot: A bundle adjustment approach," in *International Symposium on Experimental Robotics (ISER)*, New Delhi, India, 12/2010 2010.

[10] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proceedings of the International Conference on Computer Vision*, vol. 2, Oct. 2003, pp. 1470–1477. [Online]. Available: http://www.robots.ox.ac.uk/~vgg

[11] "Fast sift image features library." [Online]. Available: http://sourceforge.net/projects/libsift/

[12] M. Ciocarlie, K. Hsiao, E. G. Jones, S. Chitta, R. B. Rusu, and I. A. Sucan, "Towards reliable grasping and manipulation in household environments," in *Proceedings of RSS 2010 Workshop on Strategies and Evaluation for Mobile Manipulation in Household Environments*, 2010.

[13] Z.-C. Marton, D. Pangercic, R. B. Rusu, A. Holzbach, and M. Beetz, "Hierarchical object geometric categorization and appearance classification for mobile manipulation," in *Proceedings of 2010 IEEE-RAS International Conference on Humanoid Robots*, Nashville, TN, USA, December 6-8 2010.

[14] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *Int. J. Comput. Vision*, vol. 59, no. 2, pp. 167–181, 2004.

[15] D. Katz and O. Brock, "Manipulating articulated objects with interactive perception," in *IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena,USA, may 2008.

[16] S. Hinterstoisser, V. Lepetit, S. Ilic, P. Fua, and N. Navab, "Dominant orientation templates for real-time detection of texture-less objects," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010.

[17] U. Klank, D. Carton, and M. Beetz, "Transparent object detection and reconstruction on a mobile platform," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May, 9–13 2011.

Improving Object Detection and Recognition for Semantic Mapping with an Extended Intensity and Shape based Descriptor

Erickson R. Nascimento Gabriel L. Oliveira Mario F. M. Campos Antônio Wilson Vieira

***Abstract*— We propose BASE, an extended descriptor for RGB-D images, that efficiently combines intensity and geometrical shape information to improve discriminative power. We use this new descriptor to detect and recognize objects under different illumination conditions and apply it within an adaptive boost classification framework to provide semantic information in a mapping task. We compare the performance of our descriptor against two standard ones in the literature. Experimental results show that in spite of the simplicity of the descriptor and of the Adaboost training approach, high accuracy classification is obtained with fast processing time.**

I. INTRODUCTION

In order to achieve higher levels of abstraction, mobile robots must be able to build structured representations of their environment through categorizing spatial information [20]. Such categorization can be used to generate semantic information which would enable robots to distinguish objects, to identify events and to execute high-level tasks. The importance of including semantic information to understand the environment has been advocated in several works, of which [11] and [4] are examples.

The richness of information engrafted in images has naturally driven the use of image based techniques in several categorization methods. Therefore, image based object detection and recognition are among the fundamental issues in Computer Vision and Robotics, and constitutes the core of important tasks such as tracking and Simultaneous Localization And Mapping (SLAM).

Visual classification tasks are typically tackled by extracting image features which are then used to represent individual characteristics of objects and classes. The high dimensionality of data is then greatly reduced enabling increased performance of the matching process and the reduction of memory usage both in training and in recognition steps. Therefore, feature point descriptors are at the heart of a large number of state-of-the-art of classification methodologies. The Computer Vision literature enrols numerous works which use different cues for recognition based on appearance, such as Scale Invariant Feature Descriptor (SIFT) [15], Speed Up Robust Descriptor (SURF) [1], Random Ferns [18] and Binary Robust Independent Elementary Features (BRIEF) [3].

The authors are affiliated with the Computer Vision and Robotic Laboratory (VeRLab), Computer Science Department, Universidade Federal de Minas Gerais, MG, Brazil. This work has been supported by grants from CNPq, CAPES and FAPEMIG. E-mails: {erickson,gabriel,mario}@dcc.ufmg.br

Antônio Wilson Vieira is also affiliated to CCET, Unimontes, MG, Brazil. E-mail: awilson@dcc.ufmg.br

In nearly all these approaches, features are obtained from images alone, and other information such as geometry are rarely used. Consequently, common issues with real scenes such as variation in scene illumination and textureless objects may dramatically decrease the performance of image only based classifiers. This shortcoming has fostered the development of descriptors that include the 3D shape, such as Spin-Image [9].

The combination of visual and shape cues, which is a very promising approach for object recognition, is still in its infancy. However, as far as efficacy is concerned, Lai et. al. [13] have already shown that the combined use of intensity and depth outperforms view-based distance learning using either intensity or depth alone. The reason that many descriptors have not used shape information can be partially explained by the fact that until recently object geometry was not easily and quickly obtained so as to be synchronously combined with image feature data.

With the recent introduction of fast and inexpensive RGB-D sensors (where *RGB* actually implies trichromatic intensity information and *D* stands for depth) the integration of synchronized intensity (color) and depth has become feasible. RGB-D systems outputs color images and the corresponding pixel depth information, enabling the acquisition of both depth and visual cues in real-time. These systems have opened up the opportunity to obtain 3D information with unprecedented richness. One such system is the Kinect [16], a low cost commercially available system that produces RGB-D data in real-time for gaming applications. It can be quickly and easily integrated into robots enabling the execution of several tasks, including mapping.

In this paper, we propose a descriptor composed of Binary Appearance and Shape Elements (BASE) based on BRIEF that efficiently combines texture and geometrical shape information to improve discriminative power. We use this new descriptor to detect and recognize objects under different illumination conditions and use it in a adaptive boost classification framework to provide semantic information in a mapping task. Experimental results presented later in the paper show that in spite of the simplicity of the descriptor and of the *Adaboost* training approach, high accuracy classification was obtained with processing time in the order of few milliseconds running on current processors.

The main contributions of this paper are: i) A feature point descriptor which efficiently merges appearance and geometrical information, and ii) a simplified and fast object recognition and detection technique based on *Adaboost* approach which is largely robust, specially to variations in

illumination.

II. RELATED WORK

SIFT [15], SURF [1], and more recently BRIEF [3], are the most used algorithms for keypoint extraction and descriptor creation for 2D images. The first two build their feature detectors and descriptors upon local gradients and specific orientations to achieve rotational invariance. BRIEF uses binary strings to build a descriptor that can be computed using simple intensity difference tests, which leads to a small memory usage and low processing time.

Keypoint extraction from 3D data has been successfully obtained by *spin-image* [9], which creates a 2D representation of the surface patch surrounding a 3D point. Object borders constitute an important challenge that has been tackled by another descriptor for 3D point clouds known as Normal Aligned Radial Feature (NARF) [22], which identifies borders of objects based on transitions between foreground and background.

Other works have been reported in the literature that combine image intensity with range (which very often are obtained by lasers). Those works usually have to restrict the scope to the image being analyzed, but nevertheless produce interesting results, in spite of the constraints imposed by the limited view of the environment. Our method also aims at fusing intensity and range information to enrich the discrimination power of our semantic extraction phase. Three dimensional mapping [7] and object recognition [13] have shown results on public available RGB-D object dataset [12].

If on one hand image texture information can usually provide better perception of object features, on the other hand depth information produced by 3D sensors is less sensitive to lighting conditions. Our descriptor brings forth the advantages of both texture and depth information with a small memory budget and fast processing time. Our descriptor has been used in semantic knowledge acquisition during the map building process of environments with several objects.

Recent works in semantic mapping may be divided into three basic groups. The first group is comprised of approaches that focus on cognitive maps. Object location in the environment is the main goal of the methodologies in the second group and the third group is composed by those efforts that seek to classify each unit of the discretized space into semantic categories.

Contemporary techniques on image based topological mapping have been termed *cognitive mapping*. These techniques have been discussed in some recent works that address the problem of adding semantic information to geometric maps [23], [26], [19], [8], [10]. Krishnan et al. [10] propose a hybrid map which combines semantic and topological cues. The top layer of their map is a semantic graph in which each node is a topological entity such as a hallway or room, and the edges represent the transition regions between those classes, for example, a door. Tapus et al.[23] proposed an incremental and automatic topological mapping which deals with the problems of cycle closure and of a non static environment. Their mapping updating is based on the entropy of a probability distribution over the possible positions of the robot.

The second group on semantic mapping consists of spatial representations characterized by objects which can be used to distinguish between different regions of the final map. Galindo et al. describe a navigation approach which is composed of symbolic commands [6]. In [20] the authors deal with SLAM in dynamic environments, where the robot detects and tracks natural landmarks, which are later used for relocalization and to handle uncertainties both in the system and in the environment.

The third group of semantic maps consists of a hierarchical representation, where the discretized spatial information is related to a label or annotation in the semantic information, of which [14], [28], [17], [25], [27] are examples. Lookingbill [14] developed a method for learning models of the environment using as reference the activity observed locally, and called it *activity-based map*.

In this work we describe and apply our descriptor to improve object recognition for building semantic maps, more specifically, *cognitive maps*.

III. METHODOLOGY

In this section we detail the design of our descriptor and show that it can be used to recognize objects, and finally we show how it can be used to efficiently produce semantic maps.

Objects are modeled as weighted sets $\mathcal{O}$ of descriptors $\mathbf{f_o}$ computed using selected points called keypoints. A judicious choice of these keypoints ascertains not only a good object detection from multiple views, but also a decrease of the search space making it adequate for online applications.

The weight of each set is computed by a learning process using the *Adaboost* algorithm [5]. In order to classify a new RGB-D image we first find the nearest neighbor match for all sets of object models, followed by a voting mechanism to select the model among all weighted sets.

Unlike other descriptors that use only one type of information information such as [15], [1], [3], [18] (texture) or [9], [22] (shape), the keypoint descriptor developed in this work encodes geometrical and appearance information. Later on, when we describe the experiments, we show that the combination of visual and geometrical cues greatly improves the discrimination power of our keypoint descriptor.

A. BASE DESCRIPTOR

Our descriptor is inspired by the work of [3]. In that work the authors encoded point information as a binary string, and have shown the efficiency of this encoding in their feature point descriptor. They have also shown that in spite of the descriptor's simplicity, it yields higher recognition rates. We propose an extension to their work by embedding geometrical cues during descriptor construction.

The first step to compute the set of descriptors of an RGB-D image is the selection of a subset of keypoints $\mathcal{K}$. For that, we use an efficient keypoint detector called FAST [21]. After the detection step, each keypoint will have its corresponding

estimated 3D location in the point cloud. This enables the analysis of a small surface patch around the corresponding point on the object, based on a patch $\mathbf{p}$ defined on the image.

The center of the image patch $\mathbf{p}$ of size $S \times S$ is positioned at the location of each keypoint $\mathbf{k} \in \mathcal{K}$ found by the FAST detector. For all positions in a set of $(\mathbf{x}, \mathbf{y})$-locations we evaluate the function:

$$f(\mathbf{p}) = \begin{cases} 1 & \text{if } \mathbf{p}(\mathbf{x}) < \mathbf{p}(\mathbf{y}) \vee \langle \mathbf{N}(\mathbf{x}), \mathbf{N}(\mathbf{y}) \rangle \geq \rho \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

where $\mathbf{p}(\mathbf{x})$ is the pixel intensity at position $\mathbf{x} = (u, v)^T$ and $\mathbf{N}(\mathbf{x})$ is the normal vector of the projection of pixel $\mathbf{p}(\mathbf{x})$ in the point cloud. To capture the characteristic change in the surfaces we compute the dot product $\langle \mathbf{N}(\mathbf{x}), \mathbf{N}(\mathbf{y}) \rangle$ between the normals.

The final descriptor is encoded as a binary string computed by:

$$b(\mathbf{p}) = \sum_{1}^{256} 2^{i-1} f(\mathbf{p}, \mathbf{x}_i, \mathbf{y}_i). \quad (2)$$

The main reason to use a binary operator was to maintain the simplicity and computational efficiency of the descriptor. We empirically tested different operators, such as XOR, AND and OR, to fuse these information, and the best result was obtained using the OR operator. We also performed experiments with larger signatures to separately handle intensity and normal by concatenating them in order to avoid ambiguity. However, the results were similar to the approach combining them with the OR operator (which is more efficient both in processing time and memory usage). Based on these findings we chose to use the OR operator even knowing that there exists a small probability (about 5%) of ambiguity.

B. OBJECT RECOGNITION

One of the simplest methods to classify a test set of descriptors $\mathcal{T}$ as belonging to an object $\mathcal{O}$ is to find the nearest neighbors of each descriptor $\mathbf{f_t} \in \mathcal{T}$ that minimizes a distance function D:

$$g(\mathbf{f_t}, \mathcal{O}) = \min_{\mathbf{f_o} \in \mathcal{O}} \mathrm{D}(\mathbf{f_t}, \mathbf{f_o}). \quad (3)$$

Since descriptors are strings of bits, the *Hamming* distance D is used as the distance metric [3]. One of the greatest advantages of this approach, besides its simplicity, is its low computational cost. On the downside of this naïve approach is that it tends to produce several false positives in the final classification. Therefore, to improve classification, we use a multiclass discriminative algorithm which returns the probability of a datum belonging to a given class.

Our classifier is composed of binary weak classifiers h_i, $i \in \{1, \ldots, n\}$ integrated by the *Adaboost* algorithm. Each weak classifier contains a set of descriptors $\mathcal{O}$ which represents an object. The probability that a test set $\mathcal{T}$ corresponds to the object is given by:

$$h(\mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{f_t \in \mathcal{T}} \mathcal{I}(g(\mathbf{f_t}, \mathcal{O}) \leq \tau), \quad (4)$$

where $\mathcal{I}$ is an indicator function that returns 1 if the condition in the argument is true and 0 otherwise. The term $\mathbf{f_t}$ is the descriptor vector of the test object $\mathcal{T}$. The threshold τ restricts the minimum distance for a valid match.

The multi-class classifier then selects a classifier H with maximum membership probability. Hence, the class of a test RGB-D image with a set of descriptors $\mathcal{T}$ is given by:

$$c^* = \arg\max_{H \in \mathcal{H}} \sum_{i=1}^{|H|} w_i h_i(\mathcal{T}), \quad (5)$$

where $\mathcal{H}$ is the set of trained classifiers, w_i is the weight of the weak classifier h_i and c^* is the class represented by classifier H.

C. SEMANTIC MAPPING

A particle filter was used for robot localization by selecting its most probable position. The classifier returns a class label c^* for each frame acquired from the RGB-D sensor during robot navigation. If the label is different than "none" then it is indexed to the current location. Algorithm 1 describes the mapping process.

IV. EXPERIMENTS

To evaluate the performance of our descriptor and the proposed classification approach, we initially collected several images with a Kinect mounted on a Pioneer P3-AT, as shown in Fig. 2. The final dataset was composed of 17 samples of 9 objects with different shapes and textures and 30 samples of random images from our lab to represent the negative dataset, Fig. 1. Two images of each object from distinct views were used to extract the sets of keypoints to build weak classifiers in the training step.

We performed two tests: i) First we trained the classifier and verified the quality of the classification using all other images in dataset not used in the learning stage; ii) next, the objects in the dataset were randomly positioned in the

Algorithm 1 Semantic Map($\mathcal{H}$)

while true **do**
  $p \leftarrow \mathrm{ParticlefilterPosition}()$
  $f \leftarrow \mathrm{getRGBDimage}()$
  $\mathcal{K} \leftarrow \mathrm{FAST}(f)$
  $\mathcal{T} \leftarrow \{b(\mathbf{p}) | \mathbf{p} \in \mathcal{K}\}$
  Find label class c^* solving:

$$c^* = \arg\max_{H \in \mathcal{H}} \sum_{i=1}^{|H|} w_i h_i(\mathcal{T})$$

  if $c^* \neq$ "none" **then**
    $\mathrm{map}[p] \leftarrow c^*$
  end if
end while

Fig. 1. Objects used for classification and detection experiments. From left to right: Toolbox, Cone, Nomad Robot, Pionner Robot Model 2, iCreate Robot, PC, Pioneer Robot Model 1, Keyboard box and Cabinet classes. The last image is a example of negative sample used in the training and test steps.

hallways of the Computer Science building and a map with the location of each detected object was created.

Finally we compared the performance of our descriptor with SURF, a standard 2D descriptor in the literature, and with BRIEF.

Even though the comparison with SURF and BRIEF may seem unfair, our point was not to show which descriptor is best. We actually wanted to show that better recognitation performance may be attained by simply including information of different nature, such as geometry, instead of using highly sophisticated, robust descriptors.

Other approaches in the literature that use shape information or other geometric descriptor (i.e. Spin Image or NARF) would possibly provide high recognition rate but at the a substantially larger computation time on a standard CPU. We also aimed at obtaining good recognition results but at with a significantly reduced computational requirements. Therefore, we chose to compare with methods that would run under similar hardware constraints such as BRIEF and SURF.

A. Matching Performance

To evaluate the correct matching rate using our descriptor, we selected one image from the dataset in which the object was directly facing the sensor and computed the set of descriptors. These descriptors were matched with the descriptors of all others 16 images of the object as well as with the 30 negative images.

Figure 3 summarizes the result of true and false positive rates as Relative Operating Characteristic (ROC) curves for all objects in the dataset. Better matchings are closer to the upper-left corner. Six out of nine objects had their curves very close to the upper-left corner. Even though the curves of three objects (PC, Cone and Toolbox) were not as close as those of other objects, their true positive rate was larger than 80% with a false positive rate lower than 20%.

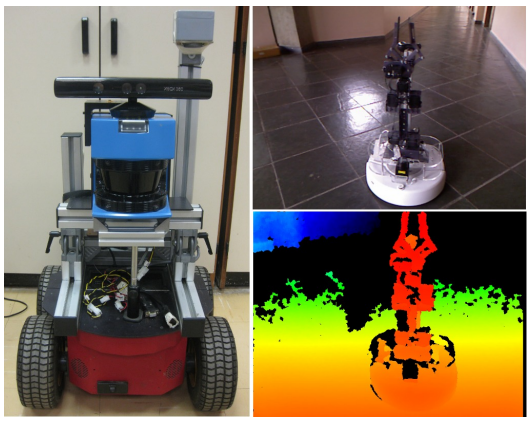


Fig. 2. Experimental Setup. On the left a picture of the mounted Kinect RGB-D camera in a Pionner P3-AT. On the right show RGB camera and the depth camera views.

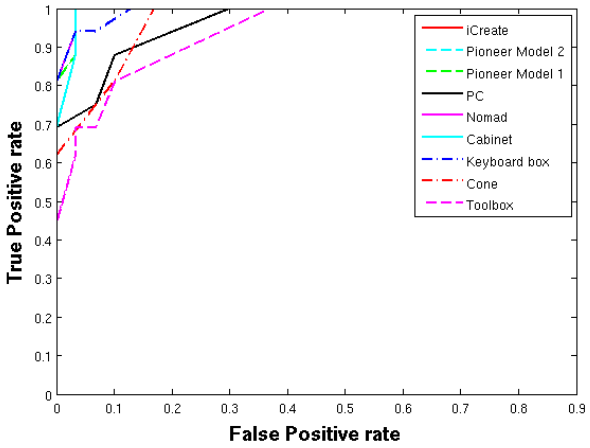


Fig. 3. ROC curve of matching using BASE descriptor. The best matching is closer to the upper-left corner. We note high true positive rate with low false positive rate for all objects in the dataset.

B. Learning and Classification Performance

Keypoint descriptors are at the heart of a large number of vision based machine learning algorithms. In spite of the ever growing performance of computer systems, the overwhelming amount on visual data now available tends to be processed and used on mobile devices with limited resources. Therefore faster and efficient keypoint descriptors need to be developed.

In order to estimate the performance of BASE descriptor, we run 5 times and measured CPU time for the learning and classification algorithms on our dataset. We compared the performance with two intensity only descriptors: BRIEF and SURF. Fig. 4 shows that in both steps – learning and classification –, our descriptor was faster than the others. The learning time of BASE was 60% faster than SURF and 15% faster than BRIEF. For the classification step, BASE descriptor run 2 times faster than BRIEF and almost 4 times faster than SURF.

One reason why BASE runs faster than BRIEF is due to the fact that BASE includes more meaningful information to build the classifier. In our experiments we observed that BRIEF uses more than one weak classifier for its binary classifier. This leads to a matching with more than one

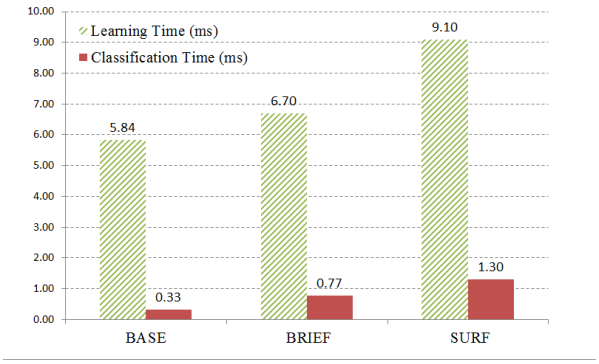


Fig. 4. CPU time for Learning and Classification steps. In both experiments BASE was faster than BRIEF and SURF. While the use of BASE in learning step is approximately 2 times of SURF and it is closer to BRIEF, in the classification performance our descriptor was almost 2 times faster than BRIEF and 4 times faster than SURF.

set of descriptors. This also demonstrates the discrimination superiority of BASE over BRIEF.

A far as memory use is concerned, BASE and BRIEF have similar performance, since both use binary strings which imply low memory utilization.

C. Classification

We now present the results of employing different types of descriptors in a low computational cost classifier. As described in Section III, we have adopted an ensembled approach using *Adaboost* algorithm. The descriptors used in the experiments were BRIEF, SURF and BASE. We train the classifiers with 9 positive samples of objects in different views and 9 negatives samples. The parameters τ and ρ used in the matching process were set experimentally.

By comparing the confusion matrices among the nine classes in Fig. 5 we observe that classification that uses BASE obtains results significantly better than the others that do not. Although the values in the confusion matrix of BASE shown in Fig. 5(a) are slightly more spread than for BRIEF and SURF, this matrix clearly shows better accuracy by the diagonal squares that can not be observed in the other descriptors confusion matrices. Also, an analysis of the BRIEF and the SURF confusion matrices shows that the classifiers built with those descriptors present a strong bias toward a given class (e.g. Toolbox).

D. Semantic Mapping

We performed experiments to evaluate different aspects of the use of the RGB-D descriptor to build a semantic map. We tested semantic RGB-D mapping spreading objects throughout the hallways of our computer science building. A Pioneer P3-AT robot navigated in the environment using the Vector Field Histogram [2] and a particle filter [24] technique. The algorithms were implemented on Player 3.0.2 and the experiments were performed on a computer running Linux on a Intel core i5 with 6 Gb of RAM.

Figure 6(a) shows the results obtained with our approach that demonstrates a superior number of true positive recognitions and higher rate of classified objects, with a small amount of false positive detections. Comparing these results respectively with BRIEF and SURF approaches, Fig. 6(b) and (c) clearly show the superiority of our descriptor both in recognition rate and robustness to false positive.

In spite of the large variation in lighting conditions of the different moments when data were collected for training and for testing, our method shows to be less affected by lighting conditions since it takes advantage of range information.

V. CONCLUSION

We have proposed a new lightweight descriptor that efficiently combines intensity and shape information to construct fast and low memory consumption signatures for keypoints. This descriptor was compared against standard descriptors in the literature on data provided by low cost RGB-D system developed for entertainment applications. Improved robotics mapping and object detection and recognition was attained in several experiments.

A dataset with 17 samples of each of the 9 objects was built to test the matching capability of the proposed descriptor. As shown in the ROC curves, our descriptor obtained high true positive detection rate with low false positive rate for all objects.

To evaluate the use of the BASE descriptor for object detection and recognition, we propose an efficient and simple framework based on Adaboost algorithm. Measurements of learning and classification time were obtained with our descriptor and two other algorithms: BRIEF and SURF. Our descriptor demonstrated superior accuracy in the confusion matrix and faster execution times for both learning and classification steps.

We have also demonstrated an application of the classification framework with the proposed descriptor in Semantic Mapping. We analyzed the performance by comparing the results with BRIEF and SURF. Again, our approach outperforms the other descriptors both in detection and recognition.

The results presented here reinforce the conclusion of [13] that merging intensity and shape information is advantageous in perception tasks. Shape and intensity information enables higher performance than using either information alone.

A larger dataset such as [12] will be used to evaluate the behavior of our descriptor for a larger number of classes. As far as mapping is concerned, we are currently investigating the use our methodology to enhance loop closure and modelling of 3D environment.

REFERENCES

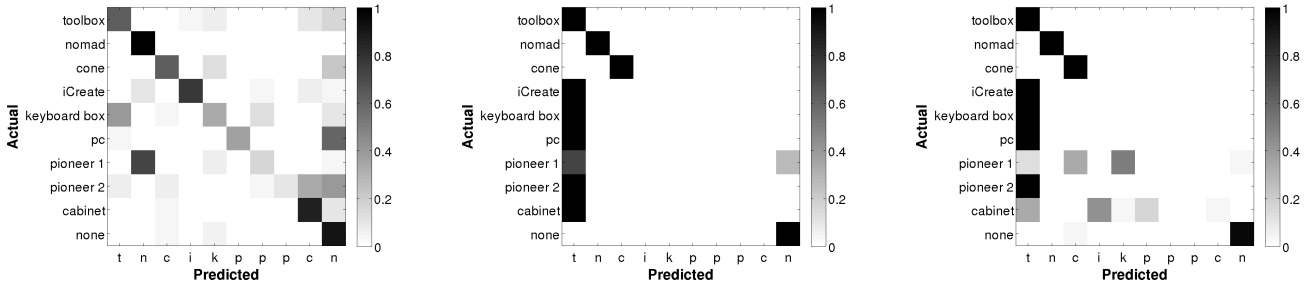
[1] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. *Proc. ECCV*, pages 404–417, 2006.

[2] J. Borenstein, Y. Koren, and Senior Member. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7:278–288, 1991.

[3] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary Robust Independent Elementary Features. In *Proc. ECCV*, September 2010.

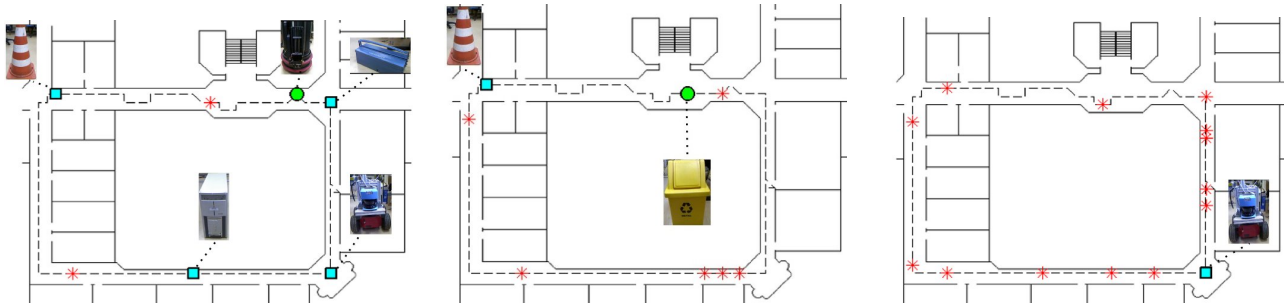
[4] R. Chatila and J.P. Laumond. Position referencing and consistent world modeling for mobile robots. In *Proc. ICRA*, pages 138–145, 1985.

[5] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proc. of the 2nd European Conf. on Computational Learning Theory*, pages 23–37, London, UK, 1995. Springer-Verlag.



(a) (b) (c)

Fig. 5. Confusion matrices (rows-normalized) between the nine classes for (a) BASE, (b) BRIEF and (c) SURF descriptor. We observe a much better classification for BASE descriptor justified by the clear diagonal on its confusion matrix even consuming less CPU time. We also note that the classifiers built with BRIEF and SURF descriptors present a strong bias toward Toolbox class.



(a) (b) (c)

Fig. 6. Semantic Map result using (a) BASE, (b) BRIEF and (c) SURF descriptor. The use of BRIEF or SURF results in a large number of false positive detection (Red stars). While was detected (Cyan squares) only one object with SURF and BRIEF detected two, BASE found six without generating too much false positive detections. Green circles mean correct detection and classification.

[6] C. Galindo, Saffiotto A., Buschka P., J. Fernandez-Madrigal, and Gonzalez J. Multi-hierarchical semantic maps for mobile robotics. In *Proc. IROS*, pages 3492–3497, 2005.

[7] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In *International Symposium on Experimental Robotics (ISER)*, 2010.

[8] Paul Newman Ingmar Posner, Mark Cummins. Fast probabilistic labeling of city maps. In *Proceedings of Robotics: Science and Systems IV*, Zurich, Switzerland, June 2008.

[9] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Trans. PAMI*, 21(5):433–449, 1999.

[10] A. Krishnan and K. Krishna. A visual exploration algorithm using semantic cues that constructs image based hybrid maps. In *IROS 2010 workshop: Semantic Mapping and Autonomous Knowledge Acquisition*, 2010.

[11] B. Kuipers and Byun Y. A robot exploration and mapping strategy based on semantic hierarchy of spatial representation. *Journal of Robotics and Autonomous Systems*, 1(8):47–63, 1991.

[12] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Proc. ICRA*, May 2011.

[13] K. Lai, L. Bo, X. Ren, and D. Fox. Sparse distance learning for object recognition combining rgb and depth information. In *Proc. ICRA*, May 2011.

[14] A. Lookingbill, D. Lieb, D. Stavens, and S. Thrun. Learning activity-based ground models from a moving helicopter plataform. In *Proc. ICRA*, pages 3948–3753, 2005.

[15] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, pages 91–110, 2004.

[16] Microsoft. Microsoft kinect. http://www.xbox.com/en-US/kinect, February 2011.

[17] M. Mozos, Triebel R., Jensfelt P., Rottmann A., and Burgard W. Supervised semantic labeling of places using information extracted from sensor data. *Robotics and Autonomous Systems*, 55(5):391–402, 2007.

[18] Mustafa Ozuysal, Michael Calonder, Vincent Lepetit, and Pascal Fua. Fast keypoint recognition using random ferns. *IEEE Trans. PAMI*, 32:448–461, 2010.

[19] I. Posner, Schroeter D., and Newman P. Using scene similarity for place labeling. In *International Symposium on Experimental Robotics (ISER)*, 2006.

[20] F. Ramos, Nieto J., and Durrant whyte H. Combining object recognition and slam for extended map reprepsentations. In *International Symposium on Experimental Robotics (ISER)*, 2006.

[21] Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Trans. PAMI*, 32:105–119, 2010.

[22] B. Steder, R. B. Rusu, K. Konolige, and W. Burgard. Point feature extraction on 3d range scans taking into account object boundaries. In *Proc. ICRA*, May 2011.

[23] A. Tapus and R. Siegwart. A cognitive modeling of space using fingerprints of places for mobile robot navigation. In *Proc. ICRA*, pages 1188–1193, 2006.

[24] S. Thrun. Particle filters in robotics. In *Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI)*, 2002.

[25] R. Triebel, R. Schmidt, M. Mozos, and W. Burgard. Instance-based amn classification for improved object recognition in 2d and 3d laser range. In *Proc. of the 20th International Joint Conf. on Artificial Intelligence (IJCAI)*, pages 2225–2230, 2007.

[26] S. Vasudevan, Gachter S., Nguyen V., and R. Siegwart. Cognitive maps for mobile robots - an object approach. In *Robotics and Autonomous Systems*, pages 359–371, 2007.

[27] U. Weiss and P. Biber. Semantic place classification and mapping for autonomous agricultural robots. In *IROS 2010 workshop: Semantic Mapping and Autonomous Knowledge Acquisition*, 2010.

[28] D. Wolf and G. Sukhatme. Semantic mapping using mobile robots. *IEEE Transactions on Robotics and Automation*, 24(1):245–258, 2008.

Objects Search : a Constrained MDP approach

Matthieu Boussard and Jun Miura
Department of Computer Science and Engineering,
Toyohashi University of Technology

***Abstract*—We consider the object search problem, where a robot has to explore its environment in order to localize some objects. We use a two step process, where the robot first detects candidate objects, and later identifies them using another algorithm. Since there are several candidate objects and the outcomes of the object recognition algorithm are uncertain, we model the planning process as an MDP. Furthermore, we give a certain amount of time to the robot to fulfil its mission. This leads to a problem where we want the robot to find as many objects as possible and as fast as possible within a limited time. This paper shows early results for the deterministic case. We model this by a Constrained Deterministic MDP, and we propose an incremental algorithm, based on a sequence of Mixed Integer Linear Program to compute the policy.**

I. Introduction

In order to perform more and more complex tasks, robots have to get a better understanding of their environment. We are considering an indoor environment, like offices or personal houses, where the robot can interact with humans. One of the most important tasks for a mobile robot is to preserve its integrity, and thus has to know where it may go safely. This issue has been well studied and has leaded to many SLAM algorithms (Simultaneous Localization and Mapping) [1]. To fulfil complex tasks, the information present on this map is not sufficient anymore. Instead, the robot has to know what kind of objects are in its environment and should be able to locate them on a map. The problem of searching and locating those objects on the map is called the objects search problem.

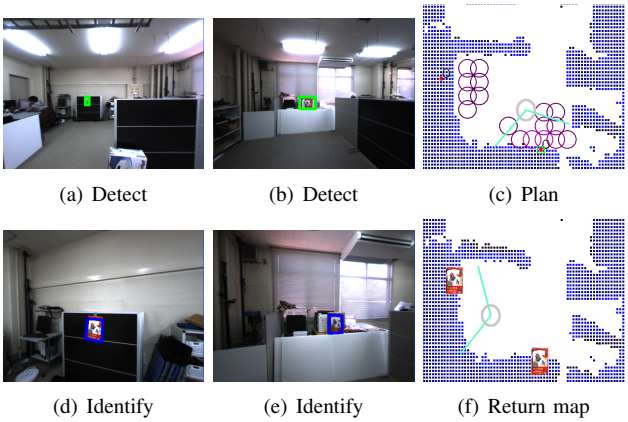
Many work has been done to search efficiently for objects. Sjöö et al. [2] present an attention mechanism and methods for depth computation, used to control the zoom level in order to perform an SIFT matching at an accurate distance measure. In [3], Meger et al. present *Curious George*, a combination between an attention system and a SLAM algorithm. This attention system allows the robot to take high definition pictures of potentially interesting area, which are used offline to perform the object detection. The principle of alternatively performing a move and an observation action is used by Shubina and Tsotsos in [4], where they compute the probability of an object's presence and the probability of an object detection using a certain type of recognition algorithm.

The lack of a long term policy, by selecting only the best next viewpoint, may lead to sub-optimal results. To obtain a long term plan, Aydemir et al. [5] are using a high level planner to select low level strategies to find a target object. The algorithm of Masuzawa et al. [6] that first detects candidates objects. Instead of directly selecting the best next viewpoint, they compute a long term policy. The authors rely on an ad-hoc world modelization in order to speed up their planning algorithm, but still, since they are performing an exhaustive search, is too slow to be solved for larger problems online.

Our problem is to recognize as many objects as possible and as fast as possible given a time limit. The contributions of this paper are the modelization of the problem as a Constrained Markov Decision Process (CMDP), a simplified Mixed Integer Linear Program (MILP) to solve it, and an incremental algorithm to control the calls to the MILP solver.

II. Problem

We define a candidate object as the location where something has been detected as potentially being a searched object. We call the location from where this candidate object can be identified a viewpoint. Fig.1 shows the object search problem. First, candidate objects are detected using a long range algorithm (here a color histogram search). Those detections are set as candidate objects on the map. Those candidates objects can be identified using an accurate and short ranged algorithm (here SIFT matches). For each object we define a set of viewpoints from where it is possible to apply the identification algorithm. We assume that this algorithm is not perfect and that we can estimate its probability of success for each viewpoint. We obtain a planning problem to select the optimal sequence of viewpoints, and once solved we can apply the selected action. At the end of the mission, we obtain a map augmented by objects information. In this paper, we will add a time constraint for the mission. We will focus on the planning algorithm, thus the exploration part will not be presented here. Furthermore, we will remove the uncertainty and will focus on how to manage the constrained planning problem.



(a) Detect (b) Detect (c) Plan

(d) Identify (e) Identify (f) Return map

Fig. 1. Object searching

III. MODEL

A. Markov Decision Processes

MDPs [7], [8] allow the formalization of a sequential decision problem under uncertainty. This process is fully observable, i.e. the observed state is the actual state of the system. A fully observable MDP is a 4-tuple $\langle S, A, P, R\rangle$

- S is the (finite) set of states,
- A is the (finite) set of actions,
- $P : S \times A \times S \rightarrow [0; 1]$ is the transition function,
- $R : S \times A \rightarrow \mathbb{R}$ is the reward function.

The unique optimal value function V^* is given by the Bellman equation [7] for the discounted expected reward for a discount factor $\gamma \in [0; 1]$. $\forall s \in S$:

$$V^*(s) = \min_{a \in A} \left(r(s, a) + \gamma \sum_{s' \in S} p(s, a, s')V^*(s') \right) \quad (1)$$

B. Observation planning model

From the raw sensor data, we need to build an MDP to compute the observation policy. We use model presented in [9], and defined by :

a) States set S: Since transitions are history independent, the states have to contain all previous information needed to take the decision. It is a way to handle the partial observability.

- the current position (x, y) of the robot,
- the list of visited viewpoints. Since the robot should not observe twice the same object from the same viewpoint, we need to keep the list of visited viewpoints $\{\{vp_1^1, vp_1^2, \ldots, vp_1^m\}, \ldots, \{vp_n^1, vp_n^2, \ldots, vp_n^m\}\}$ for all n objects, and vp_i^j is the jth observation for the ith object and m the maximum number of observations allowed.
- the information I_i about object i's status. it can be identified, rejected, or unknown.

A state s can be written :
$s = \langle x, y, \{\{vp_1^1, \ldots vp_1^m\}, \ldots, \{vp_n^1, \ldots vp_n^m\}\}, \{I_1, \ldots I_n\}\rangle$

b) Actions set A: The robot has only one kind of action. It is a macro action preforming a move action followed by an observation action. The robot selects a viewpoint, goes there and observes the related object. Its outcomes will be described by the transition function. We also add a *stop* action, available in every state, which makes the robot move to the starting position and reach the final state. This *stop* action is also executed when there is no more candidate objects.

c) Transition function P: In a general model, we consider move actions as deterministic and observation actions as stochastic. The probability of successfully identifying a candidate can be computed for each viewpoint according to known parameters (object type, location etc.). Since it is impossible to observe twice from the same viewpoint, this MDP is acyclic.

d) Reward and Cost function R: The observation planning problem can be naturally expressed by two criteria : the overall mission time, and the number of recognized objects. In [9], the authors focused on optimizing only the time criteria, whereas in this paper we want to recognize as many objects as possible given a time constraint. We define $C(s, a) < 0$ the cost (time) of executing a in s (the time to reach the viewpoint plus the time to recognize the object), and $R(s, a) > 0$ the reward for having identified an object (set to 1 if s has just recognized one object, 0 otherwise. This value can be changed to introduce preferences between objects).

In the following, we will restrict this general model to the deterministic case, where observations always succeed. This process becomes a Deterministic MDP. Observing one object immediately change its status to identified, hence the observation history can be removed from the state's definition. Even, this simplified model is a first step towards the stochastic one, the obtained plan can still be useful. For instance, it can be used to decompose the whole problem into the object observation order planning and the viewpoint planning for each object or as a heuristic value.

IV. CONSTRAINED MDP

As previously shown, the robot has to deal with rewards and costs of different natures and we don't optimize a weighted sum of the two criteria. Instead, we manage those two criteria separately through a CMDP, see [10] for a survey. It is an extension of MDP where the long term expected reward is subject to constraints on other resources. A mutli-criteria reinforcement learning algorithm has been proposed in [11] which ensures a minimum expected reward for every state before optimizing the other criterion. But it is working on a sub-class of CMDP where our problem can't be expressed.

When optimizing the number of objects using a time constraint, once satisfied, it is not optimized anymore. For instance, if a mission is given an infinite time, any policy that selects, for each object, the viewpoints having highest probability of recognition will be optimal, regardless to the global observation order! Even this behaviour is rational from an optimization perspective, it is unacceptable for a real application. Thus we have to optimize the mission's time using an expected number of recognized objects as a constraint.

The methods the most widely used are based on linear programming. The linear programming approach has been first introduced in [12]. We present here the dual of this linear program (LP) since it is more suited to solve CMDP [13]. The occupation measure $x_{s,a}$ represents the discounted number of time action a is taken in s, and $\alpha_s : S \rightarrow [0, 1]$ the initial probability distribution over states ; C being negative, the dual linear program to find the fastest policy is formulated as :

$$\begin{aligned} &\text{Maximise} \\ &\quad \sum_{s,a} C(s, a)x_{s,a} \\ &\text{Subject to} \\ &\quad \sum_a x_{s',a} - \gamma \sum_{s,a} x_{s,a}p(s, a, s') = \alpha_{s'} \\ &\quad x_{s,a} \geq 0 \end{aligned} \quad (2)$$

Once this linear program solved the optimal policy[1] can be computed by :

$$\pi(s,a) = \begin{cases} \frac{x_{s,a}}{\sum_a x_{s,a}}, & \text{if } \sum_a x_{s,a} > 0 \\ \text{arbitrary}, & \text{if } \sum_a x_{s,a} = 0 \end{cases} \quad (3)$$

It is possible to add extra constraints on the minimum expected number of recognized object R_{min} to the linear program LP.2. Those constraints are defined by the Eq.4 :

$$\sum_{s,a} R(s,a) x_{s,a} \geq R_{min} \quad (4)$$

Adding Eq.4 to LP.2 implies that the optimal policy becomes stochastic, which is not wanted. In [14] the authors showed that computing an optimal deterministic policy is NP-Complete. They compute a deterministic policy by adding a non linear constraint to the LP, $\forall s \in S, a, a' \in A, a \neq a'$:

$$|x_{s,a} - x_{s,a'}| = x_{s,a} + x_{s,a'}, \quad (5)$$

In [13], the authors change those additional constraints so that the mathematical program becomes an MILP. Since more tools are available to solve MILP than general mathematical program, the MILP may be easier to solve. They introduce $\Delta_{s,a}$ a binary variable to express the (unique) selected action a in s, and $X \geq x_{s,a}$ a constant to force $x_{s,a}/X \in [0;1[$. They compute the optimal deterministic policy by adding to LP.2 :

$$\begin{cases} \sum_a \Delta_{s,a} \leq 1 \\ x_{s,a}/X \leq \Delta_{s,a} \\ \Delta_{s,a} \in \{0;1\} \end{cases} \quad (6)$$

The CMDP defined to solve the observation planning problem, see Sec.III-B[2] has interesting properties : the starting state is known, it is acyclic and any policy will lead to the final state. Then we will use the same principle that combines LP.2 and Eq.6, but here, thanks to those properties, we can simplify Eq.6 by defining $x_{s,a}$ as binary variables and we finally propose the following MILP :

Maximise

$$\sum_{s,a} C(s,a) x_{s,a}$$

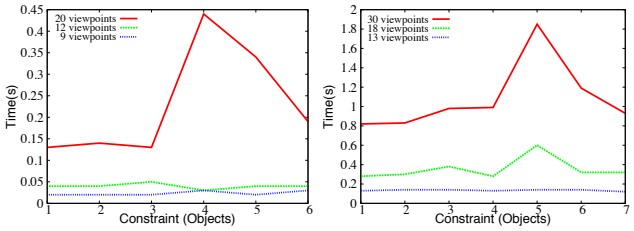
Subject to

$$\begin{aligned} \sum_a x_{s',a} - \sum_{s,a} x_{s,a} p(s,a,s') &= \alpha_{s'} \\ \sum_{s,a} R(s,a) x_{s,a} &\geq R_{min} \\ x_{s,a} &\in \{0;1\} \end{aligned} \quad (7)$$

Theorem 1: MILP.7 computes the optimal deterministic policy for an acyclic DMDP with unique and known starting state and $\gamma = 1$.

[1]Note that even the policy may appear stochastic, without constraint this policy is always deterministic

[2]We add for the constrained problem a stop action which can end the mission.



(a) 6 Candidate objects (b) 7 Candidate objects

Fig. 2. MILP solving time

Proof: The starting state s_0 is known, thus : for the starting state s_0, with $\gamma = 1$ and $\alpha_{s_0} = 1$ we have $\sum_a x_{s_0,a} = 1$. Since $x_{s,a} \in \{0;1\}$ one and only one action a_0 will be selected having $x_{s_0,a_0} = 1$. Since the problem is deterministic, there is only one state s' such that $p(s_0,a_0,s_1) = 1$, furthermore, since the MDP is acyclic, there is no other state s' such that $x_{s',a'} p(s',a',s_1) \neq 0$. We have $\sum_a x_{s',a} - x_{s_0,a} = 0$, and so on until the process reach the final absorbing state.■

V. Results

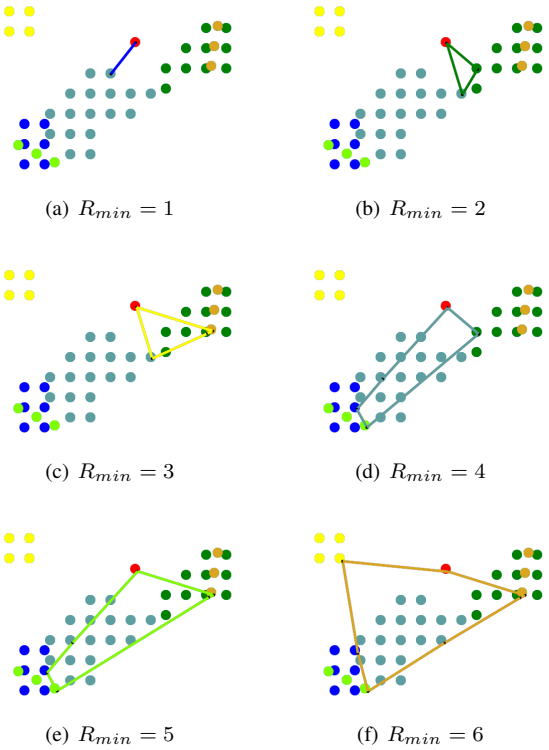
Fig.2 shows the computation time to solve the MILP for six and seven objects in the model, different number of viewpoints, and for various constraint value R_{min}. We use Ilog CPLEX with default options. For highly constrained problem ($R_{min} = 7$ obj) or low constrained problem ($R_{min} = 1$ obj), the optimal policy can be found quickly. But for "in-between" problems the computation time increases dramatically (6 candidates, 4 obj and 7candidates, 5 obj). Fig.3 shows the computed policy for different constraint values. In this picture, each color represents one object, and each circle represents one viewpoint for that particular object.

VI. Iterative MILP

We propose an iterative algorithm, Alg.1, which will at every step, find a solution to a problem constrained by a given minimum number of expected identified objects. Thus the solution found will be the fastest for that number of objects. We define $nbObj$ as the total number of candidate objects in the model. If the expected mission time $\sum_{s,a} C(s,a) x(s,a)$ is under the time limit, we increase the constraint value (line 6) in order to find a suitable plan. Fig.2 shows that some instances are very difficult to solve and should be avoided if possible. Alg.1 controls the search and can try to avoid those particular values when selecting R_{min} (line 6). For instance, if the robot has a lot of time, it can first plan for the maximum number of objects and, if succeed, doesn't need to solve for other values. When a little time remains, even many candidates could be checked, it is better to first search for a plan that recognize a few objects.

VII. Discussion

We showed how we can compute an observation plan for object recognition under time constraint. This is an early work, and the next step is to include uncertainty in the transition



(a) $R_{min} = 1$ (b) $R_{min} = 2$ (c) $R_{min} = 3$ (d) $R_{min} = 4$ (e) $R_{min} = 5$ (f) $R_{min} = 6$

Fig. 3. Policy execution for different constraint values R_{min}

function. The MILP can't simply use binary occupation measure variables anymore, we have to use one of the previous approach [13], [14]. Since the observations may fail, and since we want the robot to be able to try again, we have to keep a history of observation in the state, leading to a huge increment of the state space size. In previous works, this is solved by using Monte-Carlo algorithm and a limited horizon planning. It is possible to quickly compute a heuristic for the best policy to recognized all object (see [9]), and we can use it to get a high-level object recognition order, and then use it build an approximate, potentially sub-optimal, MILP. This observation order will force the plan to finish recognizing one object before continuing to the next one. In that case, the object observation history will be limited to the current object, the previous object being solved, and the next one not yet observed so the transition function will be limited to the current local plan.

Algorithm 1: Iterative MILP

Data: MDP $model$, mission time t_{max}
Result: π satisfying t_{max}
Generate MILP (see MILP.7) from $model$;
$R_{min} \leftarrow 1$;
repeat
  Set constraint R_{min};
  Solve MILP;
  $R_{min}++$;
until $E^{\pi}\left[\sum_{t=0}^{\infty}\gamma^t C_t | s_0 = s\right] > t_{max}$ *or* $R_{min} > nbObj$;
return π^*

VIII. Conclusion and Future Works

In this paper we presented the observation planning problem with limited time resource. We showed how we can use the properties of the observation planning problem to propose a simplified MILP. We showed early works using an iterative algorithm that solve a sequence of MILP. Once the observation planning problem is viewed as a MILP it is possible to use both the optimization techniques on the problem itself (Hierarchical planning, approximate MILP generation) or on the way of solving the MILP itself (approximate the solution of the generated MILP). Since the MILP are well-studied, having the observation planning expressed by those enables the use of many proved property, and also many efficient algorithms.

Acknowledgment

This work is supported by NEDO (New Energy and Industrial Technology Development Organization, Japan) Intelligent RT Software Project.

References

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, September 2005.
[2] K. Sjöö, D. Gálvez-López, C. Paul, P. Jensfelt, and D. Kragic, "Object search and localization for an indoor mobile robot," *Journal of Computing and Information Technology, Special Issue on Advanced Mobile Robotics*, vol. 17, no. 1, 2009.
[3] D. Meger, M. Muja, S. Helmer, A. Gupta, C. Gamroth, T. Hoffman, M. Baumann, T. Southey, P. Fazli, W. Wohlkinger, P. Viswanathan, J. J. Little, D. G. Lowe, and J. Orwell, "Curious george: An integrated visual search platform," in *Proceedings of the 2010 Canadian Conference on Computer and Robot Vision*, ser. CRV '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 107–114.
[4] K. Shubina and J. K. Tsotsos, "Visual search for an object in a 3d environment using a mobile robot," *Comput. Vis. Image Underst.*, vol. 114, pp. 535–547, May 2010.
[5] A. Aydemir, K. Sjöö, J. Folkesson, A. Pronobis, and P. Jensfelt, "Search in the real world: Active visual object search based on spatial relations," in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA'11)*, Shanghai, China, May 2011.
[6] H. Masuzawa and J. Miura, "Observation planning for environment information summarization with deadlines," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, October 18-22 2010, pp. 30–36.
[7] R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.
[8] M. L. Puterman, *Markov Decision Processes : Discrete Stochastic Dynamic Programming*. Wiley, 2005.
[9] M. Boussard and J. Miura, "Observation planning with on-line algorithms and GPU heuristic computation," in *ICAPS-10 Workshop on Planning and Scheduling Under Uncertainty*, May 2010.
[10] E. Altman, *Constrained Markov Decision Processes (Stochastic Modeling)*, 1st ed. Chapman & Hall/CRC, March 1999.
[11] Z. Gábor, Z. Kalmár, and C. Szepesvári, "Multi-criteria reinforcement learning," in *ICML*, J. W. Shavlik, Ed. Morgan Kaufmann, 1998, pp. 197–205.
[12] F. d'Epenoux, "A probabilistic production and inventory problem," *Management Science*, vol. 10, no. 1, pp. 98–108, October 1963.
[13] D. A. Dolgov and E. H. Durfee, "Stationary deterministic policies for constrained mdps with multiple rewards, costs, and discount factors," in *IJCAI*, L. P. Kaelbling and A. Saffiotti, Eds. Professional Book Center, 2005, pp. 1326–1331.
[14] E. A. Feinberg, "Constrained discounted markov decision processes and hamiltonian cycles," *Mathematics of Operations Research*, vol. 25, pp. 130–140, 1997.

Robust 3D Monte Carlo Localization using an RGB-D Camera and a Semantic Building Model

Maurice Fallon and John Leonard

***Abstract*— This paper presents a system for the robust localization of an RGB-D Camera, such as a Microsoft Kinect, in 3D. Our approach first focuses on the extraction of a low fidelity 3D model of the area of operation made up of semantic planar segments. In particular this emphasizes the extraction of major planes which are most likely to remain static (such as walls, floors, ceilings) while excluding non-planar point clouds (such as furniture) which we deem to be clutter and more likely to move in subsequent operation. Utilizing planar primitives, rather than voxel grids, is an important decision because (1) it allows for efficient 3D mapping relative to robot poses during exploration and (2) directly represents the real world structure. Using this map as input, an efficient Monte Carlo Localization algorithm is proposed which utilizes visual odometry (as the particle propagation mechanism) and coarse shape and color (to form the likelihood function). Demonstration of this approach is provided using an array of robots and human-mounted platforms. The application area of this approach is wide: including not just traditional robotic localization but many wearable and virtual reality applications.**

I. Introduction

Localization in a previously mapped environment is a key skill to enable lifelong robotic operation. Previous research has studied this problem with 2D LIDAR range finders: demonstrating localization in a 2D occupancy gridmaps [1], [2]. However these sensors typically cost several thousand dollars — a figure which is not suitable, for example, for the low-cost domestic robotic market. Instead this paper describes ongoing work to develop a robust and accurate localization algorithm for the significantly cheaper RGB-D Camera such as a Microsoft Kinect[1] in 3 dimensions.

This paper is split into two parts. In Section II we discuss the attributes of a (minimal) 3-D planar map of an indoor environment intended for localization and then demonstrate how the map can be generated using RGB-D data from a Kinect. In Section III a Monte Carlo Localization algorithm is outlined which utilizes only the RGB-D data to reliably position the sensor within this map. Finally a series of experimental demonstrations are presented in Section IV using several platforms — robotic and man-portable.

II. 3D Map Building

a) Pose Estimation: In this section we outline a procedure for the extraction of a 3D building model using a robot pose estimated using Simultaneous Localization and Mapping. SLAM is a fundamental subfield of robotics and has progressed from the initial work of Smith, Self and Cheeseman [3] through EKF and later particle filter SLAM, such as [4], to non-linear least squares optimization of the entire robot trajectory and observation set — so called full SLAM. The current state of art, such as iSAM [5] and HogMan [6], provide efficient and incremental smoothing via non-linear optimization of the underlying robot trajectory when given a series of measurement constraints — such as those provided by LIDAR scan-matching. Very accurate, large scale 2D maps can be created in this way.

At this initial stage of our project we will utilize a LIDAR-estimated trajectory created in this way to infer the motion of the RGB-D sensor (rigidly connected to the Hokuyo UTM-30LX on the push cart illustrated in Figure 5). Doing so allows us to simplify the 3D map building procedure and to focus on RGB-D-only localization in the later part of this paper[2]. The LIDAR-estimated pose is indicated in Figure 2.

b) Map Extraction: Using this accurate estimate of the Kinect sensor trajectory, we now wish to extract the planar semantic information from the RGB-D data. We assert that the largest planar polygons in any indoor environment are those that are (a) typically stationary from day to day (b) permanent structural features and (c) sufficient to form a recognizable 3D model of an area. Our thesis is that this type of model is sufficient for accurate robot localization in 3 dimensions.

Algorithm 1: Kinect Plane Extraction

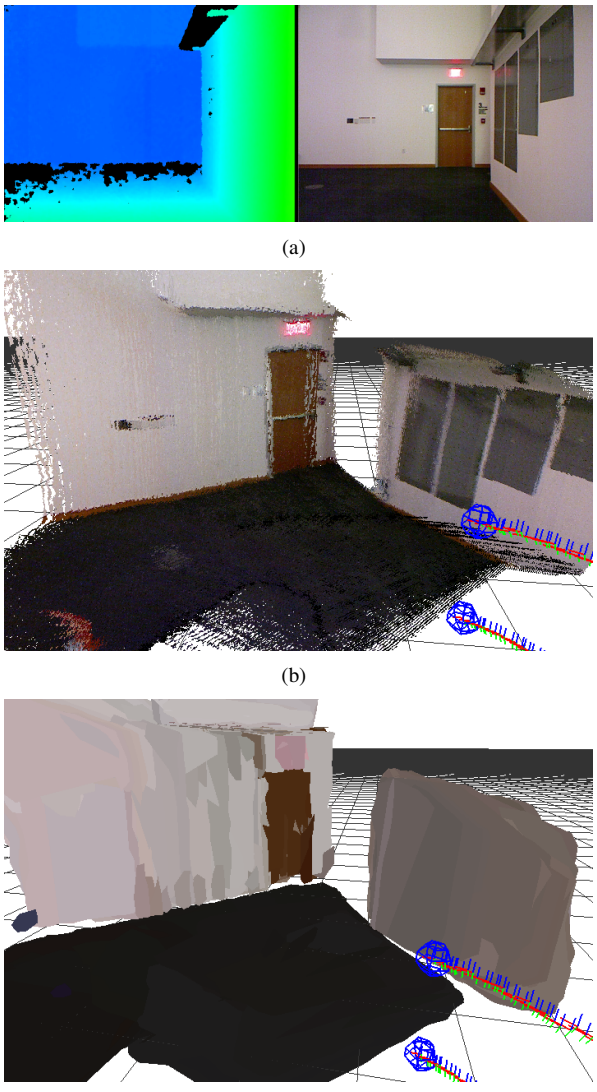
```
Given a Kinect RGB-D point cloud;
First discard all points beyond 5m range (and hence in
the inaccurate non-linear range of the sensor);
Downsample point cloud using voxel tree with leaf of
size 30cm;
while 30% of points remain or the last extracted plane
was greater than 1m^2 do
    Extract the largest plane (with points within 0.03m)
    from the remaining cloud using RANSAC;
    Remove any disconnected points from this plane;
    if the plane area exceeds 1m^2 then
        Retain the plane, its coefficients, centroid and
        pose;
        The plane color is determined to be the median
        color of the original points;
```

The authors are with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA mfallon,jleonard@mit.edu

[1]In this work we will typically refer to the Kinect, but other sensors based on infrared projection are in development

[2]Ongoing visual SLAM research, such as [7], [8], would allow us to drop the requirement for a LIDAR sensor in the map building module in future.

We present in Algorithm 1 a series of steps which extracts these planes from successive scans of RGB-D data taken by the sensor moving through an indoor building. Typically this procedure results in the extraction of 3–8 large planes representing the major planar objects within the sensor field of view. The location of the plane polygons can then be converted into the global reference frame using the aforementioned sensor pose. This procedure is then repeated for each successive scan within 5m of a central pose. Typically this results in multiple observations of the same plane, as large planes are typically observed many times as the sensor passes.



(a)

(b)

(c)

Fig. 1. Plane extraction procedure: A frame of raw kinect data (top) is first projected into a 3D point cloud (middle). Then large planar objects are successively extracted (bottom, for several consecutive frames). Also shown are the sensor pose triads.

A second algorithm (Algorithm 2) is then required which combines each planar observation into a final representative planar polygon representing a wall, floor, door or ceiling. The set of all such planes within this region then represents an accurate submap for this area, again illustrated in Figure 1. A typical submap has an extent of about 5m x 5m and are overlapped slightly so that approximately 30 submaps each containing about 30 planes can represent an entire building floor of MIT's Stata Center. We envisage that a single submap would represent an read the size of individual office rooms but no such logic or understanding is yet in place.

Algorithm 2: Planar Submap Determination

Given a set of overlapping planes;
Choose a center point for the submap (eg the pose of the sensor at some time);
Find the set of planes whose centroids are within a 5m of this point;
while *planes remain in the original set* **do**
  Choose a plane, incrementally find the set of planes which intersect it, using these conditions;
  - Both plane normals are within 4 degrees;
  - 2nd plane polygon can be projected onto the first with less then 5cm of translation;
  - The percentage overlap of the projected points is greater than 80% ;
  for *all accepted planes* **do**
    Merge into a single new plane, re-estimating the centroid and the plane coefficients and Convex Hull polygon — weighted by the number of points of the originating cloud;
  Remove all merged planes from the original stack;

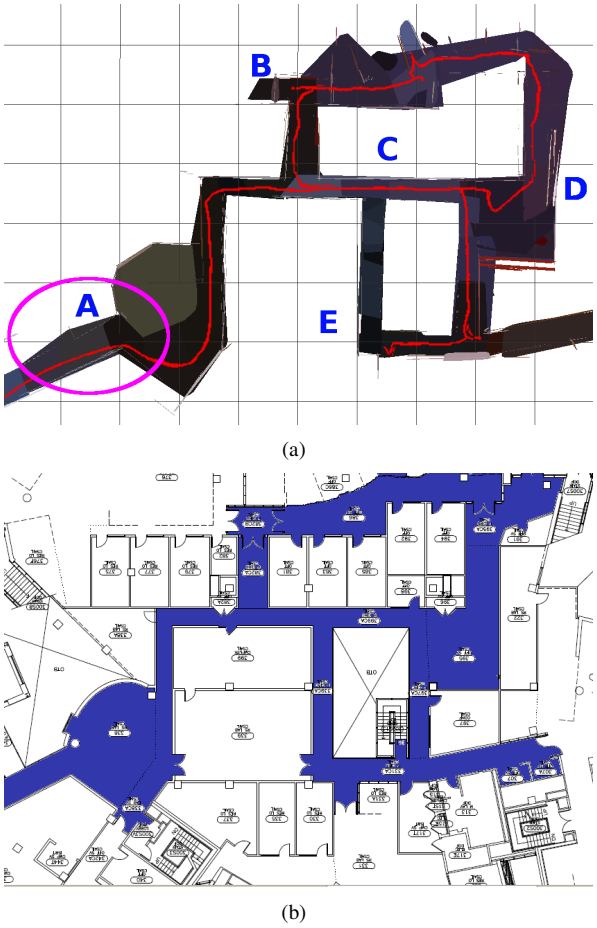
A. Model Utility and Storage Requirements

In this section we we will overview some design properties of this map and explain why we believe it to be a useful representation.

Other efficient 3D mapping and storage systems have been proposed and implemented — often building on the octree data structure. For example OctoMap [9] demonstrated efficient octree-based occupancy grid mapping allowing for compact storage of large scale maps with minimal memory and disk storage requirements. Their representations of the New College dataset of 40,000m^2 requires 50 MB of memory or 1 MB of disk space using OctoMap.

An alternative approach is proposed by Magusson et al [10] as in extension of the Normal Distributions Transform (NDT) algorithm to 3 dimensions.

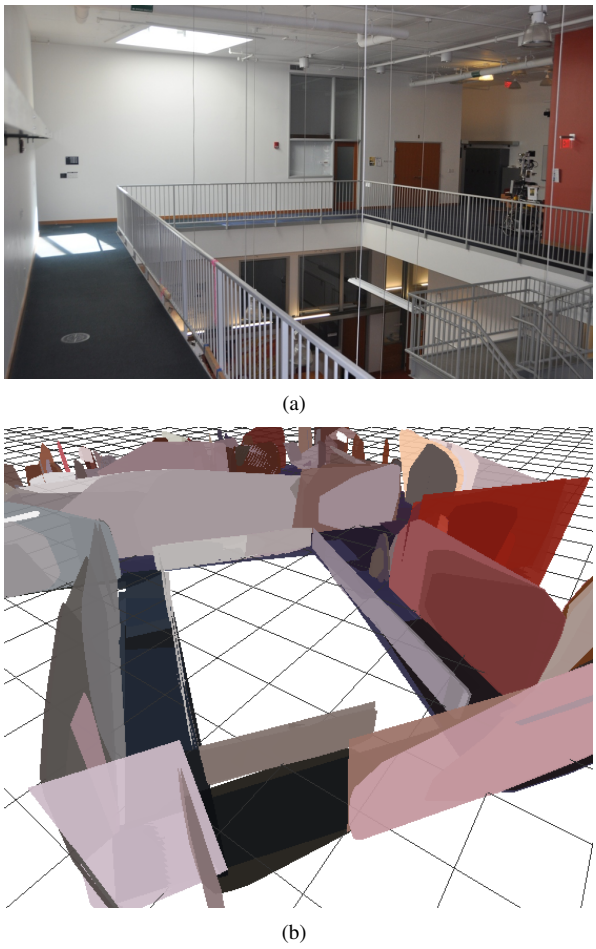
However, our thesis is that by utilizing a fixed grid of points the resultant octree is in a sense 'baked in' and disconnected from the underlying robot pose used to construct it. It is difficult to adjust the structure of the tree — should a loop closure be detected by an exploring robot. In comparison the planar polygons are connected to poses in the pose graph optimization mentioned previously by a relative transformation. Should loop closures or adjustments be detected the location of any of the planes can easily be adjusted — updating and improving the 3D model.



(a)

(b)

Fig. 2. Using the sensor pose estimated from the LIDAR (red), a global estimate of the plane locations can be inferred and an accurate 3-D model of the entire floor can be constructed as shown in this topdown view (top). Note the accuracy of the top-down model relative to the building floor-plan (bottom). The size of a typical submap is indicated by the purple loop (about 5m x 5m).



(a)

(b)

Fig. 3. Illustration of a section of the 3-D building model. Note that the model is intended to indicate the geometric information - not the precise textural information. Plane colors are for illustration purposes only at this stage.

Our representation is also efficient — we project that it will require only 2 MB of disk space to represent the entire 70,000m^2 9 floor MIT Stata Center. It intentionally does not represent small furniture, chairs, tables etc. This type of information is typically not required for long-range path planning and is often transient in any particular location. Semantic labeling of the representation is also feasible - attaching the building connectivity graph or the room function to room-sized submaps.

Nonetheless, octree occupancy maps have many uses in path planning and free space detection. We believe that algorithms can be developed to transform between the two representations to allow for complementary use.

Finally, we do not claim that the procedure outlined here is in anyway optimal and we can ourself envisage a number of optimizations to improve the appearance of these 3D plane models. However using this semantic map representation we aim to demonstrate accurate localization using only the Kinect sensor.

III. RGB-D Monte Carlo Localization

Having created the planar map, we propose to utilize particle filter localization to estimate the pose of a robot moving in this environment. Particle Filtering, more generally known as Sequential Monte Carlo (SMC), was initially proposed by Gordon et al. [11] and is well reviewed in [12]. Initial adaptations of SMC to robot localization were later reported [1], [2] using laser range finders and optionally wheel odometry to localize in 2D. Through the development of the ROS AMCL software package this approach is widely used by many researchers in the community.

Laser range finders (FOV 180–270 degrees, range 30–80m), are very different in quality to RGB-D sensors — which typically are highly non-linear beyond 5m and provide no returns at all beyond 9m. Although RGB-D does, of course, do provide 3D estimation within its 60 degree field of view. Nonetheless we wish to use the RGB-D data to estimate the sensor pose in 3D at each time-frame k

$$\mathcal{A}_k = (x_k, y_k, z_k, \phi_k, \theta_k, \psi_k) \quad (1)$$

as well as the associated velocities in each filter dimension.

However the addition of a state dimension to a particle filter typically requires an exponential increase in the number of particles.

c) Height, Pitch and Roll Estimation: While the proposed likelihood function can estimate in the full 6-DOF, it is prudent to reduce the dimensionality where possible. For this reason we will assume that the three constrained degrees of freedom — namely pitch, roll and height — can be accurately estimated independent of the particle filter. This reduces the state dimension to 3 using one of the following:

- Use of an IMU
- Assumption of horizontal motion (in the case of a ground robot)
- Direct estimation of the floor plane from the RGB-D depth data.

In the experiments in Section IV we typically estimated the floor plane direclty from the RGB-D data. The state vector will become

$$\mathcal{A}_k = (x_k, y_k, \phi_k, \dot{x}_k, \dot{y}_k, \dot{\phi}_k) \tag{2}$$

A. Particle Propagation

Our goal is to estimate the posterior distribution of the sensor state recursively using the standard two step Bayesian update rule. We will use sequential Monte Carlo methods to approximate the recursion of the non-linear and non-Gaussian system. In this way was we can represent complex probability distributions by a set of weighted Monte Carlo importance samples. We will assume that the initial state distribution, $p(\mathcal{A}_0)$, is known or can be estimated as suggested in Section V.

For each subsequent frame we will propagate the previous state estimate according to the state transition distribution, $p(\mathcal{A}_k|\mathcal{A}_{k-1})$ using the estimate produced by the FOVIS visual odometry algorithm [7]. For each dimension the propagation equations are of the form (in this case for the $\mathcal{X}$-dimension)

$$x_k = x_{k-1} + \triangle T \dot{x}_k + \mathcal{N}(0, \sigma_x^2) \tag{3}$$
$$\dot{x}_k = \dot{x}_{k,\text{vo}} + \mathcal{N}(0, \sigma_{\dot{x}}^2) \tag{4}$$

where the final term in each equation adds a small amount of normally distributed noise so as to support unexpected target motion using $\sigma_x^2 = 0.004$ and $\sigma_{\dot{x}}^2 = 0.0004$. The term $\dot{x}_{k,\text{vo}}$, is the relative (2-D) visual odometry translation estimate, $[v_{k,\text{vo}}, v_{k,\text{vo}}, \phi_{k,\text{vo}}]$, transformed into the particle's global coordinate frame

$$\dot{x}_{k,\text{vo}} = \frac{v_{k,\text{vo}} \cos(\phi_{k-1}) - w_{k,\text{vo}} \sin(\phi_{k-1})}{\triangle T} \tag{5}$$
$$\dot{y}_{k,\text{vo}} = \frac{v_{k,\text{vo}} \sin(\phi_{k-1}) + w_{k,\text{vo}} \cos(\phi_{k-1})}{\triangle T} \tag{6}$$
$$\dot{\phi}_{k,\text{vo}} = \frac{\phi_{k,\text{vo}}}{\triangle T} \tag{7}$$

Typically the period of time between frames is $\triangle T = 0.1$ seconds.

For smooth and continuous motion, the FOVIS algorithm demonstrates relative odometry estimates with a mean velocity error of 0.08m/s in typical indoor environments. However, during abrupt accelerations and sharp turning motions the feature-based visual odometry algorithm will suffer from periods of total failure. These failures are typically due to motion blur and other problems with the rolling shutter of the Kinect camera. (Given space restrictions, the interested reader is directed to [7] for extensive testing of the visual odometry.)

Fortunately, these failures are indicated by low levels of feature matching, when this is detected we will instead propagate the particle set using a noise-driven dynamical model replacing Eq 4 with

$$\dot{x}_k = \dot{x}_{k-1} + \mathcal{N}(0, \sigma_{\dot{x}}^2) \tag{8}$$

and $\sigma_{\dot{x}}^2 = 0.001$. If the failure is relatively short in duration (less than 3 seconds), it is possible for the MCL algorithm to overcome this failure entirely[3]. For longer duration failures, we envisage abandoning the current particle set and reinitializing the system anew using visual bag of words. This is discussed in Section V.

B. Likelihood Function

Having proposed the particles for the current instance, we now wish to evaluate a likelihood for each particle using the current sensor depth data and to use it to update the particle weights from the previous iteration. Typically, the majority of computing time is spent evaluating the particle filter likelihood function and we have given careful thought to its design.

Firstly we propose to down-sample the incoming data. Our experimentation has shown that from the 640x480 pixel image/cloud, a sufficiently informative likelihood function is possible using only 160–200 of the pixels. Using these points we wish to determine which of the particle poses is most justified.

d) Euclidean-based Likelihood Function: Our current likelihood function probabilisticly measures the fit between the RGB-D point cloud (translated by the proposed particle pose) and the planar submap using Euclidean distance. This is carried out in a manner somewhat like the *cost function* of the Iterative Closest Point (ICP) algorithm - although we carry out no iterative procedure.

For a given particle $\mathcal{A}_k^{(p)}$, the RGB-D cloud is first transformed onto the particle pose and the minimum point-to-plane distance is then found by comparing each point, $z_i^{(p)}$, from the cloud to each plane, s_j, in the submap mentioned above

$$d_{i,\text{min}}^{(p)} = \arg\min_j \| z_i^{(p)} - s_j \| \tag{9}$$

where $\| * \|$ represents the distance from point to plane. Given this distance, the likelihood of this point is evaluated

[3]By comparison, momentary failure of visual odometry as part of a SLAM system — for even a small number of frames — can result in problems for Vision-only SLAM

as follows

$$p(\mathbf{z}_k^i|\mathcal{A}_k^{(p)}) = \beta c_r \mathcal{N}(d_{i,\text{min}}^{(p)}; 0, \sigma_r^2) + (1-\beta) r_{\text{max}}^{-1} \mathcal{U}(0, r_{\text{max}}) \quad (10)$$

where the maximum range is $r_{\text{max}} = 3m$ and all ranges beyond this are set to r_{max}. An appropriate normalization constant, c_r, had been added for the truncated normal distribution and the measurement variance σ_r^2 is estimated to be 0.5 meters. The uniform distribution supports heavy tailed behavior and in doing so each point in the cloud has only a small effect on the overall likelihood function. The parameter $\beta = 0.01$ was found to give good experimental performance.

Finally, the overall likelihood of this particle is the product of the point likelihoods across the entire cloud

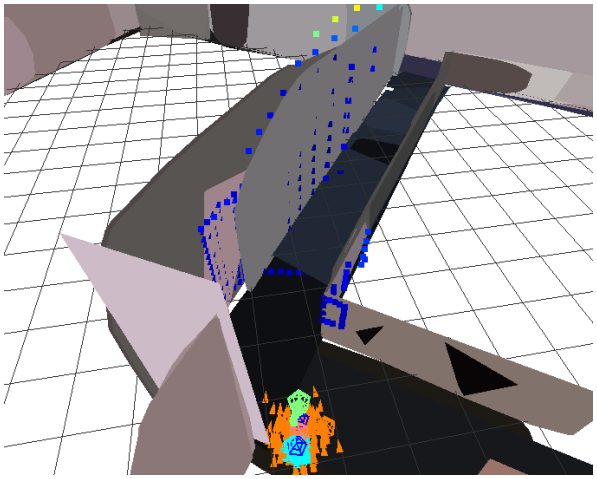
$$p(\mathbf{Z}_k|\mathcal{A}_k^{(p)}) = \prod_{i=1}^{N_i} p(\mathbf{z}_k^i|\mathcal{A}_k^{(p)}) \quad (11)$$

where N_i is the number of RGB-D points. An example of the evaluation of this likelihood function is demonstrated in Figure 4. This procedure is repeated for each of the particles in the cloud and the weights are then updated to produce an estimate of the posterior distribution at the current time

$$\tilde{w}_k^{(p)} \propto \tilde{w}_{k-1}^{(p)} p(\mathbf{Z}_k|\mathcal{A}_k^{(p)}) \quad (12)$$

Residual resampling is carried out whenever the effective sample size of the particle set falls below 0.5.

Note that ongoing work is considering replacing this likelihood function by a ray-based projective likelihood function as well as considering the contribution of color information.



(a)

Fig. 4. Example illustration showing the particle pose (green pentagon) which the best fits the 3D map given the current RGB-D data — that is the particle with the largest likelihood. The blue/green/yellows dots indicate the Euclidean distance from each RGB-D data point to the nearest plane (blue is a low distance, yellow is high). In addition, the orange triangles indicate the pose of the entire particle set. This view point corresponds to the Kinect sensor located along the left hand side of Fig. 3

IV. Localization Examples

In this section we will present a number of examples demonstrating accurate localization in the 3D map environment illustrated in Figure 2. In each case data was collected and post-processed to generate the numerical results in Table I which also indicates the path taken in each case.

Each platform had either a Kinect or a Primesense RGB-D sensor. For the man-portable system and the push cart an accurate estimate of motion was estimated using a Hokuyo UTM-30LX mounted in the primary plane of motion and was used to estimate the localization error of the MCL system. Note that the PR2 and iRobot Create ROS toolchain is currently in development.

No major failure of the localization algorithm occurred in these experiments (i.e. the entire particle set diverging) although troublesome locations containing little or no visual or geometric features do exist within the building. As indicated in the table, 2D RMS error of the order of 50cm was seen, which is encouraging at this early stage. This value is inflated due to poor performance in the aforementioned locations, which perhaps is to be expected. The data for the map-portable exhibits significant motion blur and occasional visual odometry failure; thus the results for this system indicate the robust nature of our approach.

Stable real-time operation with 150 particles has been realized on a 4-core 2.53GHz Pentium Core2 powered laptop — utilizing one core each for data capture, visual odometry and Monte Carlo localization and typically processing at 10Hz. In regions of reduced uncertainty as few as 10 particles are required with operation at several times real-time. For that reason, implementation of an adaptively resizing particle set could useful in such circumstances [13].

Finally we would like to reemphasize that only the RGB-D Kinect sensor was used in these experiments so as to demonstrate the robustness of Monte Carlo localization with such a low cost sensor. Localization of a nodding or rotating LIDAR within this planar map is straightforward and is likely to be more accurate — given the improved accuracy and range of such a sensor. Also, additional sources of odometry such as wheel odometry or an IMU could have been used to improve the prediction model and to address some of failure modes of the visual odometry mentioned above. We have avoiding doing so for simplicity and generality.

Platform	Path	Duration	Distance	RMS Error
Man Portable	ECDBCE	101	88.07	0.885
Quad Rotor	ECDC	48	35	n/a
Push Cart	ACDBC	153	95.33	0.4306
Willow Garage PR2		Toolchain in development		
iRobot Create		Toolchain in development		

TABLE I

Error of RGB-D MCL (using 150 particles) compared to LIDAR-based SLAM estimate. Units: duration seconds; distance and 2D RMS error meters. Path letters correspond to those in Figure 2

(a)

Fig. 5. Platforms used in testing, from top-right clockwise: man-portable mapping unit, Willow Garage PR2, iRobot Create, quad-rotor and a push cart.

V. CONCLUSIONS AND FUTURE WORK

This paper presented an algorithm for the extraction of a geometrically-accurate building model built using planar segments extracted from RGB-D data from the low cost Microsoft Kinect. The model can be efficiently stored and is amenable to improvement and updating.

The model was then used to localize a series of robotic and mapping platforms moving within the mapped environment using a particle filter. Our results illustrate that this Kinect-based localization algorithm is accurate, is robust to the failure of its sub-systems, as well as operating in realtime. The application area of this approach is wide: including not just traditional robotic localization but many wearable and virtual reality applications.

The work presented herein is under active development and there are many avenues for improvement of the algorithm. As mentioned above we make no use of color information. A projective error module as the ability to improve our performance in sparse environments while a GPU-based implementation has the capacity to vastly increase the speed of operation.

In addition, we would like to explore the possibility of so-called *kidnapped robot* localization using an appearance-based bag-of-words (BOW) such as [14], [15]. We envisage that when a location is proposed by the BOW algorithm, it can be used to propose a portion of the particle set which can then be used to resolve any localization ambiguity.

Finally, open loop operation of the algorithm (for point-to-point navigation) currently being investigated on a number of the platforms mentioned above.

VI. ACKNOWLEDGMENTS

This project makes significant use of the Point Cloud Library [16] (particularly in the map building modules in Section II) and the Sequential Monte Carlo Template Class [17]. Thanks to Abraham Bachrach, Albert Huang and Daniel Maturana and Nick Roy for collection of the quadrotor dataset and for use of the FOVIS library.

REFERENCES

[1] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo localization for mobile robots," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 1999.

[2] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, May 2001.

[3] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Autonomous Robot Vehicles*, pp. 167–193, Springer Verlag, 1990.

[4] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto, and E. Nebot, "FastSLAM: An efficient solution to the simultaneous localization and mapping problem with unknown data association," *Journal of Machine Learning Research*, vol. 4, no. 3, pp. 380–407, 2004.

[5] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (Shanghai, China), May 2011.

[6] G. Grisetti, R. Kümmerle, C. Stachniss, U. Frese, and C. Hertzberg, "Hierarchical optimization on manifolds for online 2D and 3D mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (Anchorage, Alaska), May 2010.

[7] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *Proc. of the Intl. Symp. of Robotics Research (ISRR)*, (Flagstaff, USA), August 2011.

[8] J. McDonald, M. Kaess, C. Cadena, J. Neira, and J. J. Leonard, "6-DOF multi-session visual SLAM using anchor nodes," in *European Conference on Mobile Robotics*, (Örbero, Sweden), 2011.

[9] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems," in *Proc. of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, (Anchorage, AK, USA), May 2010.

[10] M. Magnusson, T. Duckett, and A. J. Lilienthal, "Scan registration for autonomous mining vehicles using 3d-ndt," *IEEE Trans. Robotics*, vol. 24, pp. 803–827, Oct. 2007.

[11] N. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," vol. 140, pp. 107–113, 1993.

[12] A. Doucet, N. de Freitas, and N. Gordon, eds., *Sequential Monte Carlo methods in practice*. New York: Springer-Verlag, 2000.

[13] D. Fox, "Adapting the sample size in particle filters through KLD-sampling," *Intl. J. of Robotics Research*, vol. 22, pp. 985–1003, Dec. 2003.

[14] C. Cadena, D. Gálvez, F. Ramos, J. Tardós, and J. Neira, "Robust place recognition with stereo cameras," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, (Taipei, Taiwan), October 2010.

[15] M. Cummins and P. Newman, "Highly scalable appearance-only SLAM - FAB-MAP 2.0," in *Robotics: Science and Systems (RSS)*, (Seattle, USA), Jun 2009.

[16] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *icra*, (Shanghai, China), May 2011.

[17] A. M. Johansen, "SMCTC: Sequential Monte Carlo in C++," *Journal of Statistical Software*, vol. 30, pp. 1–41, 4 2009.

Robust Semantic Place Recognition with Vocabulary Tree and Landmark Detection

Lin Yuan, Kai Chi Chan and C.S. George Lee

***Abstract*—Semantic place recognition problem has attracted growing interests in autonomous robots to expand their application domain. Due to the large in-class variance in semantic place recognition, the recognition performance has been lackluster. In this paper, we hypothesize that the large in-class variance is due to the fact that connections between places cannot be suitably assigned a label. We verified this hypothesis on the COLD localization database. We then propose a robust method that can effectively detect these connections (landmarks), thus improving the accuracy of semantic place recognition systems. The proposed method uses image sequences for landmark detection instead of a single image, thus providing robust results which can be used for topological mapping for mobile robots under different lighting conditions.**

***Index Terms*—Semantic Place Recognition, Bag-of-Words, Visual Vocabulary, Dynamic Time Warping**

I. Introduction

The semantic place recognition of an environment that a robot is traveling will be helpful in autonomous navigation and various human-robot interaction tasks. Efforts in semantic place recognition or classification have emerged since 2005. The semantic place classification problem refers to distinguishing differences between different environmental locations (*e.g.* distinguishing a kitchen from an office). The semantic place recognition problem refers to differentiating different locations when they even may be of the same type (*e.g.* distinguishing office A from office B). Researchers first employed range sensors to solve the semantic classification problem. The distance measurements from range sensors provide a nature information about how cluttered the environment is. These measurements form well distinguishable features for different type of environments. Mozos *et al.* [1] proposed using AdaBoost algorithm for classifying different type of semantic environments (*e.g.* rooms, hallways, doorways, etc.) from range sensor readings. Various geometric measurements calculated from laser range data are then used as weak features for AdaBoost.

As robust feature extraction methods are developed in computer vision [2], [3], vision-based localization methods become a popular research topic and experiments with visual sensors have been carried out to improve the recognition performance [4], [5]. Our hypothesis (misclassification happens mostly at connection between semantic places) is inspired by research [6], [7] in the vision-based localization context. For simplifying naming conventions, we define "landmarks" to be an area on a 2D map where two semantic place units join, which is similar to Ranganathan *et al.* [8]. From now on, we will use the term "landmark" to stand for connections between two adjacent semantic places.

Lin Yuan, Kai Chi Chan and C.S. George Lee are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, U.S.A. Email: {yuanl, chan56, csglee}@purdue.edu

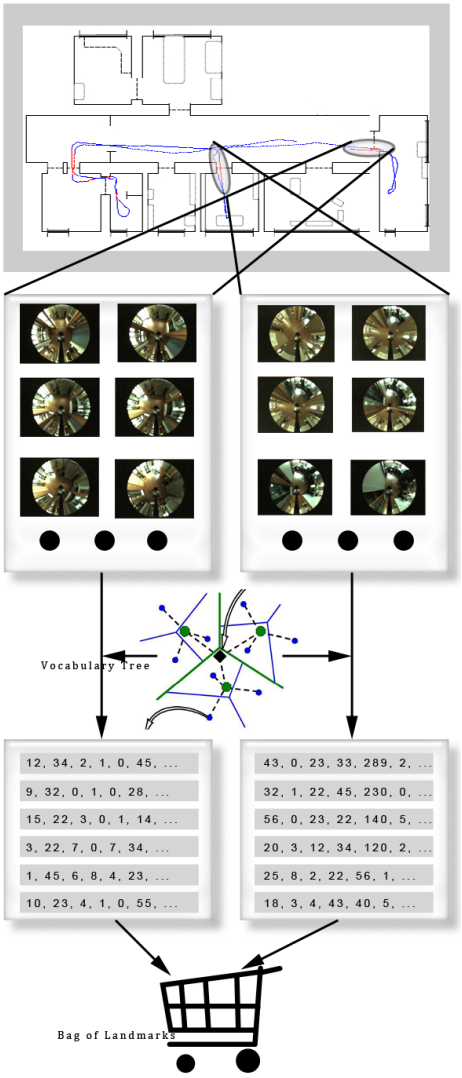


Fig. 1. Landmarks represented with time series of visual words.

II. Related Work

Semantic place recognition and topological mapping share a close relationship because each semantic place is usually a node in the graph of a topological map. And the semantic recognition problem appears earlier in the context of topological mapping. Tapus and Siegwart [9] used line and corner features in omni-directional images to generate signatures for each semantic location. Thus the topological map is represented by a collection of signatures. With a hierarchical SLAM system, Kouzoubov and Austin [10] can locate the

robot in a topological map. Friedman *et al*. [11] used Voronoi random fields to extract the topological structure of an indoor metric map. Ranganathan *et al*. [8] proposed a topological mapping algorithm without metric map. In these topological mapping literatures, a node in the topological map is usually a room on the floor plan, which is a semantic place unit.

Later on, researchers employed Nearest Neighborhood (NN) classification method for vision-based localization. In these NN classification systems, observations are made that misclassification happens mostly at "landmark" positions. Using visual cameras, Zivkovic *et al*. [6] employed SIFT features within images to match against the database images in order to locate a robot to a semantic location. The semantic location is represented with a node in the topological map built with graph-partitioning algorithms. They tested their localization performance by taking one image from the database, matching it against all other images in the database and assigning the nearest neighbor's class to the image. The database is simply a collection of images from one run of the robot at a specific time (i.e., no lighting variations). Thus they were able to achieve 90% recognition rate. However, they pointed out that the misclassification happens mostly at the boundaries between different partitions of their map. Knopp *et al*. [7] also made similar observations. They suppressed confusing features from being used for image classification to achieve higher classification rate. Valgren and Lilienthal [12] investigated the impact of different lighting conditions on the SIFT and SURF descriptors for vision-based localization systems. They pointed out that vision-based localization systems cannot achieve good recognition rate with training and testing sets across different lighting conditions based on a single image.

In recent years, the appearance-based, loop-closure problem has gained significant improvement. In the FAB-MAP system proposed by Cummins and Newman [13], they employed a bag-of-words model for images and used the k-means clustering to generate a visual vocabulary. The vocabulary itself carries moderate information about the location where an image was taken, but is pruned to various conditions. They further employed a graphical model, called Chow-Liu tree, to capture the correlation between those visual words. The resulting learned graphical model significantly helped in the perceptual-aliasing problem.

This paper proposes a new framework under which novel methods can be developed to effectively detect "landmarks" to improve the performance of semantic place recognition (*c.f*. Fig. 1) over alternative methods in [4], [5]. The proposed approach is inspired by [6], [7], [13], but uses a different method for generating vocabulary [14]. The proposed method is based on forming a time-series sample of Bag-of-Landmarks from a sequence of images. We call our method as BoLTS for Bag-of-Landmarks using time series. We generate the visual "landmarks" for a small number of images (10-80) within an image sequence where "landmarks" are identified by a human. Given image sequences collected by the robot, a human picks up segments of images where there is a "landmark". By using the visual vocabulary built with [15], a simplified version of [14], we obtained a high recognition rate for "landmarks". And by removing these "landmarks" from the training set for the semantic place recognition task, we improved the recognition rate, thus validating our hypothesis that misclassifications happen mostly at "landmark" positions. BoLTS advances the state-of-art of visual landmark recognition, by broadening the data-format into time-series with image sequences.

III. Robust Semantic Place Recognition

Our robust semantic place recognition approach consists of two components: a vocabulary tree image classifier and a landmark detector based on time-series pattern matching. We use the vocabulary-tree method [14] to generate signatures for images, which is a histogram of visual words from a pre-built visual vocabulary. These signatures are later used to form an image classifier for semantic place recognition. In order to address the specific boundary issue (*c.f*. Fig. 4) faced in semantic place recognition task, we propose to develop a time-series pattern matching approach, called "Bag-of-Landmarks using time series," for detecting "landmarks".

A. Vocabulary Tree

Bag-of-words modeling of images has been introduced by Sivic and Zisserman [16]. Clustering methods are typically employed for building the visual word dictionary. Several clustering methods (*e.g*. k-means, vocabulary tree, etc.) have been incorporated into an appearance-based localization system [13]. In this section, we will demonstrate how a visual word dictionary built with the vocabulary-tree method can be used for semantic place recognition.

The vocabulary-tree method [14] is a hierarchical iterative k-means clustering method with parent nodes being a quantization representation of their children. Even though the k-means clustering proved to be effective in various applications, we find that the visual word extracted by the vocabulary-tree method together with SIFT features is more consistent under moderate change in lighting condition. This is a crucial factor for robotics applications because the more reproducible the word is, the better the place recognition will be. Thus, we choose to use the vocabulary-tree method for our bag-of-words model with SIFT features.

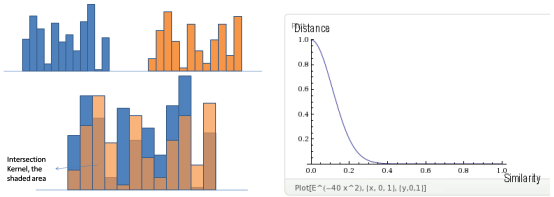
Given the signature of images, typical image categorization methods will build up a nearest neighborhood (NN) classifier from the image database, and then perform image classification. When directly applying this method to semantic place recognition, we face a boundary issue. In these cases, the images collected at the "landmark" positions are difficult for people to assign label for training. These "landmarks", as we mentioned before, are important landmarks in topological mapping [17]. Hence, we propose a landmark detection method based on time-series pattern matching, in which each index of the time series is an image signature. If a "landmark" is detected, we can preclude these images from being used for semantic place recognition, hence improving the recognition rate. In order to do the time-series matching, the distance between two signatures needs to be matched.

Instead of using the normalized difference measure [14], we use a Gaussian histogram intersection kernel measure. Define N as the number of leaves in the vocabulary-tree (dictionary size). Given a query signature $\{s_q : s_q^1 \dots s_q^N\}$ and a database signature $\{s_d : s_d^1 \dots s_d^N\}$, the histogram intersection kernel similarity $sim(s_q, s_d)$ is described in Eq. (1). Note that the similarity ranges from 0 to 1.

$$sim(s_q, s_d) = \frac{\sum_{k=1}^{N} (\min(s_q^k, s_d^k))}{\sum_{k=1}^{N} s_d^k} \quad (1)$$

Next, we need to generate a distance measure based on this similarity measure. Using images found in the database, we find that the similarity measure for two images taken within close proximity to one another can only peak around 0.2. This indicates that there cannot be a linear relationship between the similarity and the distance. Through empirical experience, we find that the Gaussian function e^{-kx^2} in the range of [0,1] is a fit for the data. Equation (2) shows the distance $diff(s_q, s_d)$ generated from the histogram intersection kernel. In practice, we used $k = 40$ for our experiments. Figure 2 further illustrates the histogram intersection kernel and the relationship between distance and similarity.

$$diff(s_q, s_d) = e^{-ksim(s_q, s_d)^2} \quad (2)$$



(a) (b)

Fig. 2. (a) Histogram intersection kernel (b) Relationship between similarity and distance measure

B. Dynamic Time Warping

Dynamic time warping (DTW) is a well-known method for matching time-series data. The advantage of using DTW over other time-series matching algorithms is that the matching between two time series can be of variable length. This advantage is especially suitable for matching image sequences collected by a robot because a robot usually can vary its velocity, thus the number of images collected by the robot over a specific range can be quite different. DTW is also a suitable choice for other distance traveled based schemes [13] since it handles the overlap of images well. Meanwhile, by applying DTW, we assume that the robot traverses through the "landmark" position in a fixed manner. This is generally not a problem if the "landmark" is like a doorway connecting two places, where the only option for the robot is to "go in" or "go out".

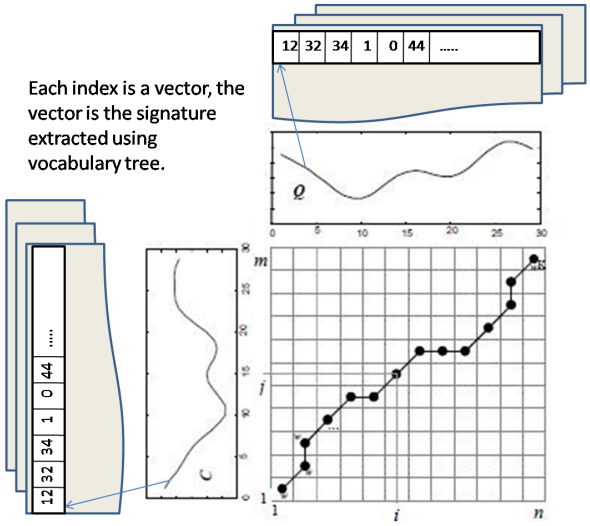


Fig. 3. Iterative Vector Dynamic Time Warping.

The DTW algorithm is a dynamic programming algorithm and is described in detail in [18]. Following Fig. 3, each time series can react with an elastic behavior such that each index of the query time series can find its best match with the index of the reference time series. When matching two time series $[L_q : s_1..s_n]$ and $[L_d : s_1..s_m]$, the cost of matching s_i and s_j, where $1 \leq i \leq n$ and $1 \leq j \leq m$ is described in Eq. (3).

$$DTW(s_i, s_j) = diff(s_i, s_j) + \min(DTW(s_{i-1}, s_j), DTW(s_i, s_{j-1}), DTW(s_i, s_j)) \quad (3)$$

C. Bag-of-Landmarks using Time Series (BoLTS)

In order to apply the DTW algorithm to robot systems, we need to solve the computational issue due to the nature of image sequences collected by the robot. When a new image comes from the visual sensor, the robot will need to concatenate it with variable length of buffered historic images as a query time series L_q. Then the set of variable length query time series is used to match against the "landmarks" of time series trained beforehand, because the "landmarks" of time series in database can have different length. The traditional DTW used here will bring about significant amount of redundant computation. Thus, we save the distance between buffered signatures after each DTW is done in an global buffer. We refer to our modified version of DTW as Iterative Vector Dynamic Time Warping (IV-DTW). Using IV-DTW, whenever a new image is collected, only one more distance needs to be calculated.

By using the vocabulary-tree method with our adapted histogram intersection kernel and modified DTW algorithm, the entire BoLTS for detecting "landmarks" can be described using Algorithm 1. The IV-DTW algorithm is described in Algorithm 2.

D. Integration

As we have introduced the vocabulary-tree method and the BoLTS method, we will put these two components

Algorithm 1 : Bag-of-Landmarks detection using Time Series

Require: m landmarks $L_1 \dots L_m$ in database. $\forall i$, $L_i = \{s_1 \dots s_{k_i}\}$, k_i is the length of the image sequence used for landmark i.
Initialize $B = 80$, $minlen = 10$
Initialize buffer D, d with size $B \times B$.
for $t = 1$ to T **do**
  Collect new image I_t, generate signature s_t.
  for $j = minlen$ to B **do**
    $dist_j$ = IV-DTW($[s_{t-j}..s_t]$,$\{L_1;..;L_m\}$,D,d)
  end for
  store $Dist_t = \min_j(dist_j)$
end for
Output: The local minimums for $Dist$ below some threshold will be the detected landmarks.

Algorithm 2 : Iterative Vector Dynamic Time Warping (IV-DTW)

Require: m landmarks $L_1 \dots L_m$ in database. $\forall i$, $L_i = \{s_1 \dots s_{k_i}\}$, k_i is the length of the image sequence used for landmark i. Query time series (sequence of signatures): $L_q = \{s_1 \dots s_{k_q}\}$. Buffered distance matrix: d and buffered cumulative distance D.
for $i = 1$ to m **do**
  if d is empty **then**
    d=$[\bigcup_{k_1=1}^{k_i} \bigcup_{k_2=1}^{k_q} diff(s_{k_1}, s_{k_2})]$
  end if
  Initialize $D_c = [d, D]^T$
  for every $d(u, v)$ **do**
    $D_c(u, v) = d(u, v) + \min(D_c(u+1, v), D_c(u+1, v+1), D_c(u, v+1))$
  end for
  Compute $dist_i$ between L_q and L_i by backtracking from the top-right cell of D_c
end for
Output: $\min_{1 \le i \le m} (dist_i)$.

together to build our semantic place recognition system. The procedure of our system is described in Algorithm 3.

Algorithm 3 : Robust Semantic Place Recognition

Require: m landmarks $L_1 \dots L_m$ in database. $\forall i$, $L_i = \{s_1 \dots s_{k_i}\}$, k_i is the length of the image sequence used for landmark i.
for $t = 1$ to T **do**
  Collect new image, generate signature using vocabulary tree.
  if BoLTS report landmark detection **then**
    Mark the matched time series as landmark.
    Don't do semantic place recognition.
  else
    Do semantic place recognition
  end if
end for

IV. Experimental Work

We validated our semantic place recognition system on the COsy Localization Database (COLD) [19]. The "bag-of-words" and "siftpp" code written by Vedaldi [15] is used for generating signatures using the vocabulary-tree method. We evaluated the performance of our system using the 10 image sequences collected on Path A within the COLD-Freiburg set. First, we demonstrated that the places for misclassification using the vocabulary-tree method are located at "landmark" positions. Next, we use each image sequence as the test set and the "landmarks" in the other 9 image sequences to form the "bag-of-landmarks". The landmark detection rates of BoLTS for all 10 image sequences

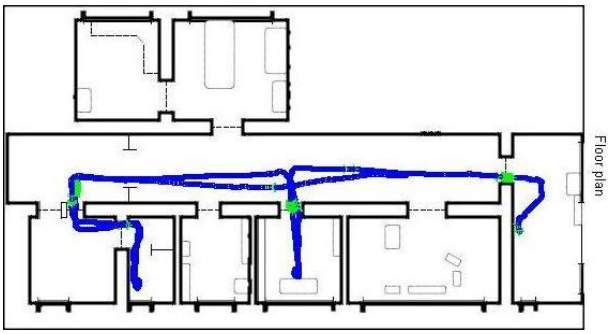


Fig. 4. Misclassified places on the map.

TABLE I
Comparison of Confusion Matrix

(a) Confusion matrix with pruning; (b) Confusion matrix without pruning

(a)

	P_1	P_2	P_3	P_4	P_5
P_1	0.9939	0	0.0061	0	0
P_2	0.0103	0.9897	0	0	0
P_3	0	0	1	0	0
P_4	0	0	0	1	0
P_5	0	0	0	0.0273	0.9727

(b)

	P_1	P_2	P_3	P_4	P_5
P_1	0.9351	0.0227	0.0185	0.0237	0
P_2	0.0163	0.9837	0	0	0
P_3	0.0181	0	0.9819	0	0
P_4	0	0	0	0.9655	0.0345
P_5	0	0	0	0	1

are discussed. Finally, we compared our system (with and without pruning using BoLTS) to existing methods [4], [5] and demonstrated the robustness of our proposed method.

A. Importance of Landmarks

Inspired by Zivkovic *et al.* [6], our hypothesis is that misclassification happens mostly at "landmark" positions. To verify our hypothesis, we used seq1_cloudy1 as a training set and tested the semantic place recognition using only the vocabulary-tree method (VTM). There are 5 semantic places on Path A, and we trained the vocabulary tree with 100 images per place. As shown in Fig. 4, the green marks indicated the images that got misclassified. The obtained confusion matrix for 5 places are compared in Table I. We can see that pruning images at "landmark" positions improved the recognition rate. Furthermore, we trained another vocabulary tree using 3 sequences (seq1_sunny1, seq1_cloudy1, seq1_night1), with 100 randomly sampled images per place. Then we tested the semantic place recognition system on 7 other sequences. The ratio of misclassified images taken at boundaries over all misclassified images is shown in Fig. 5.

B. Landmark Detection Performance

The COLD-Freiburg database [19] was used to test the performance of landmark detection. In the database, there were 10 sequences of images collected at different times under various lighting conditions. Leave-one-out cross-validation was conducted on the 10 image sets to estimate the accuracy of the detection algorithm in practice.

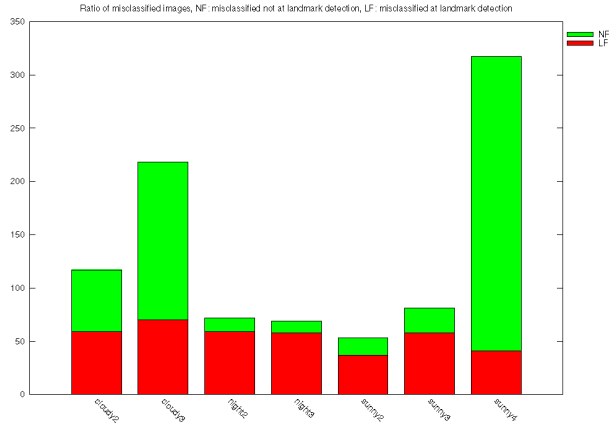


Fig. 5. Misclassification ratio. Red bar represents the misclassified images at boundaries.

In the training image sets, 8 "landmarks" were trained based on the bag-of-words generated with vocabulary tree. Then, the test images were compared to the "landmarks" using the proposed BoLTS. Figure 6 shows the distance measure for different testing image sets. Each point on the curve represents the minimum distance among distances to the 8 "landmarks" and a sequence of 10 to 80 images ending at that point. The local minima of the curve indicate the ending time of the matching with different "landmarks". Assuming that every landmark is detected at least 60 frames from each other and by sorting the distances of points at different frames, the 8 "landmarks" with at least 60 frames from each other could be found. * The end of time series (each matched landmark) is marked by a red segment in Fig. 6.

For each "landmark" detected, the corresponding sequence of images was retrieved. The accuracy of landmark detection was computed based on the sequence of images. The false positive rates of landmark detection are shown in Table II for all 10 image sequences. Since the ground truth of sequences of images at different "landmarks" were not available, the false positive rates were estimated manually by checking every image in the detected sequence of images. If the image is manually accepted as taken at a "landmark" position, it is treated as a landmark image. Detection results are shown in Fig. 7. In Fig. 7, which shows one of the detection results, the blue line represents the path of a moving robot, and the red line segments indicate the detected "landmarks". All the "landmarks" in the seq1_cloudy3 sequence were detected correctly.

C. Improved Semantic Place Recognition System

We compare our semantic place recognition system with similar visual place classification experiments performed on the same COLD-Freiburg database. We name our methods VTM (vocabulary-tree method) and VTBL(vocabulary tree with bag-of-landmarks). Wang and Lin [5] used a Hull Census Transform (HCT) method to generate features for each omni-directional image and used this HCT feature together with SVM for semantic place recognition. Pronobis *et al*. [4] reported the multi-model place classification performance on the COLD-Freiburg database. Figure 8 shows that both of our methods, VTM and VTBL, outperformed existing work.

*With a robot collecting images at 30 frames/sec, this means that the robot cannot travel from one "landmark" to another "landmark" within 2 seconds, which is reasonable since a robot usually cannot go from one door to another door within 2 seconds.

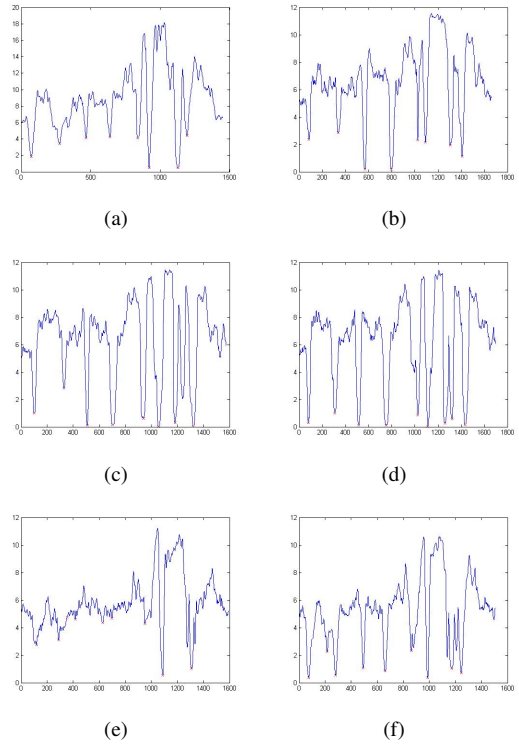


Fig. 6. Extracted IV-DTW distance for example robot run, the red crosses are identified "landmark" positions. The sequence from top-left to bottom-right is (a) cloudy1, (b) cloudy3, (c) night2, (d) night3, (e) sunny1, (f) sunny2. The local minimas show that "landmarks" are properly detected.

TABLE II
RECOGNITION RATES FOR LANDMARK DETECTION

Test	TP%	FP%	TN%	FN%	Total No.
cloudy1	17.2	0	82.8	0	1459
cloudy2	14.0	1.0	84.3	0.7	1632
cloudy3	14.5	0	85.5	0	1672
night1	15.6	0.7	83.7	0	1911
night2	16.1	0.9	81.9	1.0	1582
night3	19.0	0.9	80.1	0	1703
sunny1	11.3	0.9	81.2	6.6	1598
sunny2	17.8	1.0	81.2	0	1514
sunny3	13.8	1.8	82.7	1.7	1451
sunny4	10.4	1.5	83.2	4.9	1777

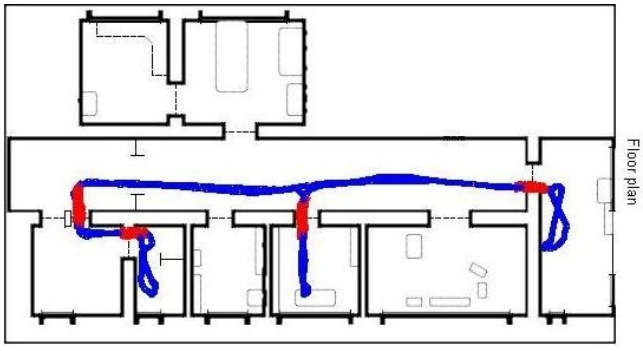


Fig. 7. An example run of landmark detection on the floor plan tested with seq1_cloudy3 image sequence.

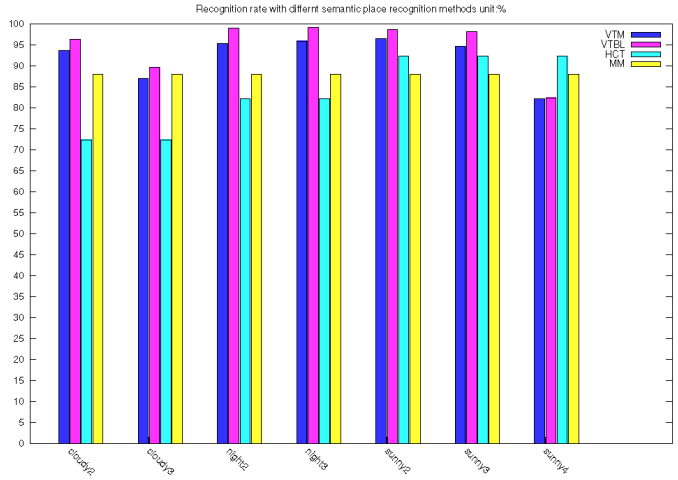


Fig. 8. Comparison of different semantic place recognition algorithms on COLD-Freiburg database. VTM: vocabulary tree without pruning; VTBL: vocabulary tree with bag-of-landmarks pruning; HCT: Hull Census Transform; MM: Multi-model semantic place classification. The result used for HCT is obtained from [5], where their best result for each weather is chosen. The result used for MM is an averaged result of their classification rate on all 3 sequences reported in their paper [4]. It is unknown which 3 sequences of COLD-Freiburg did they use.

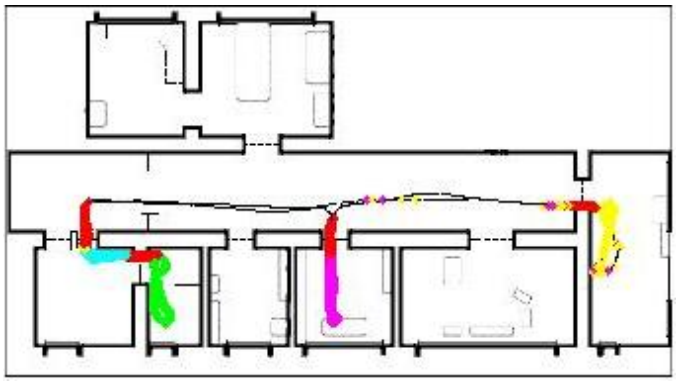


Fig. 9. Final run with our approach for seq1_cloudy3, which has 89% recognition rate.

An example run with a final label of the 5 semantic places as well as detected "landmarks" is shown in Fig. 9. More details can be found in the video attachment of this paper.

V. Discussion and Future Work

This paper proposed and developed a semantic place recognition system with vocabulary tree and BoLTS. The proposed system yielded significant improvement over existing methods for the semantic place recognition task. The proposed landmark detection method (BoLTS) is a time-series-based supervised learning approach, which is novel in visual landmark detection context. However, the preparation of training set of transition places may be very dependent on robot trajectory. We will use of salient features other than SIFT and incorporating probabilistic framework like [13] into our landmark detection method to achieve a more robust and general time-series landmark detector, thus requiring no human assistance for marking image segments.

References

[1] O. M. Mozos, C. Stachniss, and W. Burgard, "Supervised learning of places from range data using AdaBoost," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Barcelona, Spain, 2005, pp. 1742–1747.

[2] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[3] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," *Computer Vision and Image Understanding (CVIU)*, vol. 110, no. 3, pp. 346–359, 2008.

[4] A. Pronobis, L. Jie, and B. Caputo, "The more you learn, the less you store: Memory-controlled incremental svm for visual place recognition," *Image and Vision Computing*, vol. 28, no. 7, pp. 1080 – 1097, 2010.

[5] M.-L. Wang and H.-Y. Lin, "An extended-hct semantic description for visual place recognition," *The International Journal of Robotics Research*, vol. 30, no. 8, July 2011.

[6] Z. Zivkovic, B. Bakker, and B. Krose, "Hierarchical map building using visual landmarks and geometric constraints," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2005, pp. 2480–2485.

[7] J. Knopp, J. Sivic, and T. Pajdla, "Avoiding confusing features in place recognition," in *European Conference on Computer Vision*. Springer Berlin / Heidelberg, 2010, vol. 6311, pp. 748–761.

[8] A. Ranganathan, E. Menegatti, and F. Dellaert, "Bayesian inference in the space of topological maps," *IEEE Transactions on Robotics*, vol. 22, no. 1, pp. 92 – 107, Feburary 2006.

[9] A. Tapus and R. Siegwart, "Incremental robot mapping with fingerprints of places," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, August 2005, pp. 2429 – 2434.

[10] K. Kouzoubov and D. Austin, "Hybrid topological/metric approach to slam," in *IEEE International Conference on Robotics and Automation*, vol. 1, 2004, pp. 872 – 877 Vol.1.

[11] S. Friedman, H. Pasula, and D. Fox, "Voronoi random fields: extracting the topological structure of indoor environments via place labeling," in *IJCAI'07: Proceedings of the 20th international joint conference on Artifical intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007, pp. 2109–2114.

[12] C. Valgren and A. J. Lilienthal, "Sift, surf & seasons: Appearance-based long-term localization in outdoor environments," *Robotics and Autonomous Systems*, vol. 58, no. 2, pp. 149 – 156, 2010.

[13] M. Cummins and P. Newman, "FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, June 2008.

[14] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, vol. 2, 2006, pp. 2161 – 2168.

[15] A. Vedaldi, "Vlfeat." [Online]. Available: http://www.vlfeat.org/ vedaldi/code/bag/bag.html

[16] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," *IEEE International Conference on Computer Vision*, vol. 2, p. 1470, 2003.

[17] A. Ranganathan and F. Dellaert, "Bayesian surprise and landmark detection," in *IEEE International Conference on Robotics and Automation*, May 2009, pp. 2017–2023.

[18] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana, "Fast time series classification using numerosity reduction," in *Proceedings of the 23rd international conference on Machine learning*. New York, NY, USA: ACM, 2006, pp. 1033–1040.

[19] A. Pronobis and B. Caputo, "COLD: COsy Localization Database," *The International Journal of Robotics Research (IJRR)*, vol. 28, no. 5, May 2009.

Semantic Object Search in Large-scale Indoor Environments

Manabu Saito, Haseru Chen,
Kei Okada, Masayuki Inaba
Johou Systems Kougaku Laboratory
The University of Tokyo
{saito, chen, k-okada, inaba}@jsk.t.u-tokyo.ac.jp

Lars Kunze, Michael Beetz
Intelligent Autonomous Systems Group
Technische Universität München
{kunzel, beetz}@cs.tum.edu

***Abstract*— Many of today's mobile robots are supposed to perform everyday manipulation tasks autonomously. However, in large-scale environments, a task-related object might be out of the robot's reach, that is, the object is currently not perceivable by the robot. Hence, the robot first has to search for the object in its environment before it can perform the task.**

In this paper, we present an approach for object search in large-scale environments using different search strategies based on semantic environment models. We demonstrate the feasibility of our approach by integrating it into a robot system and by conducting experiments where the robot is supposed to search for objects within the context of fetch-and-delivery tasks within a multi-level building.

I. Introduction

Autonomous mobile robots that are to perform everyday manipulation tasks need the appropriate capabilities to obtain task-related objects from the environment. That is, object search is a prerequisite to autonomous object manipulation.

When we look at the performance of humans in object search tasks, it seems, that in most cases humans can find objects in their environment relatively effortless and at very high success rates. This is not only because humans have excellent perception capabilities, but also because humans can rely on a large body of commonsense knowledge to conduct the search for objects more effectively.

Although perception plays a key role in object search within robotics, it is also very important to employ semantic knowledge to narrow down the search space, especially in large-scale environments. Pruning the search space is mainly important because of two reasons, first, robot perception is computational expensive, and second, if robots have no clue about potential object locations, objects will only be found by employing exhaustive search methods or by chance.

In the present work, we investigate the problem of how robots can use semantic environment models to perform object search tasks in large-scale environments more effective and efficient. The contributions of this work are as follows. First, we extend the representations and reasoning methods for semantic maps that have been introduced in [1] in order to account for large-scale indoor environments, i.e. multi-level buildings. Second, we utilize commonsense knowledge acquired from Internet users to bootstrap probabilistic models about typical locations of objects which can be updated during the robot's lifetime according to its observations. Third, we present several object search strategies using the above models while also considering the current context of the robot. Finally, we integrate the developed methods within a robot system and provide experimental results for object search in fetch-and-delivery tasks within a multi-level building. Additionally, we show how the semantic search for objects is integrated into an iPad user interface.

In the remainder of the paper we first describe the fetch-and-delivery scenario in more detail in Section II. Then, we explain how we represent the semantic environment models in Section III, and how we use them within various search strategies in Section IV. The integration of these methods with a robot system is presented in Section V. Experimental results are provided in SectionVI. Finally, we put the paper into the context of related work in Section VII, before we conclude in Section VIII.

II. The Fetch-and-Delivery Scenario

In fetch-and-delivery tasks robots are, for example, supposed to serve food and drinks, deliver letters, or collect used cups from workplaces and take them to the kitchen. An essential sub-task of such tasks is to search for the task-related objects in the environment. In this paper, we investigate two scenarios of object search in the context of fetch-and-delivery tasks. In the first scenario the robot is supposed to find cups in its environment and bring them back to its starting position. In the second scenario, we ask the robot to get us a sandwich.

Let's first look at the cup scenario in more detail. For example, consider a situation where a robot is supposed to fetch a particular cup, let's say Michael's cup. In order to find the cup in its environment the robot first replaces the possessive attribute with one or more perceptual attributes, e.g., *Owns(Michael, Cup)* is replaced by *HasLogo(Cup, PR2)*. Either the robot knows about the perceptual attributes of the cup because of its own perceptual experience or this information has to be provided somehow externally. Then the robot tries to retrieve the set of known cup instances from its belief state that fulfill the partial description, e.g. *HasLogo(Cup, PR2)*, or at least, do not contradict it. However, if the robot does not know about any cup instance in the environment, it has to rely on more general knowledge about cups. For example, the robot could infer that cups are typically stored in the cupboards of a kitchen and that they are occasionally located in a dishwasher. Furthermore, if we assume that similar objects are placed next to each other, the robot could reason that cups are similar to glasses

and that therefore the robot could try to find the cup nearby instances of glasses it knows about. In the next step, the robot retrieves the poses of the potential cup locations as well as the poses from which the cups might be perceivable from the semantic map. The latter poses are used as goal locations for the autonomous navigation of the robot. In order to find the cup, the robot moves to the respective goal locations while taking path costs (or the expected rate of success) into account. Having reached a potential cup position the robot uses perception routines for detecting and localizing the cup in the environment. If a cup is detected and fulfills the partial object descriptions the robot applies more specialized perception routines in order to gather all relevant information needed for effectively manipulating the cup. However, if no cup was found, the robot will repeat the procedure until it has searched the remaining potential cup locations.

In the second scenario, the robot is supposed to find and deliver a sandwich. Obviously, similar search strategies like explained above can also be employed. For example, the robot can infer that sandwiches are perishable and that therefore they are typically stored in a fridge.

However, with this example we want to point to the problem that an object might even not exist at the time when looking for it. Some objects in our daily life will only come into existence if we trigger the right processes. For example, food items like sandwiches are created in meal preparation tasks. Mostly these tasks or processes which create instances of certain object types take place at specific locations, e.g., meal preparation tasks are typically carried out in a kitchen or a restaurant. Having this kind knowledge, the robot can go to the designated places and trigger the appropriate processes. Our point here is, that knowledge about the environment is not only beneficial for limiting the search space but in some cases it is even a necessary condition to locate objects.

III. Semantic Environments Models

In this section, we explain the underlying representations and the knowledge that is represented in the semantic environment models. Figure 1 gives an overview of the different ontological concepts and relations, whereby we distinguish mainly between three types of relations: assertional, computable, and probabilistic. In the following we first explain the ontological representation and afterwards we elaborate on the acquisition and formalization of a probabilistic model for commonsense reasoning about object locations.

A. Semantic Maps of Large-scale Environments

In this work, we basically build on the representation formalisms as described in [1]. The underlying representation is based the Web Ontology Language (OWL). In OWL, classes are defined within a taxonomy and derive all the properties of their super-classes. Relations between classes are described by properties. Objects are represented as instances of classes and can also have relations with other instances. The upper ontology is derived from OpenCyc[1].

With respect to semantic environment maps, this formalism allow us to structure the knowledge about classes hierarchically, e.g. a *ProfessorOffice* is an *Office* which is a *RoomInAConstruction*. Given the mechanism of (multiple-) inheritance a class derives all the properties of its super-class(es), e.g., *ProfessorOffice* have also the property *roomNumber* since it has been defined for *RoomInAConstruction*. Objects in the map are described by instances and have a certain *type*, e.g. *Office*, properties to other instances, e.g. *workplaceOf*, and simple data properties like *widthOfObject*. The spatial extensions of an object are described by their bounding box, i.e. depth, width, and height. Although this representation is very simple, it allows us to reason about several spatial relations between objects like *in-Container* or *on-Physical* more efficient. Furthermore, objects are related to instances of perception events. These events contain information about the object's pose at a point in time. Thereby we can track the pose of an object over time and answer questions like where was an object five minutes ago.

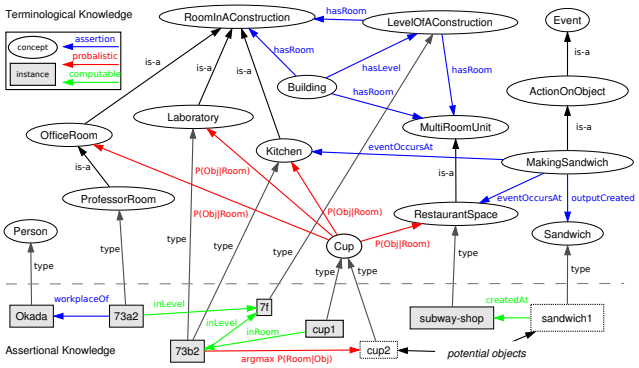


Fig. 1. Simplified overview of concepts and relations of the semantic model for large-scale environments.

In [1], the map representation is mainly focused on small environments like rooms, especially kitchens. However, in the context of fetch-and-delivery tasks it is important to represent also the knowledge about environments at a larger scale. Therefore we introduced concepts like *Building*, *LevelOfAConstruction*, *RoomInAConstruction*, and *Elevator*. Figure 1 show a small excerpt of the extended ontology. Additionally, we introduced properties like *floorNumber*, *roomNumber*, *inLevel*, *inRoom*, and *workplaceOf*. Whereas properties like *roomNumber* are directly asserted to particular instances (assertional property), other properties like *inRoom* are computed based on the actual physical configuration of the environment (computable property), i.e. the current pose and dimensions of an object are taken into account. Thereby we can infer whether spatial relations between objects like *in* and *on* hold at some point in time.

In this work, we created a semantic environment map of the Engineering Building No. 2 at the Hongo Campus of The University of Tokyo. Figure 2 visualize some aspects of the semantic map including levels, rooms, furniture and places.

B. Commonsense Reasoning about Objects Locations

In this section, we explain how we bootstrap a probabilistic model for reasoning about object locations. For example, if

[1]http://www.opencyc.org/

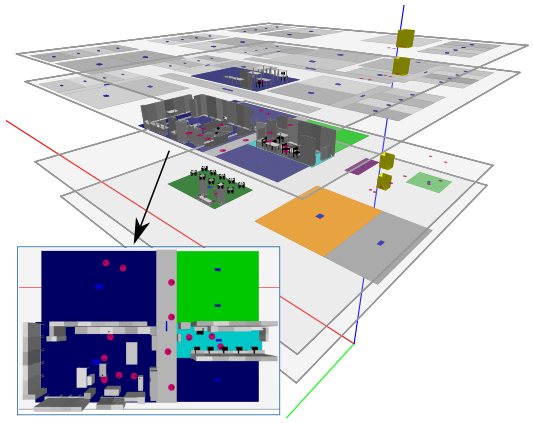


Fig. 2. Engineering Building No. 2, Hongo Campus, University of Tokyo. The map comprises several floors, elevators (yellow boxes), rooms with different types, furniture, and a subway restaurant.

the robot has no information about instances of a certain object type, it should look for these instances at locations where the probability to find this type of object is maximal.

In [2], we investigated how commonsense knowledge that was acquired from Internet users within the Open Mind Indoor Common Sense (OMICS) project [3] can be transformed from natural language to formal representations and integrated into a robot's knowledge base.

The *locations* relation in the OMICS database describes objects and their typical locations, i.e. rooms. Following our previous approach, we first transform the natural language database entries to ontological concepts. For example, the tuple *(mug,kitchen)* is mapped to well-defined ontological concepts *(Cup,Kitchen)*. Then, we calculate the conditional probability of an object given the room by counting the database entries as suggested by [3], i.e.:

$$P(x_i|\omega) = (C(x_i, \omega) + \lambda)/(C(\omega) + \lambda n)$$

where x_i denotes an object, ω denotes a room, and λ denotes the parameter according to Lidstone's law. The λ parameter basically influences how the probability distribution account for unseen tuples. In our experiments, we set $\lambda = 0.5$ (Jeffrey-Perk's law).

In total *locations* database table has more than 3500 entries. However, since we could not map all entries to ontological concepts and we restricted the set of rooms concepts to nine different types that fit our map, we used only 448 entries for calculating the probabilistic properties.

IV. Semantic Object Search

In this section, we first present various search strategies when looking for an object, and second, we explain how the search context influences the robot's behavior.

A. Search Strategies

In the introduction of this paper we already mentioned the excellent ability of humans to find objects, sometimes even in previously unknown environments. Among other reasons, this is because humans make use of different strategies when looking for an object. Therefore, robots should also be equipped with a repertoire of strategies they can employ, depending on what knowledge they have about the environment. For example, a robot that has knowledge about some instances of a sought object class should exploit this information before considering more general knowledge. However, if a robot does not have any information about object instances, it has to rely on common sense. How these different kinds of search strategies should be orchestrated is another interesting problem which beyond the scope of this paper. Hence, in this work we assume that different searches are conducted in a kind of *try-in-order* procedure.

In general, we identified three different situations in which a robot can be when looking for an object:

1) the object is currently not perceivable by the robot
2) the object is perceivable by the robot
3) the object is physically attached to the robot

The search for an object can basically be started in all three of these situations. If the robot already holds the sought object in its hand, it can immediately finish the search successfully. If the robot already perceives the sought object within the environment, it has to localize the object effectively for its manipulation routines and pick it up. If the robot do not perceive the object at all, the robot has to reason about the potential locations of the object, go to the locations, try to perceive the object, and if it can perceive the object try to pick it up, otherwise, the robot has to continue at the next potential location. If the robot is not able to find the object at any potential location, the robot could either switch to another search strategy, ask for help, or eventually give up the search.

In the following we explain some basic predicates that have been implemented in PROLOG in order to search for objects with different strategies using the semantic environment models described in the previous section.

locatedAt(Obj, Pose) denotes the pose of an object. Poses are described by the translation and rotation of an object decoded in a 4×4 transformation matrix.

lookForAt(Obj, Pose) denotes the pose where an object instance might be perceivable.

hasType(Obj, Type) denotes the type of an object. When only providing a type all object instances with the specified type are mentally retrieved from the map.

similarTypes(Type, SimilarTypes) denotes the semantic relatedness between an object type and other types in the ontology. The relatedness is calculated based on the *wup* similarity measure as explained in [1]. We see basically see two possibilities for using the similarity between objects in the context of search: (1) objects that are similar are often placed together, i.e. a robot can look at places of similar objects, if it has no information about the object itself, and (2), if an object cannot be found in the environment, the robot could look for a similar object instead, e.g. if the robot cannot find any cup, it could look for a glass.

Since eventually we want to localize objects of the similar types in the environment map, only types of map

instances are considered for the calculation. Finally, the similar types are sorted with respect to the *wup* measure.

similarObj(Type, Obj) retrieves an object instance from the map that is similar to a certain type. The predicate considers only instances in the environment map. Eventually the predicate returns all instances described in the map, for retrieving only a subset we have implemented more specialized predicates (see below).

mostSimilarObj(Type, Obj) retrieves only the object instances of the most similar object type from the map.

kMostSimilarObj(K, Type, Obj) retrieves the object instances of the k-most similar object types from the map.

createdAt(Type, Loc) denotes a location where instances of a given type will typically be created, e.g. a *Sandwich* will typically be created in a *Kitchen* or a *Restaurant*. Within the semantic environment map these locations are related to events using the *eventOccursAtLocation(Event, Loc)* property and if these events have a property like *outputCreated(Event, Type)* then the predicate *createdAt(Type, Loc)* holds.

locationOf(Loc, Type, P) denotes the probability P to encounter a given object type at a location. The probability is calculated from the probabilistic model explained in Section III-B using Bayes' rule:

$$P(\omega|x) = \frac{P(x|\omega)P(\omega)}{\sum_i P(x|\omega_i)P(\omega_i)}$$

To retrieve the location with the highest probability we simply apply the *argmax* operator $\underset{\omega \in \Omega}{\operatorname{argmax}}\ P(\omega|x)$.

Given these basic predicate definitions it is already possible to implement different kinds of search strategies when looking for an object. The most noticeable difference is that some predicates use knowledge about known instances in the robots environment whereas others only consider general knowledge about classes.

B. Search Context

This section describes how the search results are affected by the situational context of the robot. In this paper, the context is determined by the robot's current pose.

In the previous section, we explained some basic predicates a robot can use to retrieve objects and locations from the semantic environment map. However, the decision to which location the robot will move to and look for an object depends also on its current position. That is, instead of retrieving only instance-by-instance from the map and move to the different locations successively, the robot rather retrieves the set of all instances at once and takes the respective path costs into account. More generally, these costs should be include the probability of success to find an object at a location. Though, in this work we only consider a rough heuristic to calculate the path cost from the current position of the robot to a goal location. For the heuristic we distinguish two situations:

1) robot pose and goal pose are in the same level
2) robot pose and goal pose are in different levels

If the robot pose and the goal pose are in the same level, the path cost is determined simply by the Euclidean distance between the two poses. Obviously, the path cost calculation can also be approximated by using path planner on 2D occupancy grid maps.

If the robot pose and the goal pose are not in the same level, the overall path cost is determined by the sum of three cost calculations: (1) the path cost from the current position to the elevator in the same level, (2) the cost using the elevator, and (3) the path cost from the elevator to the goal pose. The costs (1) and (3) are calculated as explained above. The elevator cost is determined by the distance of levels. However, the cost model for using an elevator could be replaced by a more complex model considering for example the waiting time, intermediate stops, and the hour of the day. Figure 3 visualizes the path costs from room *73a4* in floor 7 to other rooms in the building.

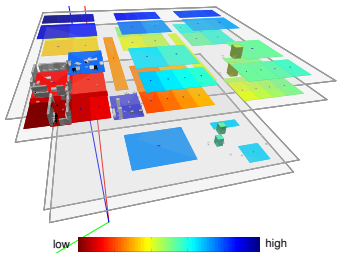


Fig. 3. Path costs visualized as heatmap calculated from room *73a4*, 7th floor (dark red).

V. The Robot System

Within the experiments, we use the PR2[2] robot platform and a ROS[3]-based software infrastructure. An overview of the different software components is shown in Figure 4.

Basically, the robot's main control program is responsible for performing the fetch-and-delivery tasks. It receives a task goal from an iPad interface and performs the task autonomously. Alternatively, the task can be carried out in interaction with a user. After having received the task goal, i.e. an object to search for, one of the search strategies is used to query the semantic environment models for potential object locations. When the task-level control program recognizes that the robot has to change the floor, the relevant information is retrieved from the semantic environment model, e.g. the number of the current and the target floor, and the position of elevator control panels. These information are then send to the inter-floor navigation module to navigate the robot to the respective locations. At the goal location the robot tries to detect the object under request by using a sift-based template matching approach. Finally, the robot uses motion planning to figure out how to grasp the object.

VI. Preliminary Experimental Results

In this section, we present the preliminary results of two example scenarios to demonstrate how a robot can use the semantic environment models in fetch-and-delivery tasks.

A. Cups

In the first scenario, we look at situations in which a robot is supposed to find cups in the environment.

[2] http://www.willowgarage.com/pages/pr2/overview
[3] http://www.ros.org/

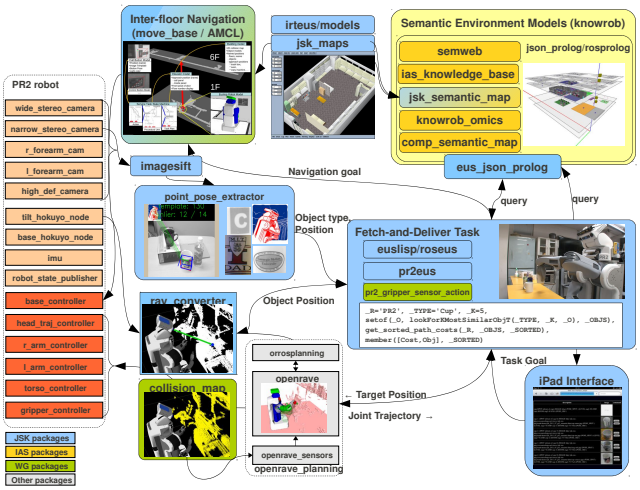


Fig. 4. Overview of the main software components realizing the execution of fetch-and-delivery tasks.

In the first situation the robot is asked to look for a particular cup, a cup that has a certain logo on it. Let's assume the robot knows some cup instances in the environment, and it also has a model of the logo. However, this logo is not associated to any of the cups, i.e., the following query fails:

```
?- hasType(C,'Cup'), hasLogo(C, 'CMU').
```

Hence, the robot has to move to all possible cup locations and try to to match the logo with one of the cups perceptually. The following PROLOG query shows how the robot can retrieve the positions of potential cup locations from the semantic map and calculate their respective path costs. The list of ordered poses is than passed to the navigation framework which uses the *lookForAt(Obj,Pose)* predicate to determine the navigation goals.

```
?- findall(Pose, (hasType(Obj, 'Cup'),
                  not(hasLogo(Obj, AnyLogo)),
                  locatedAt(Obj, Pose)), PList),
   calc_path_costs('current-pose', PList, SortedPoses).
```

Figure 5 visualizes the query result in the semantic map and show some images of the carried out robot experiment. After detecting a cup with a different logo in the first room, the robot navigates to another room where it eventually find the sought cup[4]. We varied the experiment by placing the individual cups at the different locations in various permutations or removing individual cups completely.

In addition to the autonomous object search, we also developed an interactive mode where users can search for objects using an iPad interface. Basically, users can ask for an object and receive a list of possible object locations. After taking a user's request, the system searches the semantic environment models for possible locations

Fig. 6. iPad interface.

[4]Video: http://www.jsk.t.u-tokyo.ac.jp/~kunzel/sos.html

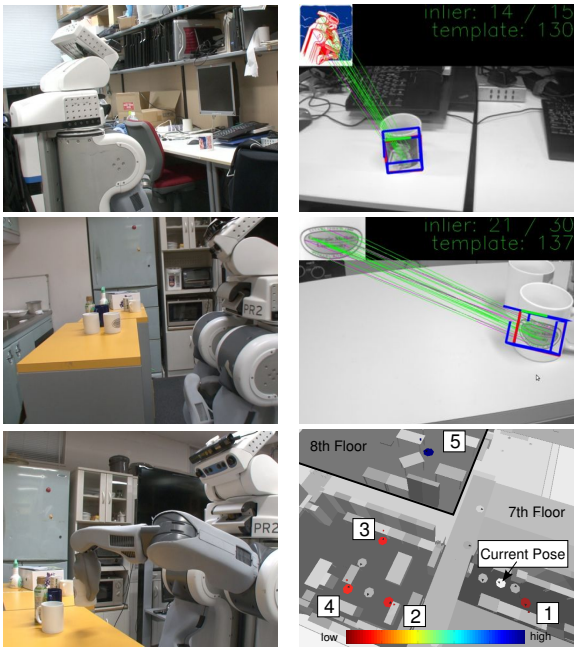


Fig. 5. Row 1: Robot at place (1). Found cup, but has PR2 logo. Row 2: Robot at place (2). Found cup with desired CMU logo. Row 3: Robot picks up cup. Visualized query results in semantic map.

using a selected search strategy, generates a list of potential locations including information about floor, room, costs, and if available an image of the object's visual appearance. The user can then decide from which location the robot should fetch the object. Figure 6 shows a query result for *Cup*.

In the previous situation we assumed that the robot had some knowledge about cup instances. However, if the robot has no such information it has to rely on more general knowledge. The following query illustrates how the robot retrieves potential locations from the probabilistic models.

```
?- findall([P, Type],
           locationOf(Type ,'Cup' , P),
           PTList),
   argmax(PTList, RoomType),
   findall(R, hasType(R, RoomType), Rs),
   member(Room, Rs),
   inRoom(Spot, Room), hasType(Spot,'Place').
```

First, the robot finds all tuples of probability (*P*) and room type (*Type*), given that it is looking for a *Cup*. Figure 7 shows some probabilities that a type of object can be found in a room. Second, it extracts only the tuple with the highest probability, i.e. *Kitchen* (given *Cup*). Third, it retrieves all room instances of this *Type* from the semantic map, and finally, it considers all known places in these room instances. At each place the robot tries to perceive a cup at different angles.

Using the above query the robot gets the information that a *Cup* instance might be found in a *Kitchen*. So, it has to search for a cup at all places in the kitchen. However, not all places in the kitchen make sense to look for a cup, e.g. one place is in front of a TV. Using additional knowledge, for example, that the places should be in front of a table, shelf, or cupboard, can further reduce the number of places.

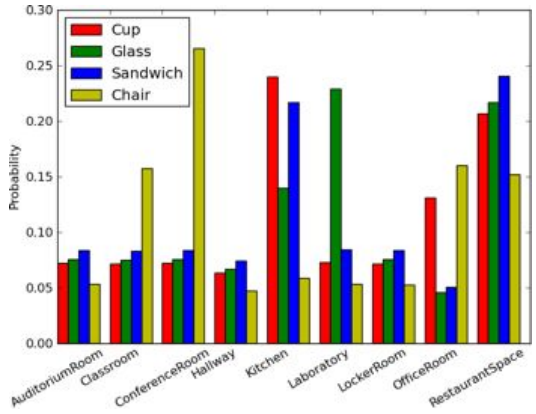


Fig. 7. Examples of probabilities to find a given object in certain room, i.e. $P(\omega|x)$. The initial configuration is bootstrapped from the OMICS database.

B. Sandwiches

In this scenario the robot is asked to get a sandwich. Let's assume the robot does not has any knowledge about a *Sandwich* instance. So, it calculates which objects (it knows about) are similar to sandwiches using the following query:

```
?- mostSimilarObj('Sandwich', Obj).
```

where *Obj* is bound to *sushi1*. This instance is of type *Sushi* and it is similar to *Sandwich* because of the common super-class *Food-ReadyToEat*. With its spatial reasoning capabilities the robot finds out that *sushi1* is currently located in *frigde1* in room *73b2*.

Another possibility to find a sandwich is to search at locations where instances of sandwiches are created, e.g. kitchens or restaurants. Using the query

```
?- createdAt('Sandwich', Loc).
```

the robots infers that sandwiches might be are created in room *73b2* which is of type *Kitchen* and/or *subway-shop* which is of type *FastFoodRestaurant*.

Fig. 8. PR2 using an elevator, approaching subway shop, and ordering a sandwich.

VII. Related Work

In recent years, the usage of semantic information has become more and more prominent in the context of robotics.

In [4], object-room relations are exploited for mapping and navigation tasks. Such kind of knowledge is often represented by the means of Description logics [5], [6], or probabilistic models [7], [8].

How to use semantic knowledge within object search tasks is explored by [9], [10], [11]. Our approach can be considered in a similar line of research. Probabilistic models are used in [9] to guide a simulated robot during object search tasks in structured indoor environments, namely supermarkets. Work by [10] utilizes also probabilistic methods on a robot system to make inferences about the spatial relations between object instances in the environment. Similar to our approach, [11] bootstraps commonsense knowledge from the OMICS database to initialize the probabilistic representations. However, other forms of reasoning are not investigated in their robot experiments.

VIII. Conclusion and Future Work

In this paper, we investigated how semantic information about the robot's environment can be used in object search tasks. To this extent, we extended the representation formalisms introduced in previous work ([1]) and implement several PROLOG programs to infer the potential locations of an object. As proof-of-concept, we integrated the developed methods into a robot system that searches objects within a multi-level building in the context of fetch-and-delivery tasks. The realized system is able to navigate to inferred objects location using different kinds of semantic information.

In future work, we will include further probabilistic models about spatial relations like *in*, *on*, and *next-to* in order to restrict the search space of objects and thereby make the search tasks even more efficient. Additionally, we will conduct more experiments to evaluate the outcomes of search tasks more systematically.

References

[1] M. Tenorth, L. Kunze, D. Jain, and M. Beetz, "KNOWROB-MAP – Knowledge-Linked Semantic Object Maps," in *Proceedings of 2010 IEEE-RAS International Conference on Humanoid Robots*, Nashville, TN, USA, December 6-8 2010.

[2] L. Kunze, M. Tenorth, and M. Beetz, "Putting People's Common Sense into Knowledge Bases of Household Robots," in *33rd Annual German Conference on Artificial Intelligence (KI 2010)*. Karlsruhe, Germany: Springer, September 21-24 2010, pp. 151–159.

[3] R. Gupta and M. J. Kochenderfer, "Common sense data acquisition for indoor mobile robots," in *Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, 2004, pp. 605–610.

[4] C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, J. Fernández-Madrigal, and J. González, "Multi-hierarchical semantic maps for mobile robotics," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Edmonton, CA, 2005, pp. 3492–3497.

[5] H. Zender, O. Martínez Mozos, P. Jensfelt, G. Kruijff, and W. Burgard, "Conceptual spatial representations for indoor mobile robots," *Robotics and Autonomous Systems*, 2008.

[6] K. Sjöö, H. Zender, P. Jensfelt, G.-J. M. Kruijff, A. Pronobis, N. Hawes, and M. Brenner, "The explorer system," in *Cognitive Systems*, H. I. Christensen, G.-J. M. Kruijff, and J. L. Wyatt, Eds. Springer, 2010.

[7] A. Bouguerra, L. Karlsson, and A. Saffiotti, "Handling uncertainty in semantic-knowledge based execution monitoring," in *IROS*. IEEE, 2007, pp. 437–443.

[8] S. Vasudevan, S. Gächter, V. Nguyen, and R. Siegwart, "Cognitive maps for mobile robots - an object based approach," *Robotics and Autonomous Systems*, vol. 55, no. 5, pp. 359–371, 2007.

[9] D. Joho and W. Burgard, "Searching for objects: Combining multiple cues to object locations using a maximum entropy model," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, AK, USA, May 2010, pp. 723–728.

[10] A. Aydemir, K. Sjöö, J. Folkesson, A. Pronobis, and P. Jensfelt, "Search in the real world: Active visual object search based on spatial relations," in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA'11)*, Shanghai, China, May 2011.

[11] M. Hanheide, C. Gretton, and M. Göbelbecker, "Dora, a robot exploiting probabilistic knowledge under uncertain sensing for efficient object search," in *Proceedings of Systems Demonstration of the 21st International Conference on Automated Planning and Scheduling (ICAPS)*, Freiburg, Germany, June 2011.

Semantic Object Search in Large-scale Indoor Environments

Manabu Saito, Haseru Chen,
Kei Okada, Masayuki Inaba
Johou Systems Kougaku Laboratory
The University of Tokyo
{saito, chen, k-okada, inaba}@jsk.t.u-tokyo.ac.jp

Lars Kunze, Michael Beetz
Intelligent Autonomous Systems Group
Technische Universität München
{kunzel, beetz}@cs.tum.edu

***Abstract*— Many of today's mobile robots are supposed to perform everyday manipulation tasks autonomously. However, in large-scale environments, a task-related object might be out of the robot's reach, that is, the object is currently not perceivable by the robot. Hence, the robot first has to search for the object in its environment before it can perform the task.**

In this paper, we present an approach for object search in large-scale environments using different search strategies based on semantic environment models. We demonstrate the feasibility of our approach by integrating it into a robot system and by conducting experiments where the robot is supposed to search for objects within the context of fetch-and-delivery tasks within a multi-level building.

I. Introduction

Autonomous mobile robots that are to perform everyday manipulation tasks need the appropriate capabilities to obtain task-related objects from the environment. That is, object search is a prerequisite to autonomous object manipulation.

When we look at the performance of humans in object search tasks, it seems, that in most cases humans can find objects in their environment relatively effortless and at very high success rates. This is not only because humans have excellent perception capabilities, but also because humans can rely on a large body of commonsense knowledge to conduct the search for objects more effectively.

Although perception plays a key role in object search within robotics, it is also very important to employ semantic knowledge to narrow down the search space, especially in large-scale environments. Pruning the search space is mainly important because of two reasons, first, robot perception is computational expensive, and second, if robots have no clue about potential object locations, objects will only be found by employing exhaustive search methods or by chance.

In the present work, we investigate the problem of how robots can use semantic environment models to perform object search tasks in large-scale environments more effective and efficient. The contributions of this work are as follows. First, we extend the representations and reasoning methods for semantic maps that have been introduced in [1] in order to account for large-scale indoor environments, i.e. multi-level buildings. Second, we utilize commonsense knowledge acquired from Internet users to bootstrap probabilistic models about typical locations of objects which can be updated during the robot's lifetime according to its observations. Third, we present several object search strategies using the above models while also considering the current context of the robot. Finally, we integrate the developed methods within a robot system and provide experimental results for object search in fetch-and-delivery tasks within a multi-level building. Additionally, we show how the semantic search for objects is integrated into an iPad user interface.

In the remainder of the paper we first describe the fetch-and-delivery scenario in more detail in Section II. Then, we explain how we represent the semantic environment models in Section III, and how we use them within various search strategies in Section IV. The integration of these methods with a robot system is presented in Section V. Experimental results are provided in SectionVI. Finally, we put the paper into the context of related work in Section VII, before we conclude in Section VIII.

II. The Fetch-and-Delivery Scenario

In fetch-and-delivery tasks robots are, for example, supposed to serve food and drinks, deliver letters, or collect used cups from workplaces and take them to the kitchen. An essential sub-task of such tasks is to search for the task-related objects in the environment. In this paper, we investigate two scenarios of object search in the context of fetch-and-delivery tasks. In the first scenario the robot is supposed to find cups in its environment and bring them back to its starting position. In the second scenario, we ask the robot to get us a sandwich.

Let's first look at the cup scenario in more detail. For example, consider a situation where a robot is supposed to fetch a particular cup, let's say Michael's cup. In order to find the cup in its environment the robot first replaces the possessive attribute with one or more perceptual attributes, e.g., *Owns(Michael, Cup)* is replaced by *HasLogo(Cup, PR2)*. Either the robot knows about the perceptual attributes of the cup because of its own perceptual experience or this information has to be provided somehow externally. Then the robot tries to retrieve the set of known cup instances from its belief state that fulfill the partial description, e.g. *HasLogo(Cup, PR2)*, or at least, do not contradict it. However, if the robot does not know about any cup instance in the environment, it has to rely on more general knowledge about cups. For example, the robot could infer that cups are typically stored in the cupboards of a kitchen and that they are occasionally located in a dishwasher. Furthermore, if we assume that similar objects are placed next to each other, the robot could reason that cups are similar to glasses

and that therefore the robot could try to find the cup nearby instances of glasses it knows about. In the next step, the robot retrieves the poses of the potential cup locations as well as the poses from which the cups might be perceivable from the semantic map. The latter poses are used as goal locations for the autonomous navigation of the robot. In order to find the cup, the robot moves to the respective goal locations while taking path costs (or the expected rate of success) into account. Having reached a potential cup position the robot uses perception routines for detecting and localizing the cup in the environment. If a cup is detected and fulfills the partial object descriptions the robot applies more specialized perception routines in order to gather all relevant information needed for effectively manipulating the cup. However, if no cup was found, the robot will repeat the procedure until it has searched the remaining potential cup locations.

In the second scenario, the robot is supposed to find and deliver a sandwich. Obviously, similar search strategies like explained above can also be employed. For example, the robot can infer that sandwiches are perishable and that therefore they are typically stored in a fridge.

However, with this example we want to point to the problem that an object might even not exist at the time when looking for it. Some objects in our daily life will only come into existence if we trigger the right processes. For example, food items like sandwiches are created in meal preparation tasks. Mostly these tasks or processes which create instances of certain object types take place at specific locations, e.g., meal preparation tasks are typically carried out in a kitchen or a restaurant. Having this kind knowledge, the robot can go to the designated places and trigger the appropriate processes. Our point here is, that knowledge about the environment is not only beneficial for limiting the search space but in some cases it is even a necessary condition to locate objects.

III. Semantic Environments Models

In this section, we explain the underlying representations and the knowledge that is represented in the semantic environment models. Figure 1 gives an overview of the different ontological concepts and relations, whereby we distinguish mainly between three types of relations: assertional, computable, and probabilistic. In the following we first explain the ontological representation and afterwards we elaborate on the acquisition and formalization of a probabilistic model for commonsense reasoning about object locations.

A. Semantic Maps of Large-scale Environments

In this work, we basically build on the representation formalisms as described in [1]. The underlying representation is based the Web Ontology Language (OWL). In OWL, classes are defined within a taxonomy and derive all the properties of their super-classes. Relations between classes are described by properties. Objects are represented as instances of classes and can also have relations with other instances. The upper ontology is derived from OpenCyc[1].

With respect to semantic environment maps, this formalism allow us to structure the knowledge about classes hierarchically, e.g. a *ProfessorOffice* is an *Office* which is a *RoomInAConstruction*. Given the mechanism of (multiple-) inheritance a class derives all the properties of its super-class(es), e.g., *ProfessorOffice* have also the property *roomNumber* since it has been defined for *RoomInAConstruction*. Objects in the map are described by instances and have a certain *type*, e.g. *Office*, properties to other instances, e.g. *workplaceOf*, and simple data properties like *widthOfObject*. The spatial extensions of an object are described by their bounding box, i.e. depth, width, and height. Although this representation is very simple, it allows us to reason about several spatial relations between objects like *in-Container* or *on-Physical* more efficient. Furthermore, objects are related to instances of perception events. These events contain information about the object's pose at a point in time. Thereby we can track the pose of an object over time and answer questions like where was an object five minutes ago.

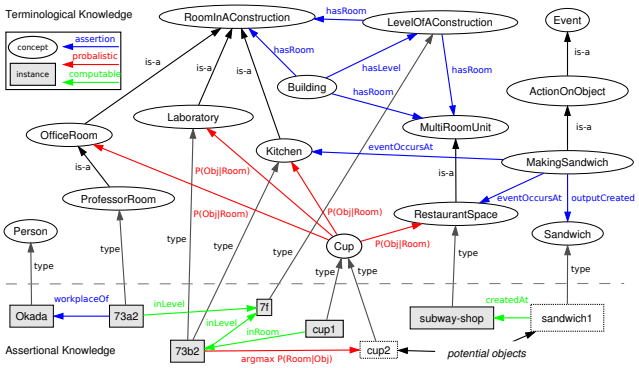


Fig. 1. Simplified overview of concepts and relations of the semantic model for large-scale environments.

In [1], the map representation is mainly focused on small environments like rooms, especially kitchens. However, in the context of fetch-and-delivery tasks it is important to represent also the knowledge about environments at a larger scale. Therefore we introduced concepts like *Building*, *LevelOfAConstruction*, *RoomInAConstruction*, and *Elevator*. Figure 1 show a small excerpt of the extended ontology. Additionally, we introduced properties like *floorNumber*, *roomNumber*, *inLevel*, *inRoom*, and *workplaceOf*. Whereas properties like *roomNumber* are directly asserted to particular instances (assertional property), other properties like *inRoom* are computed based on the actual physical configuration of the environment (computable property), i.e. the current pose and dimensions of an object are taken into account. Thereby we can infer whether spatial relations between objects like *in* and *on* hold at some point in time.

In this work, we created a semantic environment map of the Engineering Building No. 2 at the Hongo Campus of The University of Tokyo. Figure 2 visualize some aspects of the semantic map including levels, rooms, furniture and places.

B. Commonsense Reasoning about Objects Locations

In this section, we explain how we bootstrap a probabilistic model for reasoning about object locations. For example, if

[1]http://www.opencyc.org/

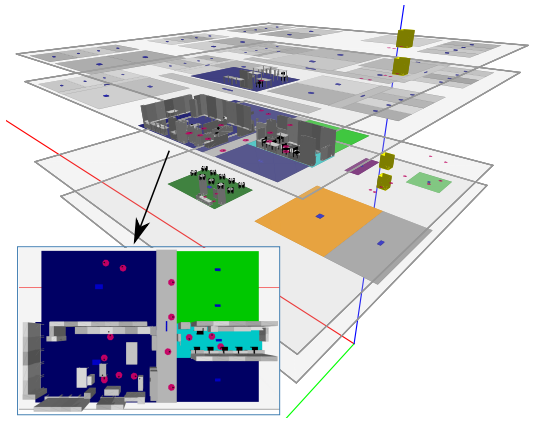


Fig. 2. Engineering Building No. 2, Hongo Campus, University of Tokyo. The map comprises several floors, elevators (yellow boxes), rooms with different types, furniture, and a subway restaurant.

the robot has no information about instances of a certain object type, it should look for these instances at locations where the probability to find this type of object is maximal.

In [2], we investigated how commonsense knowledge that was acquired from Internet users within the Open Mind Indoor Common Sense (OMICS) project [3] can be transformed from natural language to formal representations and integrated into a robot's knowledge base.

The *locations* relation in the OMICS database describes objects and their typical locations, i.e. rooms. Following our previous approach, we first transform the natural language database entries to ontological concepts. For example, the tuple *(mug,kitchen)* is mapped to well-defined ontological concepts *(Cup,Kitchen)*. Then, we calculate the conditional probability of an object given the room by counting the database entries as suggested by [3], i.e.:

$$P(x_i|\omega) = (C(x_i, \omega) + \lambda)/(C(\omega) + \lambda n)$$

where x_i denotes an object, ω denotes a room, and λ denotes the parameter according to Lidstone's law. The λ parameter basically influences how the probability distribution account for unseen tuples. In our experiments, we set $\lambda = 0.5$ (Jeffrey-Perk's law).

In total *locations* database table has more than 3500 entries. However, since we could not map all entries to ontological concepts and we restricted the set of rooms concepts to nine different types that fit our map, we used only 448 entries for calculating the probabilistic properties.

IV. Semantic Object Search

In this section, we first present various search strategies when looking for an object, and second, we explain how the search context influences the robot's behavior.

A. Search Strategies

In the introduction of this paper we already mentioned the excellent ability of humans to find objects, sometimes even in previously unknown environments. Among other reasons, this is because humans make use of different strategies when looking for an object. Therefore, robots should also be equipped with a repertoire of strategies they can employ, depending on what knowledge they have about the environment. For example, a robot that has knowledge about some instances of a sought object class should exploit this information before considering more general knowledge. However, if a robot does not have any information about object instances, it has to rely on common sense. How these different kinds of search strategies should be orchestrated is another interesting problem which beyond the scope of this paper. Hence, in this work we assume that different searches are conducted in a kind of *try-in-order* procedure.

In general, we identified three different situations in which a robot can be when looking for an object:

1) the object is currently not perceivable by the robot
2) the object is perceivable by the robot
3) the object is physically attached to the robot

The search for an object can basically be started in all three of these situations. If the robot already holds the sought object in its hand, it can immediately finish the search successfully. If the robot already perceives the sought object within the environment, it has to localize the object effectively for its manipulation routines and pick it up. If the robot do not perceive the object at all, the robot has to reason about the potential locations of the object, go to the locations, try to perceive the object, and if it can perceive the object try to pick it up, otherwise, the robot has to continue at the next potential location. If the robot is not able to find the object at any potential location, the robot could either switch to another search strategy, ask for help, or eventually give up the search.

In the following we explain some basic predicates that have been implemented in PROLOG in order to search for objects with different strategies using the semantic environment models described in the previous section.

locatedAt(Obj, Pose) denotes the pose of an object. Poses are described by the translation and rotation of an object decoded in a 4×4 transformation matrix.

lookForAt(Obj, Pose) denotes the pose where an object instance might be perceivable.

hasType(Obj, Type) denotes the type of an object. When only providing a type all object instances with the specified type are mentally retrieved from the map.

similarTypes(Type, SimilarTypes) denotes the semantic relatedness between an object type and other types in the ontology. The relatedness is calculated based on the *wup* similarity measure as explained in [1]. We see basically see two possibilities for using the similarity between objects in the context of search: (1) objects that are similar are often placed together, i.e. a robot can look at places of similar objects, if it has no information about the object itself, and (2), if an object cannot be found in the environment, the robot could look for a similar object instead, e.g. if the robot cannot find any cup, it could look for a glass.

Since eventually we want to localize objects of the similar types in the environment map, only types of map

instances are considered for the calculation. Finally, the similar types are sorted with respect to the *wup* measure.

similarObj(Type, Obj) retrieves an object instance from the map that is similar to a certain type. The predicate considers only instances in the environment map. Eventually the predicate returns all instances described in the map, for retrieving only a subset we have implemented more specialized predicates (see below).

mostSimilarObj(Type, Obj) retrieves only the object instances of the most similar object type from the map.

kMostSimilarObj(K, Type, Obj) retrieves the object instances of the k-most similar object types from the map.

createdAt(Type, Loc) denotes a location where instances of a given type will typically be created, e.g. a *Sandwich* will typically be created in a *Kitchen* or a *Restaurant*. Within the semantic environment map these locations are related to events using the *eventOccursAtLocation(Event, Loc)* property and if these events have a property like *outputCreated(Event, Type)* then the predicate *createdAt(Type, Loc)* holds.

locationOf(Loc, Type, P) denotes the probability P to encounter a given object type at a location. The probability is calculated from the probabilistic model explained in Section III-B using Bayes' rule:

$$P(\omega|x) = \frac{P(x|\omega)P(\omega)}{\sum_i P(x|\omega_i)P(\omega_i)}$$

To retrieve the location with the highest probability we simply apply the *argmax* operator $\underset{\omega \in \Omega}{\operatorname{argmax}}\ P(\omega|x)$.

Given these basic predicate definitions it is already possible to implement different kinds of search strategies when looking for an object. The most noticeable difference is that some predicates use knowledge about known instances in the robots environment whereas others only consider general knowledge about classes.

B. Search Context

This section describes how the search results are affected by the situational context of the robot. In this paper, the context is determined by the robot's current pose.

In the previous section, we explained some basic predicates a robot can use to retrieve objects and locations from the semantic environment map. However, the decision to which location the robot will move to and look for an object depends also on its current position. That is, instead of retrieving only instance-by-instance from the map and move to the different locations successively, the robot rather retrieves the set of all instances at once and takes the respective path costs into account. More generally, these costs should be include the probability of success to find an object at a location. Though, in this work we only consider a rough heuristic to calculate the path cost from the current position of the robot to a goal location. For the heuristic we distinguish two situations:

1) robot pose and goal pose are in the same level
2) robot pose and goal pose are in different levels

If the robot pose and the goal pose are in the same level, the path cost is determined simply by the Euclidean distance between the two poses. Obviously, the path cost calculation can also be approximated by using path planner on 2D occupancy grid maps.

If the robot pose and the goal pose are not in the same level, the overall path cost is determined by the sum of three cost calculations: (1) the path cost from the current position to the elevator in the same level, (2) the cost using the elevator, and (3) the path cost from the elevator to the goal pose. The costs (1) and (3) are calculated as explained above. The elevator cost is determined by the distance of levels. However, the cost model for using an elevator could be replaced by a more complex model considering for example the waiting time, intermediate stops, and the hour of the day. Figure 3 visualizes the path costs from room *73a4* in floor 7 to other rooms in the building.

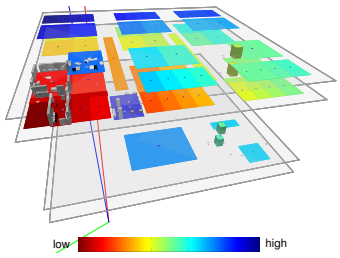


Fig. 3. Path costs visualized as heatmap calculated from room *73a4*, 7th floor (dark red).

V. The Robot System

Within the experiments, we use the PR2[2] robot platform and a ROS[3]-based software infrastructure. An overview of the different software components is shown in Figure 4.

Basically, the robot's main control program is responsible for performing the fetch-and-delivery tasks. It receives a task goal from an iPad interface and performs the task autonomously. Alternatively, the task can be carried out in interaction with a user. After having received the task goal, i.e. an object to search for, one of the search strategies is used to query the semantic environment models for potential object locations. When the task-level control program recognizes that the robot has to change the floor, the relevant information is retrieved from the semantic environment model, e.g. the number of the current and the target floor, and the position of elevator control panels. These information are then send to the inter-floor navigation module to navigate the robot to the respective locations. At the goal location the robot tries to detect the object under request by using a sift-based template matching approach. Finally, the robot uses motion planning to figure out how to grasp the object.

VI. Preliminary Experimental Results

In this section, we present the preliminary results of two example scenarios to demonstrate how a robot can use the semantic environment models in fetch-and-delivery tasks.

A. Cups

In the first scenario, we look at situations in which a robot is supposed to find cups in the environment.

[2] http://www.willowgarage.com/pages/pr2/overview
[3] http://www.ros.org/

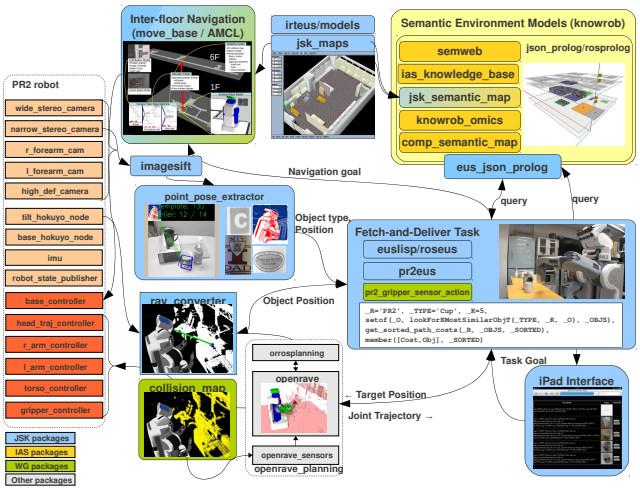


Fig. 4. Overview of the main software components realizing the execution of fetch-and-delivery tasks.

In the first situation the robot is asked to look for a particular cup, a cup that has a certain logo on it. Let's assume the robot knows some cup instances in the environment, and it also has a model of the logo. However, this logo is not associated to any of the cups, i.e., the following query fails:

```
?- hasType(C,'Cup'), hasLogo(C, 'CMU').
```

Hence, the robot has to move to all possible cup locations and try to to match the logo with one of the cups perceptually. The following PROLOG query shows how the robot can retrieve the positions of potential cup locations from the semantic map and calculate their respective path costs. The list of ordered poses is than passed to the navigation framework which uses the *lookForAt(Obj,Pose)* predicate to determine the navigation goals.

```
?- findall(Pose, (hasType(Obj, 'Cup'),
                  not(hasLogo(Obj, AnyLogo)),
                  locatedAt(Obj, Pose)), PList),
   calc_path_costs('current-pose', PList, SortedPoses).
```

Figure 5 visualizes the query result in the semantic map and show some images of the carried out robot experiment. After detecting a cup with a different logo in the first room, the robot navigates to another room where it eventually find the sought cup[4]. We varied the experiment by placing the individual cups at the different locations in various permutations or removing individual cups completely.

In addition to the autonomous object search, we also developed an interactive mode where users can search for objects using an iPad interface. Basically, users can ask for an object and receive a list of possible object locations. After taking a user's request, the system searches the semantic environment models for possible locations

Fig. 6. iPad interface.

[4]Video: http://www.jsk.t.u-tokyo.ac.jp/~kunzel/sos.html

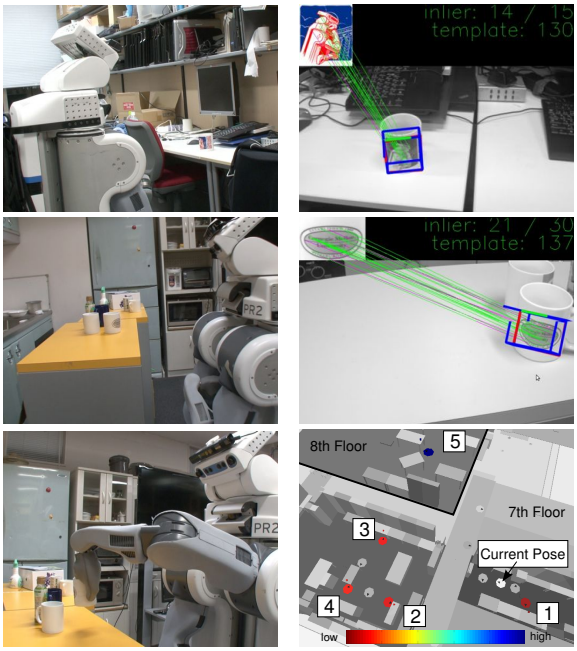


Fig. 5. Row 1: Robot at place (1). Found cup, but has PR2 logo. Row 2: Robot at place (2). Found cup with desired CMU logo. Row 3: Robot picks up cup. Visualized query results in semantic map.

using a selected search strategy, generates a list of potential locations including information about floor, room, costs, and if available an image of the object's visual appearance. The user can then decide from which location the robot should fetch the object. Figure 6 shows a query result for *Cup*.

In the previous situation we assumed that the robot had some knowledge about cup instances. However, if the robot has no such information it has to rely on more general knowledge. The following query illustrates how the robot retrieves potential locations from the probabilistic models.

```
?- findall([P, Type],
           locationOf(Type ,'Cup' , P),
           PTList),
   argmax(PTList, RoomType),
   findall(R, hasType(R, RoomType), Rs),
   member(Room, Rs),
   inRoom(Spot, Room), hasType(Spot,'Place').
```

First, the robot finds all tuples of probability (*P*) and room type (*Type*), given that it is looking for a *Cup*. Figure 7 shows some probabilities that a type of object can be found in a room. Second, it extracts only the tuple with the highest probability, i.e. *Kitchen* (given *Cup*). Third, it retrieves all room instances of this *Type* from the semantic map, and finally, it considers all known places in these room instances. At each place the robot tries to perceive a cup at different angles.

Using the above query the robot gets the information that a *Cup* instance might be found in a *Kitchen*. So, it has to search for a cup at all places in the kitchen. However, not all places in the kitchen make sense to look for a cup, e.g. one place is in front of a TV. Using additional knowledge, for example, that the places should be in front of a table, shelf, or cupboard, can further reduce the number of places.

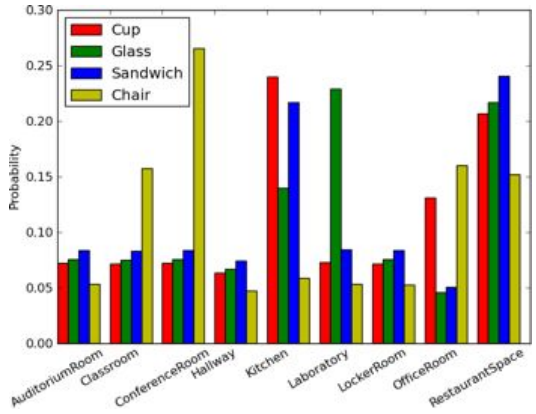


Fig. 7. Examples of probabilities to find a given object in certain room, i.e. $P(\omega|x)$. The initial configuration is bootstrapped from the OMICS database.

B. Sandwiches

In this scenario the robot is asked to get a sandwich. Let's assume the robot does not has any knowledge about a *Sandwich* instance. So, it calculates which objects (it knows about) are similar to sandwiches using the following query:

```
?- mostSimilarObj('Sandwich', Obj).
```

where *Obj* is bound to *sushi1*. This instance is of type *Sushi* and it is similar to *Sandwich* because of the common super-class *Food-ReadyToEat*. With its spatial reasoning capabilities the robot finds out that *sushi1* is currently located in *frigde1* in room *73b2*.

Another possibility to find a sandwich is to search at locations where instances of sandwiches are created, e.g. kitchens or restaurants. Using the query

```
?- createdAt('Sandwich', Loc).
```

the robots infers that sandwiches might be are created in room *73b2* which is of type *Kitchen* and/or *subway-shop* which is of type *FastFoodRestaurant*.

Fig. 8. PR2 using an elevator, approaching subway shop, and ordering a sandwich.

VII. Related Work

In recent years, the usage of semantic information has become more and more prominent in the context of robotics.

In [4], object-room relations are exploited for mapping and navigation tasks. Such kind of knowledge is often represented by the means of Description logics [5], [6], or probabilistic models [7], [8].

How to use semantic knowledge within object search tasks is explored by [9], [10], [11]. Our approach can be considered in a similar line of research. Probabilistic models are used in [9] to guide a simulated robot during object search tasks in structured indoor environments, namely supermarkets. Work by [10] utilizes also probabilistic methods on a robot system to make inferences about the spatial relations between object instances in the environment. Similar to our approach, [11] bootstraps commonsense knowledge from the OMICS database to initialize the probabilistic representations. However, other forms of reasoning are not investigated in their robot experiments.

VIII. Conclusion and Future Work

In this paper, we investigated how semantic information about the robot's environment can be used in object search tasks. To this extent, we extended the representation formalisms introduced in previous work ([1]) and implement several PROLOG programs to infer the potential locations of an object. As proof-of-concept, we integrated the developed methods into a robot system that searches objects within a multi-level building in the context of fetch-and-delivery tasks. The realized system is able to navigate to inferred objects location using different kinds of semantic information.

In future work, we will include further probabilistic models about spatial relations like *in*, *on*, and *next-to* in order to restrict the search space of objects and thereby make the search tasks even more efficient. Additionally, we will conduct more experiments to evaluate the outcomes of search tasks more systematically.

References

[1] M. Tenorth, L. Kunze, D. Jain, and M. Beetz, "KNOWROB-MAP – Knowledge-Linked Semantic Object Maps," in *Proceedings of 2010 IEEE-RAS International Conference on Humanoid Robots*, Nashville, TN, USA, December 6-8 2010.

[2] L. Kunze, M. Tenorth, and M. Beetz, "Putting People's Common Sense into Knowledge Bases of Household Robots," in *33rd Annual German Conference on Artificial Intelligence (KI 2010)*. Karlsruhe, Germany: Springer, September 21-24 2010, pp. 151–159.

[3] R. Gupta and M. J. Kochenderfer, "Common sense data acquisition for indoor mobile robots," in *Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, 2004, pp. 605–610.

[4] C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, J. Fernández-Madrigal, and J. González, "Multi-hierarchical semantic maps for mobile robotics," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Edmonton, CA, 2005, pp. 3492–3497.

[5] H. Zender, O. Martínez Mozos, P. Jensfelt, G. Kruijff, and W. Burgard, "Conceptual spatial representations for indoor mobile robots," *Robotics and Autonomous Systems*, 2008.

[6] K. Sjöö, H. Zender, P. Jensfelt, G.-J. M. Kruijff, A. Pronobis, N. Hawes, and M. Brenner, "The explorer system," in *Cognitive Systems*, H. I. Christensen, G.-J. M. Kruijff, and J. L. Wyatt, Eds. Springer, 2010.

[7] A. Bouguerra, L. Karlsson, and A. Saffiotti, "Handling uncertainty in semantic-knowledge based execution monitoring," in *IROS*. IEEE, 2007, pp. 437–443.

[8] S. Vasudevan, S. Gächter, V. Nguyen, and R. Siegwart, "Cognitive maps for mobile robots - an object based approach," *Robotics and Autonomous Systems*, vol. 55, no. 5, pp. 359–371, 2007.

[9] D. Joho and W. Burgard, "Searching for objects: Combining multiple cues to object locations using a maximum entropy model," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, AK, USA, May 2010, pp. 723–728.

[10] A. Aydemir, K. Sjöö, J. Folkesson, A. Pronobis, and P. Jensfelt, "Search in the real world: Active visual object search based on spatial relations," in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA'11)*, Shanghai, China, May 2011.

[11] M. Hanheide, C. Gretton, and M. Göbelbecker, "Dora, a robot exploiting probabilistic knowledge under uncertain sensing for efficient object search," in *Proceedings of Systems Demonstration of the 21st International Conference on Automated Planning and Scheduling (ICAPS)*, Freiburg, Germany, June 2011.

Surface Reconstruction with Growing Neural Gas

Christian A. Mueller, Nico Hochgeschwender, and Paul G. Ploeger

Abstract— **Surface reconstruction is a crucial step to convert unstructured point clouds to a compact representation. In this paper we introduce a modified Growing Neural Gas (GNG) algorithm for surface reconstruction of free-formed objects found in domestic environments. In our experiments we show that the algorithm generates consistent surfaces and is able to cope with noise in the point clouds. Therefore, the proposed algorithm is an attractive alternative to standard triangulation methods and also applicable as a basic processing step in a perception pipeline (e.g. object categorization) for service robots.**

I. Introduction

Service robots performing household tasks (e.g. pick and place) in unconstrained and domestic settings must be able to robustly perceive object information even in the presence of cluttered and dense environments. Thereby, independently from the concrete application, surface reconstruction is an important processing step in every perception system found in service robotics. However, surface reconstruction from real-world data is challenging: the observed point clouds from objects are partial, unorganized, and free-formed. For instance, they are not necessarily based on principle shapes as cylinders, spheres, boxes, or cones. Furthermore, with noisy sensors it is difficult to generate a consistent object representation because points are missing or displaced. Naïve triangulation approaches (e.g. Delaunay based) for surface reconstruction tend to generate surfaces which are potentially overfitted with noise. Therefore, we propose in this paper an alternative approach for surface reconstruction based on Growing Neural Gas (GNG) introduced by Fritzke [1]. The modified algorithm is able to cope with free-formed, noisy, and partial point clouds from noisy RGB-D cameras like the Microsoft Kinect©.

II. Related Work

Since several decades machine learning methods based on Artificial Neural Networks have been applied to different problems – ranging from data mining to computer vision. In this paper we interpret surface reconstruction as an unsupervised learning problem. The input vector is a point of a point cloud from an object and the expected output is the surface topology of the object. A similar learning problem has been proposed by Fritzke[1] where an unsupervised Growing Neural Gas (GNG) has been introduced to incrementally learn the topological structure of input vectors. The GNG preserves the topology of the input vectors and shows a contrary effect to noisy data and outliers. This is achieved through incremental adaptation of the input.

In contrast to related approaches such as Self-Organizing Maps (SOM[4]), GNG does not assume any dimensionality of the resulting network which is in particular useful for surface reconstruction of free-formed objects. Based on this fundamental works (Fritzke[1], Kohonen[4]) several authors proposed approaches for surface reconstruction (e.g. [2], [5], [3], [6]). However, the majority of these contributions do not take into account short response times and assume perfect point clouds. For instance, far more than several seconds for reconstructing a surface are required or high-resolution point clouds ($\gg$10000 points) from objects are gathered with less noisy sensors like laser range scanners. Furthermore, the evaluations are performed with noise-free CAD or synthetic models which are mostly not available in robotics and do not comply with the sensed point clouds from RGB-D cameras.

III. Surface Reconstruction with Growing Neural Gas

In an unsupervised learning manner, GNG reflects the topology of the input distribution as an undirected graph. Basically the GNG approach proposed by Fritzke[1] is used with modifications and extensions in this work. The basic algorithm is shown in Algorithm 1, whereas the modified algorithm is shown in Algorithm 2.

Consecutively a randomly selected point of a point cloud P is fired as a signal (input) into the Growing Neural Gas G (Algorithm 1: step 4). A competitive *Hebbian* learning-like algorithm(Algorithm 1: step 5-9) is applied to generate edges between neurons which are nearest to the signal. In the original algorithm [1] a neuron is only added after a certain interval λ (Algorithm 1: step 11). It is shown that after a sufficient number of iterations of firing signals into G, that G converges to a *Delaunay* triangulated mesh.

This algorithm offers further benefits over a simple meshing algorithm: due to the incremental adaptation of GNG, it *denoises* and *reorganizes* the input distribution in such a way that only concise properties of the point cloud are reflected in the number of neurons, position and the connections between the neurons. Eventually, the connected neurons of the neural gas generate a mesh for a cup as shown in Fig.1.

Concerning the surface reconstruction problem, adding a new neuron incrementally after a certain interval λ to G (Algorithm 1: step 11) does not consider the actual divergence between G and the P. In order to adapt to this problem a new neuron n_{new} is only added to G if the $\triangle error(\cdot)$ of an existing

Christian A. Mueller, Nico Hochgeschwende and Paul G. Ploeger are with the Department of Computer Science, Bonn-Rhein-Sieg University of Applied Sciences, Sankt Augustin, Germany - christian.mueller@smail.inf.h-brs.de, {nico.hochgeschwender,paul.ploeger}@h-brs.de

Algorithm 1 Basic GNG Algorithm – Fritzke[1]

1: Select randomly two points p_1 and p_2 from P with position ω_{p_1} and ω_{p_2} to initialize G (GNG).
2: Add p_1 and p_2 to G.
3: Create an edge between p_1 and p_2.
4: Select a randomly point (aka. signal) p from P.
5: Find nearest neuron n_1 and second nearest neuron n_2 in G.
6: Increment the age of edges connected n_1.
7: Add distance between n_1 and p to error counter of n_1. $\triangle error(n_1) = ||n_1 - p||_2$.
8: Update positions(ω) of n_1 and n_n where $_n$ are neighbors of n_1. Move n_1 and n_n towards p by a fraction of ε_b and ε_b.
$\omega_{n_1} = \varepsilon_b(||n_1 - p||_2)$, $\omega_{n_n} = \varepsilon_n(||n_n - p||_2)$
9: Connect n_1 and n_2 if they are not connected.If they were connected set the edge age to 0.
10: Delete edges which are older than a_{max}. Remove neurons which do not have any edge connected to.
11: After λ selected signals add a new neuron n_{new} to G as follows:
•Identify the neuron n_q with larges $\triangle error(n_q)$.
•Insert n_{new} between n_q and the neighbor n_f with larges $\triangle error(n_f)$: $\omega_{n_{new}} = 0.5(n_q + n_f)$
•Connect n_{new} with n_q and n_f and remove the edge between n_q and n_f.
•Decrease $\triangle error(n_q)$ and $\triangle error(n_f)$ by α and set $\triangle error(n_{new})$ with $\triangle error(n_q)$.
12: Decrease all $\triangle error(\cdot)$ with d.
13: Continue with step 4 till stopping criterion(e.g. G has reached max. number of neurons) is reached

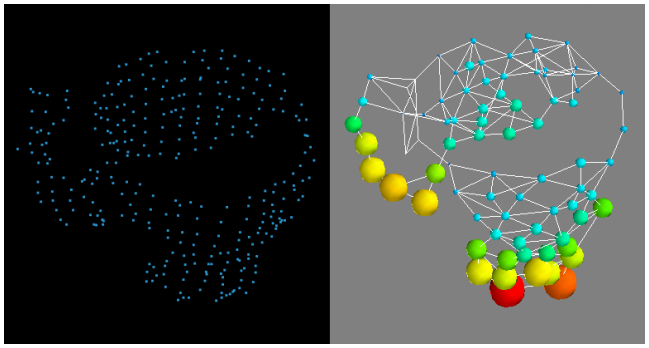


Fig. 1. A grown neural gas mesh(right) form a cup point cloud(left). Note that, a partial point cloud observation is given as input. In this and the following figures, the neurons in a GNG are sized and colored by the mean All-Pair-Shortest-Path(APSP) distances in order to illustrate the distribution and topology of a neuron with respect to the other neurons. Thereby with the APSP method the shortest paths in the GNG from each neuron to all other neurons is computed.

neuron in G exceeds a certain threshold t_γ (Algorithm 2: step 12).

As an further extension, G is *retrained*[1], i.e. the input set is repeatedly fired into G so that after each epoch, G is more adapted to the actual point cloud distribution. This guarantees that the position distribution of the neurons is close to the one of the points from the point cloud. The retraining process is stopped if the mean distance $\varepsilon(\cdot)$ between the position of the neurons n_i and the nearest points of the point cloud P reaches a lower bounded threshold t_α (Algorithm 2: step 18), instead of reaching a certain size of G (Algorithm 1: step 13). The computation of the mean distance $\varepsilon(\cdot)$ is defined as shown in Eq. 1 where N denotes the number of neurons in G. For each neuron n_i the distance to the nearest point of the point cloud P is computed; the euclidean distance (Eq. 2) is used as distance measure between two points p and q with the dimensionality of m[2].

$$\varepsilon(G) = \frac{\sum_{i=0}^{N} ||n_i - findNearestPointToNeuron(P, n_i)||_2}{N} \quad (1)$$

$$d(p,q) = ||p-q||_2 = \sqrt{\sum_{i=1}^{m}(p_i - q_i)^2} \quad (2)$$

This retraining criterion is computationally efficient and still provides a satisfying feedback about the condition of reconstruction process, for instance other measurements applied like the Hausdorff Distance in Yoon et al.[6] are computationally more expensive.

Additionally, if the distance of a neuron to the nearest point of P is above an upper bounded threshold t_β, i.e. the neuron is strongly diverted from the actual input distribution, then this neuron is removed from G – this procedure is denoted as *consistency check*($\Delta_1(\cdot)$, see Algorithm 2: step 15). This guarantees that in the current epoch of the surface reconstruction process, no nodes are contained which are certainly not part of the actual point cloud distribution. This will guide the reconstruction processes in situations where e.g. concave surfaces (e.g. cups or bowls) are reconstructed: due to the unsupervised learning manner of GNG Neurons might be attracted to the concave inner area – especially in the first epochs.

Moreover, rather than repeatedly firing the identical input set in each epoch into G, to each point of P a Gaussian noise is added ($\Delta_2(\cdot)$, see Algorithm 2: step 17) which *enriches the variation* of the input set; subsequently this results in an accelerated triangulation of a mesh compared to a mesh where adding noise is neglected (see Fig.5). We believe the acceleration can be explained by the larger variations of P with near-model-points (Gaussian noise added points of P). Due to this variation, the actual topology of the input is stronger reflected in the resulting G.

Finally in each iteration (epoch) a *refinement* procedure $\Delta_3(\cdot)$ (Algorithm 2: step 16) is applied, which only retrains

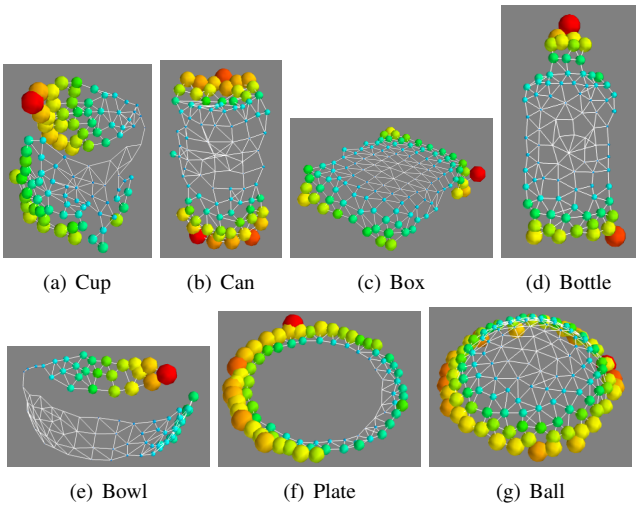
[1]Note that, a single retraining of G is also denoted as epoch.

[2]m = 3 due to the 3D Cartesian coordinates x, y, z of each point respectively neuron.

points from the point cloud which are positioned at a large curvature. This step will reinforce and consequently retain relevant surface properties of the point cloud in the GNG surface reconstruction process; e.g. surface properties which have a large curvature are edges or curves at box or cup instances.

Algorithm 2 Modified GNG Algorithm based on Fritzke[1]. Modifications are marked with $\star$.

1: $\star$ Buffer P in P_{ori}.
2: Select randomly two points p_1 and p_2 from P with position ω_{p_1} and ω_{p_2} to initialize G (GNG).
3: Add p_1 and p_2 to G.
4: Create an edge between p_1 and p_2.
5: Select a randomly point (aka. signal) p from P.
6: Find nearest neuron n_1 and second nearest neuron n_2 in G.
7: Increment the age of edges connected n_1.
8: Add distance between n_1 and p to error counter of n_1. $\triangle error(n_1) = ||n_1 - p||_2$.
9: Update positions(ω) of n_1 and n_n where $_n$ are neighbors of n_1. Move n_1 and n_n towards p by a fraction of ε_b and ε_b.
$\omega_{n_1} = \varepsilon_b(||n_1 - p||_2)$, $\omega_{n_n} = \varepsilon_n(||n_n - p||_2)$
10: Connect n_1 and n_2 if they are not connected.If they were connected set the edge age to 0.
11: Delete edges which are older than a_{max}. Remove neurons which do not have any edge connected to.
12: $\star$ A new neuron n_{new} is added to G as follows: Identify the neuron n_q with the larges $\triangle error(n_q)$. If $\triangle error(n_q) > t_\gamma$ then continue this step.
•Insert n_{new} between n_q and the neighbor n_f with larges $\triangle error(n_f)$: $\omega_{n_{new}} = 0.5(n_q + n_f)$
•Connect n_{new} with n_q and n_f and remove the edge between n_q and n_f.
•Decrease $\triangle error(n_q)$ and $\triangle error(n_f)$ by α and set $\triangle error(n_{new})$ with $\triangle error(n_q)$.
13: Decrease all $\triangle error(\cdot)$ with d.
14: $\star$ Continue with step 5 till the set of points of P are applied to G.
15: $\star$ Check for strong diverted Neurons in G, i.e. remove Neurons which exceed the distance t_β to the nearest point p in P: $G = \Delta_1(G, t_\beta)$
16: $\star$ Refine G: $G = \Delta_3(G)$
• Create points set P_{refine} from points in P at locations with a high curvature.
• Apply steps 5 to 15 where $P = P_{refine}$ – refinement is only once applied in each epoch.
17: $\star$ Adding Gaussian noise to each point of P_{ori}: $P = \Delta_2(P_{ori})$.
18: $\star$ Continue with a new epoch in step 5 till stopping criterion $\varepsilon(G) < t_\alpha$ is reached.



(a) Cup (b) Can (c) Box (d) Bottle

(e) Bowl (f) Plate (g) Ball

Fig. 2. A set of reconstructed surfaces via GNG is shown regarding the seven object categories (cup, can, box, bottle, bowl, plate, ball). Note that, missing connections like in (a) and (e) – at the rim – are due to the consistent non-existence of points in the related points clouds which also reflects the fact that *only* a partial observation from a certain perspective on the object is available; the point clouds are sensed with the Microsoft Kinect©. Moreover, note that these reconstructed surfaces do actual contain less noise, e.g. outliers.

IV. Evaluation

A. Setup

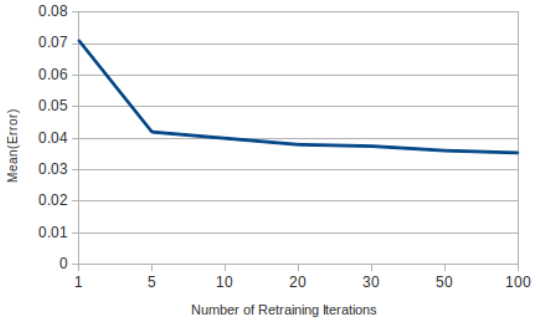
The modified GNG approach has been applied to a set of point clouds from seven different object categories, namely *cup*, *can*, *box*, *bottle*, *bowl*, *plate* and *ball*. These object categories are chosen in order to analyze the reconstructed surfaces from primitive shapes like cylinders (*cans*) or spheres (*balls*) and also to analyze the ability to reconstruct surfaces from objects with different surface properties such as convex/concave (*cups* and *bowls*), planar (*plates* and *boxes*) or spherical (*balls*) surfaces. Some instances are illustrated in Fig.2. The surfaces are reconstructed from point clouds which do not contain more than 1000 points.

The some of the following experiments are focused on one single instance from the cup category (see Fig.1); this category is chosen due to its variety of different surface properties. Furthermore, the example point cloud from Fig.1(left) is applied in different experimental conditions in order to illustrate the behavior of the proposed GNG approach.

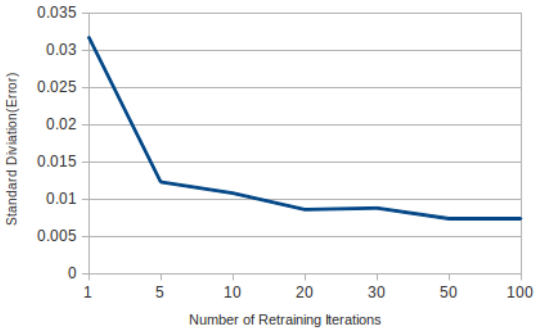
B. Experiments

Through a number of retraining epochs the GNG evolves and provides a more consistent topology projection of the input distribution. In Figure 3 the error $\varepsilon(\cdot)$ over number of epochs is analyzed between the GNG and the point clouds from the seven mentioned object categories – the error $\varepsilon(\cdot)$ is defined as shown in Eq. 1. It can be observed in Fig. 3 that the error rapidly drops after the first few epochs. Later the GNG steadily adapts to the point cloud and consequently the error decreases.

A further example is shown in Fig.4; as input the partial point cloud from Fig.1 (left) is used. It can be observed that

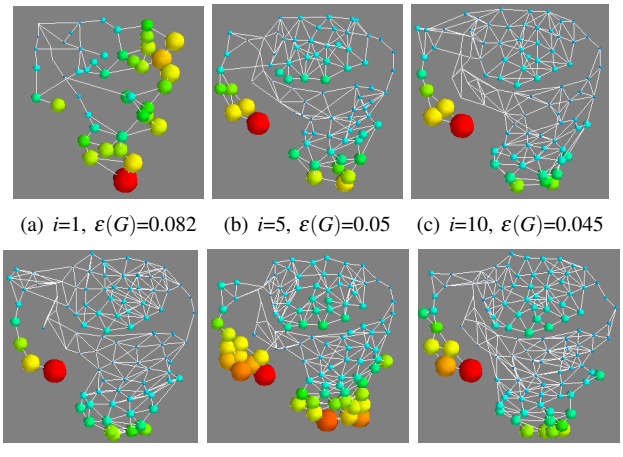


(a)



(b)

Fig. 3. Two plots which illustrate the resulting mean(a) and standard deviation(b) of the error $\varepsilon(\cdot)$ between GNGs and point clouds after a specific number of retraining iterations (epochs). The results of these plots are based on surface reconstructions from points clouds of the seven object categories (cup, can, box, bottle, bowl, plate, ball). 10 instances of each category have been used.



(a) i=1, $\varepsilon(G)$=0.082 (b) i=5, $\varepsilon(G)$=0.05 (c) i=10, $\varepsilon(G)$=0.045

(d) i=20, $\varepsilon(G)$=0.043 (e) i=30, $\varepsilon(G)$=0.041 (f) i=50, $\varepsilon(G)$=0.04

(g) i=100, $\varepsilon(G)$=0.041

Fig. 4. Example of a grown neural gas after i iterations of retraining. $\varepsilon(\cdot)$ denotes the mean error as defined in Eq. 1. The neural gas is constantly adapting to the input point cloud from Fig.1(left). The size and color of the neurons are displayed according to the mean APSP (see Fig.1).

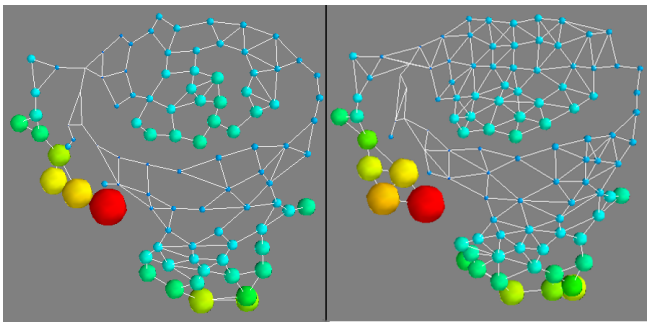


Fig. 5. Two grown neural gases trained with the point cloud from Fig.1(left). The left figure shows a grown neural gas without added noise. On the right side the figure shows a grown neural gas with added noise during retraining. In both cases the GNGs are retrained with 100 iterations. It is observable that the triangulation is more distinctive in the right figure.

the more epochs are applied the more the actual topology is projected and Delaunay triangles are observable. Also the error $\varepsilon(\cdot)$ as defined in Eq. 1 reduces over the applied epochs. However the error keeps almost steady after about 30 iterations, hence it can be assumed that the GNG is saturated by a sufficient number of neurons and their distribution is sufficiently organized over the surface. Nevertheless it is observable in Fig.4(e),(f) and (g) the error is similar with increasing iterations but the occurrences of triangles become more and more uniformly distributed.

Figure 5 shows how adding mild noise to the input point cloud[3] during the retraining process of the GNG (Algorithm 2: step 17) accelerates the Delaunay triangulation. Moreover, the number of nodes and edges have increased by applying the noise in each epoch: the number of nodes has increased from 97 to 107 whereas the number of edges from 158 to 237. As mentioned in the previous section III, we believe the acceleration of the triangulation is due to the variation enrichment of the points from the point cloud in each epoch. Based on this variation and the incremental learning process of GNG, the surface is able to be reconstruct in a more detailed and consistent manner.

Also mentioned in section III, the training process of GNG has a denoising effect on the input. This is shown in Fig. 6: the input point cloud from Fig.1(left) is reconstructed by the proposed GNG approach. Despite the noisy point cloud (e.g. caused by noisy camera input) the GNG approach is still able to cope with it and to learn the topology of the point cloud.

Finally runtime experiments have shown that the average response time is about 115 ms – considering the given seven object categories where 10 instances from each category are used.

V. Conclusion and Future Work

In this paper an approach is proposed for 3D surface reconstruction with Growing Neural Gas. Surface reconstruction based on point clouds which are sensed from the real world is challenging due to noisy sensors. The basic GNG

[3] 10% Gaussian noise to each point of the point cloud is applied.

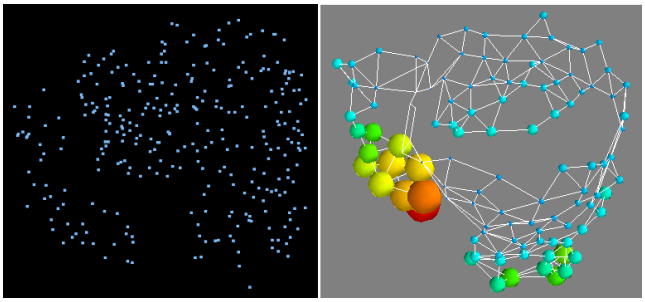


Fig. 6. The left figure displays the point cloud from Fig.1(left) with extensive noise. The resulting grown neural gas after 100 iterations is shown right. Note that the distribution – respectively topology – of the neurons is still similar to the GNG from Fig.1(right).

algorithm has been modified to adapt to the problem of 3D surface reconstruction. It has been shown that GNG provides some benefits which leads to an attractive alternative to standard triangulation approaches and also copes with noisy conditions and low response time constraints.

The future work is directed to object perception tasks, like recognition and categorization based on the proposed surface reconstruction algorithm. Typical reconstructed surfaces such as shown in Fig.2 will be used for object description purposes followed by typically learning algorithms in order to distinguish different object instances or categories. Moreover, additional investigations about the behavior of GNG are planned, e.g. measuring the quality of the triangulation and also the effect of the quality of the GNG compared to the resulting recognition, respectively categorization rate.

ACKNOWLEDGEMENT

We gratefully acknowledge the continued support by the b-it Bonn-Aachen International Center for Information Technology.

REFERENCES

[1] B. Fritzke, "A Growing Neural Gas Network Learns Topologies," in *NIPS*, 1994, pp. 625–632.

[2] I. Ivrissimtzis, W.-K. Jeong, and H.-P. Seidel, "Using growing cell structures for surface reconstruction," *2003 Shape Modeling International.*, vol. 2003, pp. 78–86, 2003.

[3] A. D. M. B. Junior, A. a. D. D. Neto, J. D. de Melo, and L. M. G. Goncalves, "An adaptive learning approach for 3-D surface reconstruction from point clouds." *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, vol. 19, no. 6, pp. 1130–40, June 2008.

[4] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[5] F. Krause, A. Fischer, N. Gross, and J. Barhak, "Reconstruction of freeform objects with arbitrary topology using neural networks and subdivision techniques," *CIRP Annals-Manufacturing Technology*, vol. 52, no. 1, pp. 125–128, 2003.

[6] M. Yoon, I. Ivrissimtzis, and S. Lee, "Self-Organising Maps for Implicit Surface Reconstruction," *Theory and Practice*, 2008.

Voxelized Shape and Color Histograms for RGB-D

Asako Kanezaki, Zoltan-Csaba Marton, Dejan Pangercic, Tatsuya Harada, Yasuo Kuniyoshi, Michael Beetz
{kanezaki, harada, kuniyosh}@isi.imi.i.u-tokyo.ac.jp, {marton, pangercic, beetz}@cs.tum.edu **

***Abstract*— Real world environments typically include objects with different perceptual appearances. A household, for example, includes textured, textureless and even partially transparent objects. While perception methods exist that work well on one such class of objects at a time, the perception of various classes of objects in a scene is still a challenge. It is our view that the construction of a descriptor that takes both color and shape into account, thereby fostering high discriminating power, will help to solve this problem. In this paper we present an approach that is capable of efficiently capturing both the geometric and visual appearance of common objects of daily use into one feature. We showcase this feature's applicability for the purpose of classifying objects in cluttered scenes with obstructions, and we evaluate two classification approaches. In our experiments we make use of Kinect, a new RGB-D device, and build a database of 63 objects. Preliminary results on novel views show recognition rates of 72.2%.**

I. Introduction

One of the great challenges in autonomous mobile robot manipulation is scaling technology to realistic tasks in real-world environments and conditions. This means that an autonomous household robot must be able to interact with many different objects of daily use, and handle them (e.g. when opening a fridge to get milk, or when performing challenging everyday tasks, such as setting the table or preparing a meal).

In recent years, descriptor-based object recognition has proved to be very successful. SIFT [1], SURF [2] and other descriptors can detect, localize and recognize many types of textured objects efficiently, reliably and under varying lighting conditions. Likewise, descriptors have been developed for perceiving objects based on their shapes (e.g. circles, cylinders, spheres and hybrid variants). The most notable are Normal Aligned Radial Features (NARF) [3] for range images and several versions of Point Feature Histograms (PFH, FPFH) [4] and Viewpoint Feature Histograms (VFH) [5] for unordered fully-3D point clouds. While these descriptors were successful, their applicability is limited because they typically work only in restricted feature spaces. In order to have a perception system based on SIFT features, the search space needs to be restricted to objects that are detectable by texture, and if 3D features are used, the objects must be distinctive with respect to their shapes. Ultimately, a perception system for autonomous robots needs to be able to perceive and decipher all objects, whether they are textured, textureless, or different shapes, or the like.

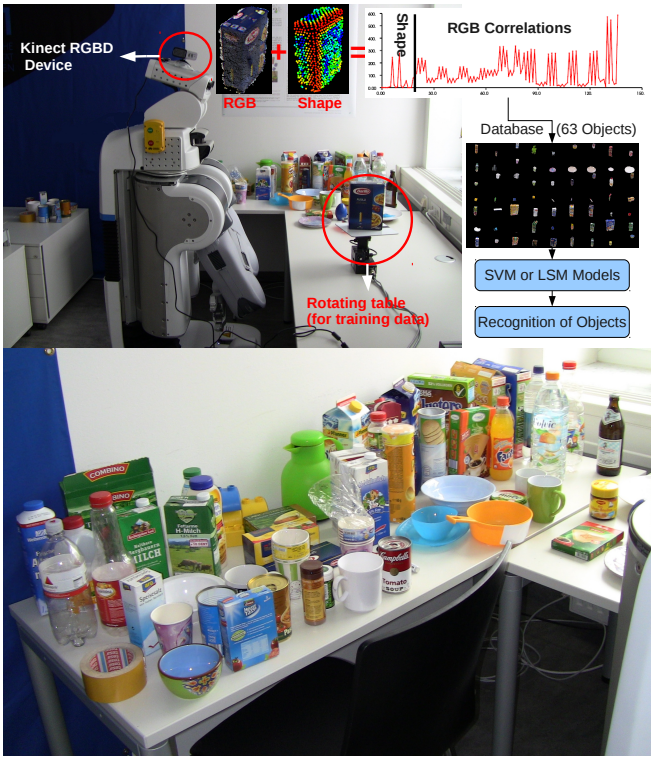


Fig. 1: Top: Autonomous service robot equipped with a Kinect sensor acquiring training data of the objects shown in the bottom image. For every view of the object, a VOSCH or ConVOSCH descriptor was estimated and a database of 63 objects was generated. An object detection pipeline with Support Vector Machines or Linear Subspace Method classifiers was then used to detect objects in natural scenes. Bottom: the objects used in our experiments.

While it seems feasible that perception routines can simply be developed by combining various descriptors [6], [7], a more promising idea is to develop descriptors that work in multiple feature spaces. The reason for this is that it might be easy to discriminate objects in a combined feature space, whereas it is impossible to discriminate them in individual spaces as depicted in Figure 2. Thus, our approach is to be seen as an alternative to the bag-of-experts type of approaches [8]. In this paper we present our work, which includes two variations of a novel descriptor that is based on the Global Radius-based Surface Descriptor

A. Kanezaki, T. Harada, and Y. Kuniyoshi are with Graduate School of Information Science and Technology, The University of Tokyo
Z.-C. Marton, D. Pangercic, and M. Michael Beetz are with Intelligent Autonomous Systems Group, TU Munich
** Graduate student authors in alphabetic order.

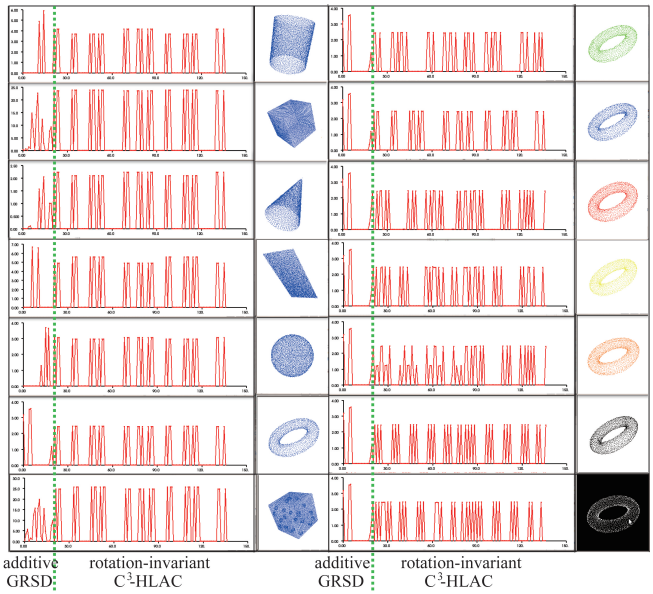


Fig. 2: Examples of scaled VOSCH histograms. Left: different categories of objects (top-down: cylinder, cube, cone, plane, sphere, torus, die) have different values in the first 20 bins of their histograms. Right: different colors of the same category of a torus have different values in the last 117 bins of their histograms.

(GRSD [7]) and the Circular Color Cubic Higher-order Local Auto-Correlation descriptor (C^3-HLAC [9]). We termed our descriptor (Concatenated) Voxelized Shape and Color Histograms – (Con)VOSCH.

The strength of our descriptor lies in its ability to consider geometric and visual features in a unified manner, thereby facilitating object classification. The two underlying features (GRSD, C^3-HLAC) were carefully selected for their similar structures and the way they are computed from voxelized 3D data. While ConVOSCH is a mere concatenation of GRSD and C^3-HLAC histograms, VOSCH is developed such that it allows for its computation in a single step because of the modified underlying algorithmic properties for GRSD and C^3-HLAC. Both features count relations between neighboring voxels, capturing geometric surface type and texture transitions at the same time. These features can be used to perform classification of one object against 63 different objects (see Figure 1) in 0.1 millisecond using Linear Subspace Method Classifier [10].

II. Related Work

Expressiveness and efficiency are both crucially important for a real-time object recognition system.The SIFT [1] descriptor, one of the most well-known 2D descriptors for object recognition, makes use of detected keypoints in scenes comparing them with referenced objects in order to identify the objects currently being observed. A combination of the visual appearance descriptor and the 2D shape is presented by Bosch et al. in [11] where they use random forests to achieve around 80% classification accuracy.

The NARF [3] descriptor detects salient points in range images of real environments. Despite its repeatability in point-to-point matching, difficulties remain in cluster-to-cluster matching, which is necessary for identifying each cluster candidate as an object in the environment. This also causes high computational costs especially when the target object database is large. In the 3D domain, VFH [5] descriptor was recently developed and presented as an extension of [4]. The feature's discriminative power is increased by the inclusion of the viewpoint, which, however, also represents a deficiency in that the feature becomes orientation variant.

There are some approaches that use both geometry and color descriptors to represent 3D bodies [12]. However, properly balancing these two different properties is difficult. Caputo and Dorko [13] learned respective kernels for shape and color representations and combined them for object recognition, while Huang and Hilton [14] balanced them by normalizing bins of shape-color histograms. An alternative solution for properly combining and balancing geometry and color information is to extract descriptors represented by patterns of shape-and-color co-occurrence. For example, the textured spin-image [15] splits the well-known spin-image [16] descriptors into several layers according to given levels of luminance. C^3-HLAC [9] splits the CHLAC [17] descriptor depending on the relative position of neighboring voxels, into RGB channels and the correlation space of these channels. In [9], the C^3-HLAC descriptor is used as a local descriptor in the training process and as a global descriptor describing each object cluster in the recognition process. This takes advantage of the combination of the Linear Subspace Method (LSM) [10] and the descriptor's additive property, which means a global descriptor for an object cluster is computed by summing up the local descriptors of its subparts.

III. System Overview

The general idea of the work presented herein is depicted in Figure 1. We first acquired the training data for 63 objects of daily use using the RGBD Kinect sensor. We then estimated ConVOSCH and VOSCH descriptors for every object view and generated training models using Support Vector Machines [18] (SVM) or Linear Subspace Method [10] (LSM) classifiers. The models were then used in the evaluation for cross-validation checks.

To construct the ConVOSCH and VOSCH descriptors, we modified the original algorithms so that the C^3-HLAC became rotation invariant and GRSD additive. We thus obtained descriptor histograms with *1001* bins for ConVOSCH and *137* bins for VOSCH. For constructing a database of object models (Figure 1 bottom) containing training examples, we used a rotating table with a pan-tilt unit that is controlled by the robot over the network. Objects placed on the rotating table were scanned at given angle intervals (15°) and then used to classify objects found in typical household settings.

Our approach is realized as part of the Robot Operating System (ROS, `http://www.ros.org/wiki/vosch`) open source framework, and makes use of modular and

robust components that can be reused on other robotic systems.

An important factor of robotic perception systems is quick performance. To achieve this we optimized the proposed descriptors, bringing the estimation time down to 0.26 s for a cluster consisting of *4632* points on average and the classification time down to under *50* milliseconds for the SVM classifier and 0.1 millisecond for the LSM.

This paper provides the following main contributions:

- A specification of a novel descriptor with two variants (ConVOSCH and VOSCH) that account for geometrical as well as visual appearance properties of objects;
- A comparison of classification results for two established classification frameworks – LSM and SVM; and,
- An extensive database of 63 objects of daily use captured with the Kinect sensor and used as training examples (which we intend to publish after the review process).

IV. Feature Estimation

As mentioned earlier, the construction of ConVOSCH and VOSCH is based on the features of C^3-HLAC and GRSD. For the sake of clarity, we briefly present an overview of how the features of HLAC and GRSD are constructed. Following this, we outline the modifications that were made to these features in order to obtain the novelty of the proposed features.

A. C^3-HLAC

C^3-HLAC descriptor is a high-dimensional vector that measures the summation of the multiplied RGB values of neighboring voxels in a $3\times3\times3$ grid around a voxel grid of arbitrary size. Each bin in a C^3-HLAC descriptor is differentiated by the RGB color space and the relative position of the two neighboring voxels. Let $\boldsymbol{x} = (x, y, z)^T$ be the position of a voxel, $p(\boldsymbol{x})$ be the flag for occupancy of the voxel and $r(\boldsymbol{x})$, $g(\boldsymbol{x})$ and $b(\boldsymbol{x})$ be its RGB values normalized between 0 and 1, respectively. By defining $r_1(\boldsymbol{x}) \equiv sin\left(\frac{\pi}{2}r(\boldsymbol{x})\right)$, $g_1(\boldsymbol{x}) \equiv sin\left(\frac{\pi}{2}g(\boldsymbol{x})\right)$, $b_1(\boldsymbol{x}) \equiv sin\left(\frac{\pi}{2}b(\boldsymbol{x})\right)$, $r_2(\boldsymbol{x}) \equiv cos\left(\frac{\pi}{2}r(\boldsymbol{x})\right)$, $g_2(\boldsymbol{x}) \equiv cos\left(\frac{\pi}{2}g(\boldsymbol{x})\right)$, and $b_2(\boldsymbol{x}) \equiv cos\left(\frac{\pi}{2}b(\boldsymbol{x})\right)$, a voxel status $\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{N}^6$ is defined as follows:

$$\boldsymbol{f}(\boldsymbol{x})=\begin{cases} [r_1(\boldsymbol{x})\ r_2(\boldsymbol{x})\ g_1(\boldsymbol{x})\ g_2(\boldsymbol{x})\ b_1(\boldsymbol{x})\ b_2(\boldsymbol{x})]^T & p(\boldsymbol{x})=1 \\ [\ 0\,0\,0\,0\,0\,0\ ]^T & p(\boldsymbol{x})=0 \end{cases}$$

Letting $\boldsymbol{a}_i$ be a displacement vector from the reference voxel to its neighboring voxel, the elements of a C^3-HLAC descriptor extracted from a voxel grid V are calculated by the following equations:

$$\boldsymbol{q}_1 = \sum_{\boldsymbol{x}\in V} \boldsymbol{f}(\boldsymbol{x}) \tag{1}$$

$$\boldsymbol{q}_2 = \sum_{\boldsymbol{x}\in V} \boldsymbol{f}(\boldsymbol{x})\ \boldsymbol{f}^T(\boldsymbol{x}) \tag{2}$$

$$\boldsymbol{q}_3(\boldsymbol{a}_i) = \sum_{\boldsymbol{x}\in V} \boldsymbol{f}(\boldsymbol{x})\ \boldsymbol{f}^T(\boldsymbol{x}+\boldsymbol{a}_i) \quad (i=0,\dots 12) \tag{3}$$

Since these values are summed up around a voxel grid, there is redundant computation of the same value over symmetric pairs of $\boldsymbol{a}_i$. As a result, the number of variations in $\boldsymbol{a}_i$ is 13, which is a half of the 26 neighbors in a $3\times3\times3$ grid. The matrix computed by (3) is expanded into a column vector of 36 elements. Therefore, the dimension of the vector calculated by (1) is 6, while that by (3) is 468 (=$36\cdot13$). The second part of the C^3-HLAC descriptor is computed from the binarized values of $r(\boldsymbol{x})$, $g(\boldsymbol{x})$ and $b(\boldsymbol{x})$. To determine the threshold of color binarization, we applied the histogram threshold selection method of [19] to the R, G and B values respectively, using the voxel colors of all objects in the database as sample data. C^3-HLAC features calculated by (2) include redundant elements, e.g., $r(\boldsymbol{x})\cdot g(\boldsymbol{x})$ and $g(\boldsymbol{x})\cdot r(\boldsymbol{x})$. Excluding the redundant elements, the dimension is 12 if color values are binarized, and 21 otherwise. Finally a full C^3-HLAC vector is obtained by concatenating the two vectors from binarized color voxel data and from original color voxel data. As a result, the dimension of the C^3-HLAC feature vector becomes 981 (6+468+21 for non-binarized data plus 6+468+12 for binarized data).

B. GRSD

GRSD is a histogram, that counts the number of transitions between different types of voxels. While it is applicable to a wide range of applications, we used it to count the transitions between the following geometric classes of voxel surfaces: *free space, plane, cylinder, sphere, rim, and noise*. Surface types were identified based on the estimation of their two principal curves' radii, r_{min} and r_{max} [20]. This approach uses the leaves of an octree as voxels, to facilitate fast ray intersection tests. The number of bins b is:

$$b = \frac{s(s+1)}{2} \tag{4}$$

where s is the number of possible surface types, resulting in 21 dimensions for the above stated 6 types of surfaces.

In order to efficiently merge the two features, the original GRSD had to be altered (see next subsection). In the new, additive GRSD$_2$ feature, transitions from *free space* to *free space* are impossible, hence its dimensionality becomes 20.

C. VOSCH

In order to generate VOSCH we refined the C^3-HLAC descriptor to be rotation-invariant by bringing all 13 different vectors given by (3) together in the following equation:

$$\boldsymbol{q}_4 = \sum_{i=0}^{12} \boldsymbol{q}_3(\boldsymbol{a}_i) = \sum_{i=0}^{12}\sum_{\boldsymbol{x}\in V} \boldsymbol{f}(\boldsymbol{x})\ \boldsymbol{f}^T(\boldsymbol{x}+\boldsymbol{a}_i) \tag{5}$$

This reduces the dimensionality of the descriptor down to 117 (6+36+12 for non-binarized data plus 6+36+21 for binarized data) with respect to the original implementation. By doing so, the refined C^3-HLAC descriptor is well-matched with the GRSD histogram, making it rotation invariant, while preserving expressivity.

The GRSD, as presented in [7], originally used ray-tracing in order to find transitions between surface types. To make

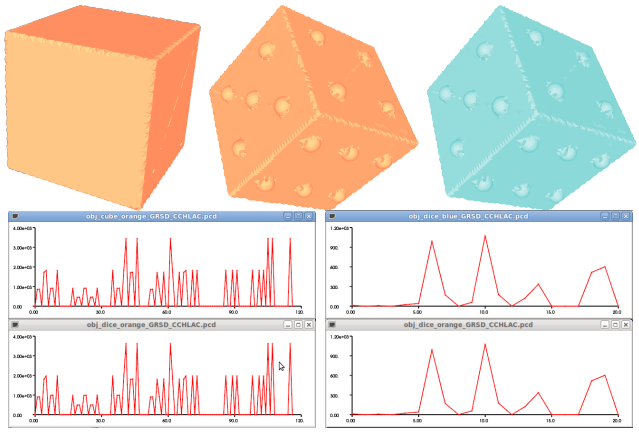


Fig. 3: Rotation-invariant C^3-HLAC can not differentiate the die from the cube (identical histograms on the left), while GRSD/$GRSD_2$ can not differentiate the different colors (identical histograms on the right). Their combination however produces distinct signatures for all of them (Figure 2).

it fit into VOSCH we i) omitted ray-tracing in favor of fast-to-compute adjacency voxels checks and ii) discarded the normalization of histogram bins – thus obtaining $GRSD_2$.

These modifications allow us to create histograms which, as in the case of C^3-HLAC, are additive. The latter property can best be described as follows: if we break an object into several parts, the object's histogram becomes equal to the sum of the histograms of its sub-parts. Furthermore, since we do not use ray-casting but consider only the direct neighbors of each occupied cell, the computation of the new $GRSD_2$ is in the millisecond range. This is 2 orders of magnitude faster than the original implementation, at the cost of some descriptiveness.

D. ConVOSCH

The idea of ConVOSCH is to simply estimate C^3-HLAC and $GRSD_2$ in two separate steps, concatenate resulting histograms and normalize the final result into a range of [0,1]. This preserves the high dimensionality and, thus, the accuracy for the color space, but it also makes the feature rotation-variant and slightly slower for use in classification schemes.

The advantage of the VOSCH and ConVOSCH features is depicted in Figure 3, where we can see that the visual appearance-based feature, such as C^3-HLAC, can not discern the orange cube from the orange die (note that the histograms in the bottom-left part are identical). On the other hand, we have a shape-based feature GRSD/$GRSD_2$ which can not discern between the orange and the blue die (likewise, the histograms in the bottom-right part are also identical).

V. Classification Methods

A. Linear Subspace Method

LSM is an established learning method for classification. For the first step, we solve PCA for all the descriptors extracted from all the trained objects, and then use the top d dimensions of the descriptor as a feature vector, in the same way as [9]. For training, several feature vectors that represent each object were extracted, and then Principal Component Analysis (PCA) for them was solved. Similarly to [9], we divided the whole voxel grid of each object into cubic subdivisions of a certain size (7 cm $\times$ 7 cm $\times$ 7 cm with 5 cm $\times$ 5 cm $\times$ 5 cm overlapping in our case) and then extracted feature vectors from all of the subdivisions. Let the number of subdivisions be N and the training feature vectors be $\boldsymbol{z}_t \in R^d, t = 1, 2, ...N$. Then the eigenvectors of $R = \frac{1}{N}\sum_{t=1}^{N} \boldsymbol{z}_t \boldsymbol{z}_t^T$ are computed.

To classify and identify a detected object in environments, the system extracts one feature vector from the whole voxel grid of the object. Letting the feature vector be $\boldsymbol{z}$ and the matrix of top c eigenvectors for the i-th object in the database be $P_i \equiv (\boldsymbol{v}_{i1}\boldsymbol{v}_{i2}...\boldsymbol{v}_{it})$, the similarity between the query part and the i-th object in the database y_i is calculated by $y_i = \|P_i^T \boldsymbol{z}\|/\|\boldsymbol{z}\|$. There is an advantage to using LSM with this kind of histogram-based descriptors which are additive. Owing to this property, one feature vector computed from each object cluster in environments can be classified by projecting it to each subspace of database objects, regardless of the size of the object cluster, thus providing a fairly fast method for classifying partially visible objects.

B. Support Vector Machine-based Classification

We also performed classification using SVM in our experiments. SVM is a fast classification method that works by learning the vectors that define hyperplanes separating the training examples according to a cost and a kernel function.

Unlike LSM, it is not well-suited for recognizing partially visible objects using the presented features, unless partial views are explicitly trained. Another problem is over-fitting to the training data, which can be limited by choosing parameters so that they maximize the results of cross-validation checks on the training data.

We used a C-SVC type SVM with a radial basis function kernel [21] with $\gamma = 0.0078125$ (0.5 for $GRSD_2$). Cost values higher than 128 did not further improve the final results.

C. Scaling of Features

To improve classification rates using LSM it is important that the values of the feature do not vary over several orders of magnitude and, thus, become very important components during PCA. Similarly, for SVM the results are also improved if the variation of each feature bin is proportional. Therefore we performed bin-wise scaling of the features for classification as follows: when going through the training data, the minimum and maximum value of each bin was recorded, and this interval was mapped to the interval [0,1] both during training and during testing.

VI. Results

To evaluate the proposed features we ran tests using SVM and LSM classifiers on the following two sets of data: i) 7 artificially generated objects with 7 different colors, and ii) 63 objects of daily use, scanned using a Kinect

	$\overline{nr_{points}}$	$\overline{t_{point}}$	$\overline{t_{object}}$
synthetic data (49 views of 49 objects)	19124	36 μs	0.69 s
real data (1512 views of 63 objects)	4632	57 μs	0.26 s

TABLE I: Times needed for the estimation of VOSCH.

	GRSD$_2$	**C^3-HLAC**	**ConVOSCH**	**VOSCH**
LSM	43.65 (39.48)	99.67 (99.6)	**99.8** (**99.74**)	**99.8** (99.67)
SVM	99.07 (73.81)	**99.87** (99.6)	**99.87** (**99.67**)	99.8 (98.94)

TABLE II: Model accuracy of the training data and for the leave-one-out test – the latter in parentheses.

sensor and a rotating table (shown in the bottom part of Figure 1). We measured the recognition rates against the original implementations of C^3-HLAC and GRSD$_2$. Since C^3-HLAC was already shown to outperform Textured Spin Images in [22] we did not include it in our experiments.

A. Feature Extraction

In all tests, the computation of the feature was parametrized as follows:

- search radius for normals: 2 cm
- search radius for r_{min} and r_{max} estimation: 1 cm
- voxel size: 1 cm

The execution times for VOSCH feature estimation are shown in Table I, where $\overline{nr_{points}}$ denotes the average number of points per object, $\overline{t_{point}}$ is the average estimation time per point and $\overline{t_{object}}$ is the average estimation time needed for an object.

B. Synthetic Data

We first carried out tests on a set of 49 artificially generated objects, consisting of 7 shapes, each in 7 different colors (Figure 2). We used one example of each object as a training sample and evaluated the model on 5 examples of each object with 10 different noise levels. The noise levels were generated for points' 3D coordinates using Gaussian distribution with the following standard deviations:

$$\sigma \in \{0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0\} \text{ [mm]}$$

The deviations were set to correspond to the actual noise levels of a Kinect sensor[1]. Note that not partial but the whole shape of each object was given as an example in this experiment. For LSM classification we set the dimension of the compressed feature space d to 50 for C^3-HLAC, ConVOSCH and VOSCH, while GRSD$_2$ was left uncompressed. The dimensionality of the subspace c was set to 3. The results of the test in Figure 4, show VOSCH (right-most two columns) outperforming all other features.

Interrestingly, LSM is slightly better than SVM at low noise levels, but is superseded by SVM as noise increases. This is because SVM avoids over-fitting by maximizing the margin between the different classes, while LSM penalizes deviations from the model (i.e. "regression line" of the training features).

Since GRSD$_2$ is insensitive to color, it can identify at most $1/7 \approx 14.2857\%$ of the data. Additionally, since objects were uniformly colored and mostly symmetrical, the fact that C^3-HLAC is not rotation invariant did not influence the results. In the original implementation C^3-HLAC had to be trained in artificial rotations to achieve rotation invariance.

[1] `http://www.ros.org/wiki/openni_kinect/kinect_accuracy`

C. Real Data

1) Acquisition of Training Data: Our database of 3D objects was obtained using the Kinect sensor mounted on the head of a robot. The set of objects (see Figure 1) encompasses those commonly used in a typical household environment (mugs, utensils, groceries, etc.).; we plan to expand this object set and release it after the review process. We rotated the objects on the rotating table with an angular step of 15° around the up-axis, and acquired partial snapshots from a perspective that best approximates the robot's viewpoint during its working cycle.

2) Evaluation of the Training Data: Table II shows the percentage of correctly classified training examples using the model computed for them. The numbers in parentheses are the percentages of correctly classified views using the model constructed using the remaining views (leave-one-out-test). For LSM classification we set d to 100 for C^3-HLAC, ConVOSCH and VOSCH, while GRSD$_2$ was left uncompressed. The dimensionality of the subspace c was set to 10 for GRSD$_2$ and 50 for the others.

3) Evaluation on Novel Views: To evaluate the proposed features of the novel views, we performed the test with the following setup : (a and b) we selected 5 types of scenes containing textured and textureless objects, (c) objects with similar shapes, (d) a scene with the substantial altered lighting conditions, and (e) objects with arbitrary rotations. We acquired the data from three substantially different views.Object candidate clusters were detected by first finding a major horizontal planar surface within the point cloud, as done in [23]. For these experiments, we tested all the choices with a break for ten (10, 20, ...) as the dimension of subspace c in LSM, and set the best one.

In total we tested 72 views, achieving the highest rates with SVM, as shown in Table III. One possible way to improve the results would be to treat each view as a separate sub-class, which should increase the actual classification results by avoiding the training of very different views of the same object into one class. Such an approach improved the success rate by 12% of GRSD based geometric classification.

While LSM proved to be robust to noise in the synthetic data, the real data is quite smooth, and SVM outperformed it in our tests. LSM has, however, the advantage of exploiting the additive property of the feature for detection in clutter. Moreover, the time required for classification is significantly shorter with LSM than with SVM. To validate LSM's robustness to parial data in clutter, additional experiments with larger number of partially observed objects are the most important future work.

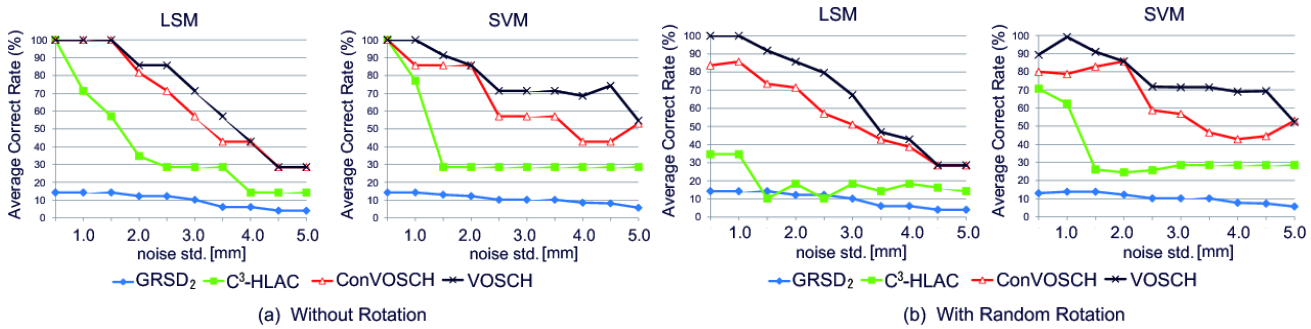


Fig. 4: The effect of simulated noise on the classification results using $GRSD_2$, C^3-HLAC, ConVOSCH and VOSCH with LSM and SVM (both with and without random rotations of test data).

		$GRSD_2$	C^3-HLAC	ConVOSCH	VOSCH
(a) texture	**LSM**	16.7%	75%	**91.7%**	83.3%
	SVM	50%	75%	**83.3%**	66.7%
(b) no texture	**LSM**	16.7%	44.4%	**61.1%**	44.4%
	SVM	44.4%	**66.7%**	61.1%	61.1%
(c) sim. shape	**LSM**	5.6%	33.3%	44.4%	**55.6%**
	SVM	22.2%	61.1%	72.2%	**77.8%**
(d) diff. light	**LSM**	5.6%	50%	50%	**61.1%**
	SVM	22.2%	**88.9%**	**88.9%**	72.2%
(e) arb. rotation	**LSM**	16.7%	16.7%	16.7%	**100%**
	SVM	0%	16.7%	33.3%	**50%**
Total	**LSM**	11.1%*	45.8%	55.6%	**63.9%**
	SVM	30.6%*	68.1%	**72.2%**	68.1%

TABLE III: Model accuracy on novel view data. *Please note that while in the original implementation GRSD was used for detecting the geometric category, here we used it for classification of objects that were only visually distinguishable (different brands of milk boxes for example).

VII. Conclusions and Future Work

In this paper, we presented and evaluated a method to efficiently combine two appearance characteristics into a single feature (both rotation variant and rotation invariant) and showed its advantage over the original methods.

With its low number of dimensions, the rotation invariant VOSCH feature promises to be an efficient and descriptive feature that scales well with the number of objects. C^3-HLAC is not rotation invariant, so it has to be trained with all rotations around the view-ray, and thus, so does ConVOSCH. However, we found that at least in regards to our 63 objects, the texture variations were not significantly different when the objects were rotated – neither for the single colored synthetic data (Figure 4), nor for the real views (Table III).

Future work will include the incorporation of other promising 2.5/3D approaches, like VFH and NARF, into the voxelized feature extraction process and further evaluations using more test and training data.

References

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[3] B. Steder, R. B. Rusu, K. Konolige, and W. Burgard, "NARF: 3D range image features for object recognition," in *Proc. IEEE IROS Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics*, 2010.

[4] R. B. Rusu, N. Blodow, and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3D Registration," in *Proc. IEEE ICRA*, 2009.

[5] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram," in *Proc. IEEE IROS*, 2010.

[6] J. Stckler and S. Behnke, "Combining depth and color cues for scale- and viewpoint-invariant object segmentation and recognition using random forests." *Proc. of IROS*, 2010.

[7] Z.-C. Marton, D. Pangercic, R. B. Rusu, A. Holzbach, and M. Beetz, "Hierarchical object geometric categorization and appearance classification for mobile manipulation," in *Proc. IEEE Int. Conf. on Humanoid Robots*, 2010.

[8] M. Varma, "Learning the discriminative powerinvariance trade-off," in *Proc. IEEE ICCV*, 2007.

[9] A. Kanezaki, T. Suzuki, T. Harada, and Y. Kuniyoshi, "Fast object detection for robots in a cluttered indoor environment using integral 3D feature table," in *Proc. IEEE ICRA*, 2011.

[10] S. Watanabe and N. Pakvasa, "Subspace method in pattern recognition," in *Proc. 1st Int. Joint Conf. on Pattern Recognition*, 1973.

[11] A. Bosch, A. Zisserman, and X. Munoz, "Representing shape with a spatial pyramid kernel," in *Proc. the 6th ACM Int. Conf. on Image and video retrieval*, ser. CIVR '07, 2007, pp. 401–408.

[12] S. Park, X. Guo, H. Shin, and H. Qin, "Surface completion for shape and appearance," *The Visual Computer*, vol. 22, no. 3, 2006.

[13] B. Caputo and G. Dorko, "How to combine color and shape information for 3D object recognition: kernels do the trick," in *Proc. NIPS*, 2002, pp. 1375–1382.

[14] P. Huang and A. Hilton, "Shape-colour histograms for matching 3D video sequences," in *Proc. IEEE ICCV Workshops*, 2009, pp. 1510–1517.

[15] G. M. Cortelazzo and N. Orio, "Retrieval of colored 3D models," in *Proc. the Third International Symposium on 3DPVT*, 2006.

[16] A. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 21, no. 1, pp. 433 – 449, 1999.

[17] T. Kobayashi and N. Otsu, "Action and simultaneous multiple-person identification using cubic higher-order local auto-correlation," in *Proc. IEEE ICPR*, vol. 4, 2004, pp. 741–744.

[18] O. Chapelle, P. Haffner, and V. N. Vapnik, "Support vector machines for histogram-based image classification," *IEEE Trans. on Neural Networks*, vol. 10, no. 5, pp. 1055–1064, 1999.

[19] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[20] Z.-C. Marton, D. Pangercic, N. Blodow, J. Kleinehellefort, and M. Beetz, "General 3D Modelling of Novel Objects from a Single View," in *Proc. IEEE IROS*, 2010.

[21] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[22] A. Kanezaki, H. Nakayama, T. Harada, and Y. Kuniyoshi, "High-speed 3D object recognition using additive features in a linear subspace," in *Proc. IEEE ICRA*, 2010.

[23] R. B. Rusu, I. A. Sucan, B. Gerkey, S. Chitta, M. Beetz, and L. E. Kavraki, "Real-time Perception-Guided Motion Planning for a Personal Robot," in *Proc. IEEE IROS*, 2009.