
| | |
|---|---|
| Project Acronym: | GRASP |
| Project Type: | IP |
| Project Title: | Emergence of Cognitive Grasping through Introspection, Emulation and Surprise |
| Contract Number: | 215821 |
| Starting Date: | 01-03-2008 |
| Ending Date: | 28-02-2012 |


| | |
|---|---|
| Deliverable Number: | D22 |
| Deliverable Title : | Closed-loop refinement of the representation attributes |
| Type (Internal, Restricted, Public): | PU |
| Authors | Ch. Papazov, D. Burschka, Janette Bohg, D. Kragic, H. Deubel |
| Contributing Partners | TUM, KTH, LMU |

| | |
|---|---|
| Contractual Date of Delivery to the EC: | 28-02-2011 |
| Actual Date of Delivery to the EC: | 28-02-2011 |

# Contents

# Chapter 1

# Executive summary

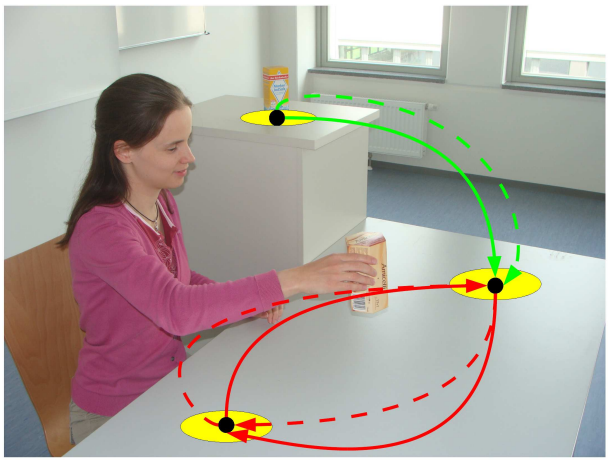
Deliverable D22 is part of WP5 -"Surprise: Detecting the Unexpected and Learning from it". According to the Technical Annex, it presents the activities in the context of

- **Task 5.2** - Evaluation of efficient methods to monitor changes in the environment that will be insensitive to sensor inaccuracies and that compensate eigen-motions/actions of the system in the environment. In collaboration with the WP4, an internal representation of the environment is generated that will define the expectations of the system. This representation goes beyond a geometric representation of the world and will define also contextual and dynamic information about the world
- **Task 5.3** - Validation of action primitives through combination of the sensor perception

The work in this deliverable relates to the following third year milestones:

- **Milestone 7** - Observing consequences of grasping; vocabulary of robot action/interactions and definition of a hierarchical structure of features
- **Milestone 9** - Integrating contextual representation in the knowledge representation and development of the attention system with view planning

In this reporting period, our focus was on: **observing human actions** and representing them in the internal action model used for system's own manipulation actions, extension of the **shape registration to deal with deformable and significantly occluded observations** as indexing technique to our a-priori knowledge database (Atlas), and **prediction, validation and parametrization of a-priori knowledge** from interaction with the environment while transferring the knowledge from the Atlas to the working memory.



- **Observing Human Actions** - A human, who is manipulating objects in the scene, usually does not repeat an action on exact the same trajectory but he or she repeats just certain properties of the given manipulation. Consequently, a robot observing a manipulation should extract these properties from the observation and generalize them instead of storing the exact 3D-trajectory. Some of the extracted properties are related to the object itself (e.g. motion constraints). On the other hand, the geometric position of an object determines the possible actions. Therefore, a *Functionality Map* is introduced, which relates the manipulation-relevant object properties to the environment. The environment is represented as abstract regions (Location Areas), which are relevant for the manipulation of a specific object. Furthermore, an *Object Container* is used, which stores the properties directly related to the object itself. The stored generalized knowledge is used to detect unexpected changes in the observed properties of a manipulation task. Unexpected events (surprise) can be detected efficiently at this level reducing the number of false positive alerts caused by variations in the human manipulation examples. The proposed system uses a general, object-centric representation. This enables a transfer not only within the same, but additionally to similar environments. Furthermore, the structure makes the system more independent of the perception source. This is an important property to achieve the goal of combination of different sensor perception required in the Task 5.3. It is important to notice, that neither the exact reconstruction of the environment in the sense of navigation nor the pure repetition of an action is of interest here. The aim is the understanding of the manipulation constraints for a given manipulation sequence.
- **Shape registration to deal with deformable and significantly occluded observations** - Significant contribution in this reporting period is also the work on registration of objects with rigid and deformable shape to the Atlas representation of the objects a-priori known to the system. The registration is a basic component for indexing the knowledge in the database and the work was extended from exact match to more generic descriptors. The deformable registration allows not only a generalization of the object descriptions in the Atlas but it allows also to map knowledge from human observation or previous interactions with similar objects to a modified geometry of the new object. This allows to generate hypothesis how to interact with a novel object based on similarities to the generic information in the Atlas.
- **Prediction, validation and parametrization of a-priori knowledge** - Finally, we started work on estimation of dynamic, physical properties of objects through direct interaction with them. We implemented an initial predict-act-observe loop which allows us to estimate the exact parameters of an object in the scene based on the parametric information stored in the Atlas (a-priori) knowledge. This work includes a fusion of dynamic engine predictions with observation of the camera to estimate the non-observable parameters of the object like: mass, its center, distribution and rigidity among others.

The following text is structured as follows. In Chapter 2.1, we give a short overview over our current results in parsing of human actions. In Chapter 2.2, we present the current results in deformable indexing to our generic object database. Chapter 2.4 presents our preliminary work on completion of physical properties of the objects in the scene.

# Chapter 2

# Description of Work

## 2.1 Surprise Detection from Observation of Human Actions

The representation of the manipulation-relevant object knowledge consists of two main components: the Object Container and the Functionality Map. The representation of the manipulation-relevant object knowledge is independent of the method used for the trajectory acquisition. An example is given in Fig. 2.1. It illustrates the Object Container with the object properties and the representation of the actions in the Functionality Map. The Functionality Map consists of the Location Areas, between which actions are performed. The properties of these actions are stored in the Functionality Map. Figure

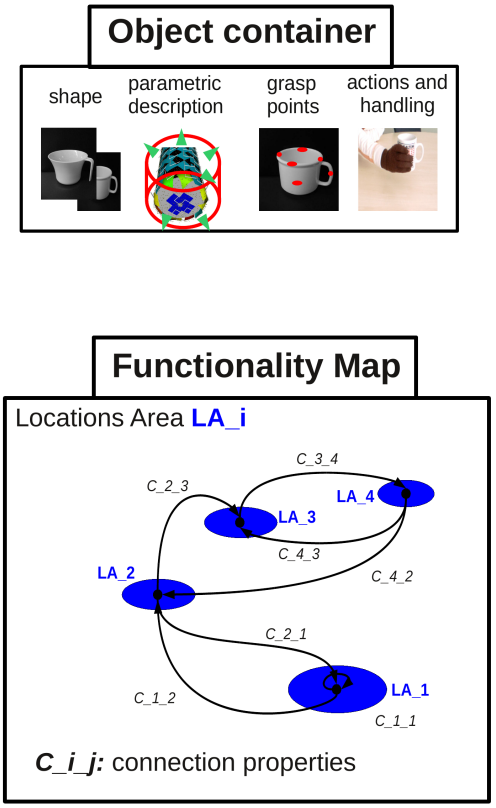



Figure 2.1: Object Container and Functionality Map. The Object Container consists of object properties. The Functionality Map is an abstract representation of the manipulation-relevant functionalities in the environment.

### 2.1.1 Manipulation-Relevant Object Representation

Robot's environment can be very complex, containing varying objects in rooms with different furniture. Consequently, the robot has to deal with human demonstrations in complex scenes. Such a complex scene requires to focus the attention on the relevant information in the scene. The robot needs to detect and observe the object, which is manipulated by the human and, therefore, currently relevant to the robot. This mission-relevant object is considered foreground in contrast to the background, which is the remaining geometric structure used for obstacle avoidance. The selection of the foreground object and its monitoring is triggered by the human interaction with it. It is analogon to our earlier Vision Interaction Cues (VICs) approach: For each object, its actions and its monitoring space around itself are defined by the object, speeding up the processing of the human actions. The Atlas (Long-term memory) stores the already attained, general object knowledge as well as a-priori knowledge. This general knowledge can be mapped into the current scene (Fig. 2.2). The information about the object in the current scene is stored in the Working Memory.

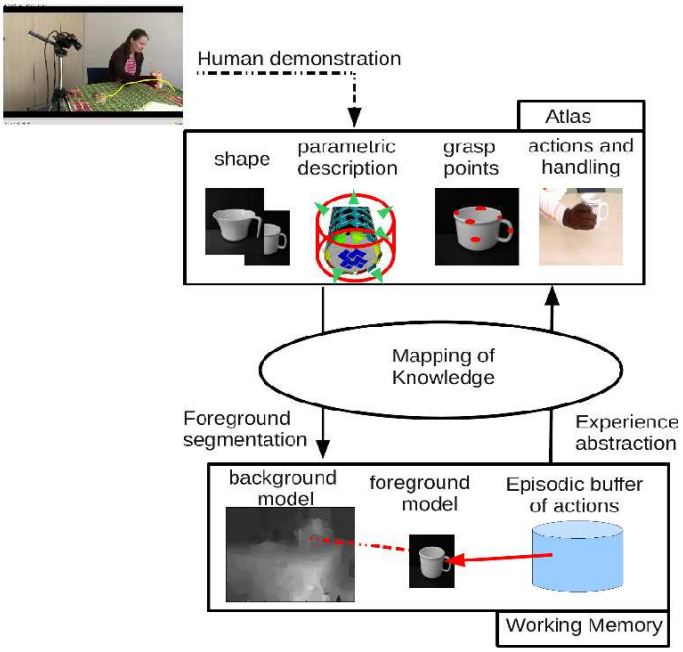



Figure 2.2: Object Container is mapped from Atlas to the Working Memory of the system.

Since a general knowledge of the object properties is of interest, not the simple records of a trajectory's x,y,z-coordinates but the abstract handling properties are of interest for comparison of human actions. We consider as important property for each object: the orientation, the maximal allowed acceleration, the mass and the center of gravity. Some of these properties are not observable with a pure vision-based system or a pure tracking system. The handling properties themselves are constraints, which limit the handling possibilities for an object in a certain situation. For example, an observed rotation of a manipulated object indicates, that the object does not need to be kept in the initial orientation. Consequently, the object-orientation does not put a constraint on the manipulation in this context. The object-centric representation has the advantage, that the representation of a constraint is independent of the exact position in space. It provides merely a constraint on the motion parameters. For example, a cup filled with coffee must not be tilted during the manipulation. Therefore, the state of the cup implies the constraint in this case. A change in handling is an indication on possible change in the state of the object. It generates a surprise event in our system that forces the system to update the internal knowledge about the object.

### 2.1.2 Functionality Graph

Functionality Graph represents the pose dependent actions on the object in a given environment. The first element of the Functionality Map are the Location Areas. They encode the locations in 3D space,

where a manipulation sequence started or ended. Location Areas are explicitly defined by the observed resting position of the object. They are areas and not single points, since an object is usually placed in a certain area and not on one exact spot. These observed Location Areas have the observed connections between them representing possible transport goals. A Location Area does not necessarily correspond to a geometric structure in the environment. The position of a handover of an object can also establish a Location Area. The established Location Area is therefore not restricted to a surface, but to a region in space.

The connections between Location Areas are the additional elements of the Functionality Graph. A connection exists between two Location Areas, if an action has been performed directly between both areas without visiting another Location Area. Manipulation properties of the actions, which are performed on this connection. Of course, the propertiest factor are the objects themselves. The second factor are the different manipulation alternatives, which can occur for each object. Therefore the system needs to store the different alternatives, which can occur for each object and its manipulation alternatives. Two exemplary objects of a Functionality Map can be seen in Fig. 2.3.

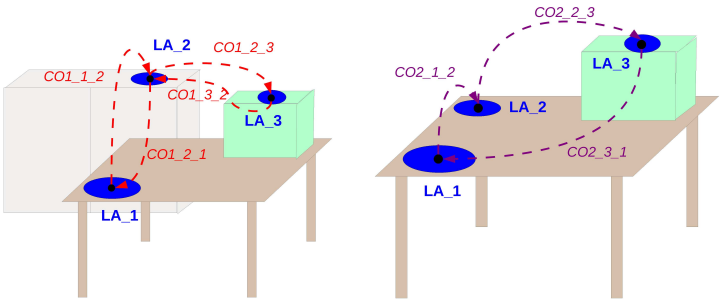



Figure 2.3: Functionality Graphs for different objects in the scene.

The properties stored in the Functionality Graph are the following:

- **pushed object vs. lifted object** - An object can be manipulated by lifting or by pushing it. A pushed object needs just to be pushed in the desired direction, whereas lifting an object requires much more e rt (e.g., knowledge about the way of grasping, the object's weight).
- **arbitrary movement vs. movement with a goal** - The trajectory between two Location Areas has either an arbitrary shape or it represents a movement with a goal. A movement with a goal connects the Location Areas in a direct manner while avoiding detours. In contrast, an arbitrary movement has not such a directed shape. Consequently, the movement with a goal sets a constraint on the possible trajectories, whereas an arbitrary movement does not.
- **action probabilities** - The probabilities of the connections show the probability of a manipulation-relevant property, based on the observed actions.
- **max. speed during pick-up** - The three phases defining an action are used: the pick-up, the transportation and the placement phase. The maximal speed during the pick-up phase is stored in the Functionality Map, since it is an indicator of the difficulty to pick up the object.
- **grasp taxonomy** - The grasp type is mainly important for the pick-up and placement phase of the manipulation and is not part of this work.
- **approach vector** - The approach vector is, similarly to the grasp type, mainly important for the pick-up and placement phase of the manipulation and is not part of this work. The approach vector is the direction, from which the object is grasped in the object-centric point of view.

### 2.1.3 Performance of the Observation Approach

We observed human actions with our system and filled the Object Container and the Functionality Graph with the appropriate information.

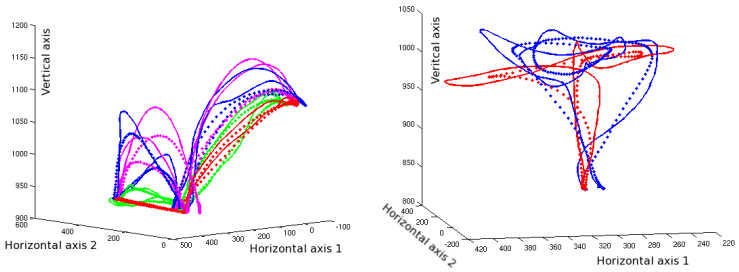



Figure 2.4: Tracking of a line and curve motions of a milk carton and arbitrary motions of a cup

For the rotation classification, a leave-one-out cross validation is made. The results of the classification show, that 42 of 45 of the movements without rotation are correctly labeled, and 30 of 45 movements with rotation are correctly classified. The statistical measures show the learned capabilities in detail. A basic measure is the accuracy, which gives the percentage of correctly classified sequences among all sequences. This measure takes not into account, that there are different numbers of sequences for each class. We refer to our publication in the appendix for numerical examples of achieved accuracies.

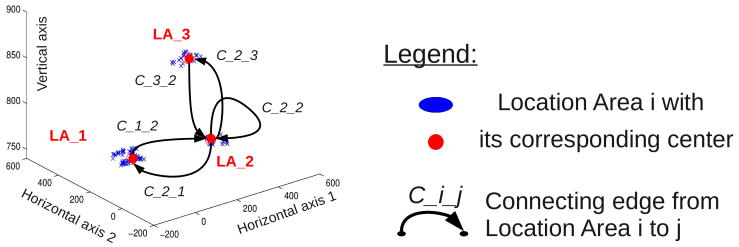



Figure 2.5: Result of the Location Areas - tracking data. The blue crosses show the computed stop-points of all sequences (in mm). The black arrows are drawn to visualize the identified connections between the Location Areas.

An example showing a real scenario and the tracked and segmented data is depicted in Fig. 2.6.

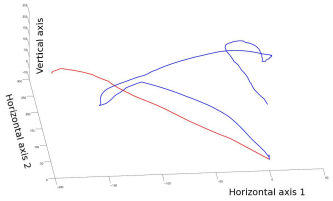



Figure 2.6: Trajectories of movements - vision data. Left: Examples of the trajectories, which are recorded with the vision system. Right: The trajectory of an arbitrary movement is drawn after the recording.

## 2.2 Object Registration

We implemented an efficient algorithm for 3D object recognition in presence of clutter and occlusions in noisy, sparse and unsegmented range data. The method uses a robust geometric descriptor, a hashing technique and an efficient RANSAC-like sampling strategy. We assume that each object is represented by a model consisting of a set of points with corresponding surface normals. Our method recognizes multiple model instances and estimates their position and orientation in the scene. The algorithm scales well with the number of models and its main procedure runs in linear time in the number of scene points. Moreover, the approach is conceptually simple and easy to implement. Tests on a variety of real data sets show that the proposed method performs well on noisy and cluttered scenes in which only small parts of the objects are visible (Fig. 2.7).

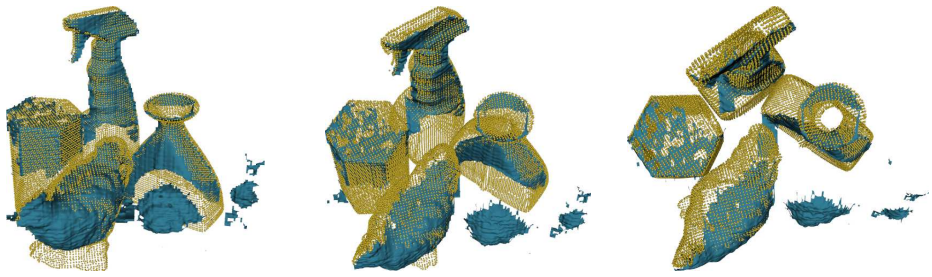


Figure 2.7: Three views of a typical recognition result obtained with our method. The scene is shown as a blue mesh and the four recognized model instances are rendered as yellow point clouds and superimposed over the scene mesh.

### 2.2.1 Model Preprocessing Phase

For a given object model $\mathbf{M}$, we sample all pairs of oriented points $(\mathbf{u}, \mathbf{v}) = ((\mathbf{p}_u, \mathbf{n}_u), (\mathbf{p}_v, \mathbf{n}_v)) \in \mathbf{M} \times \mathbf{M}$ for which $\mathbf{p}_u$ and $\mathbf{p}_v$ are approximately at a distance $d$ from each other. For each pair, the descriptor $f(\mathbf{u}, \mathbf{v}) = (f_2(\mathbf{u}, \mathbf{v}),\ f_3(\mathbf{u}, \mathbf{v}), f_4(\mathbf{u}, \mathbf{v}))$ is computed and stored in a three-dimensional hash table. Note that since $d$ is fixed we do not use $f_1$ as part of the descriptor. Furthermore, we do *not* consider all pairs of oriented points, but only those which fulfill $\|\mathbf{p}_u - \mathbf{p}_v\| \in [d - \delta_d, d + \delta_d]$, for a given tolerance value $\delta_d$. This has several advantages. The space complexity is reduced from $O(m^2)$ to $O(m)$, where $m$ is the number of points in $\mathbf{M}$. For large $d$, the pairs we consider are wide-pairs which allow a much more stable computation of the aligning rigid transform than narrow-pairs do. A further advantage of wide-pairs is due to the fact that the larger the distance the less pairs we have. Thus, computing and storing descriptors of wide-pairs leads to less populated hash table cells which means that we will have to test less transform hypotheses in the online recognition phase and will save computation time.

Note, however, that the pair width $d$ can not be arbitrary large due to occlusions in real world scenes. For a typical value for $d$, there are still a lot of pairs with similar descriptors, i.e., there are hash table cells with too many entries. To avoid this overpopulation, we remove as many of the most populated cells as needed to keep only a fraction $K$ of the pairs in the hash table (in our implementation $K = 0.1$). This strategy leads to some information loss about the object shape. We take this into account in the online phase of our algorithm.

The final representation of all models $\mathbf{M}_1, \ldots, \mathbf{M}_q$ is computed by processing each $\mathbf{M}_i$ in the way described above using *the same* hash table. In order not to confuse the correspondence between pairs and models, each cell contains a list for each model which has pairs stored in the cell. In this way, new models can be added to the hash table without recomputing it.

### 2.2.2 Time Complexity

The complexity of the proposed algorithm is dominated by three major factors: (i) the number of iterations, (ii) the number of pairs per hash table cell and (iii) the cost of evaluating the acceptance function for each object hypothesis. In the following, we discuss each one in detail.

**(i)** Consider the scene $\mathbf{S}^*$ consisting of $|\mathbf{S}^*| = n$ points and a model instance $\mathbf{M}$ therein consisting of

$|\mathbf{M}| = m$ points. We we need

$$N = \frac{\ln(1 - P_S)}{\ln(1 - P_M)} \tag{2.1}$$

iterations to recognize $\mathbf{M}$ with a predefined success probability $P_S$, where $P_M$ is the probability of recognizing $\mathbf{M}$ in a single iteration. In the classic RANSAC applied to 3D object recognition we have $P_M \approx 1/n^3$. Our sampling strategy and the use of the model hash table lead to a significant increase of $P_M$ and thus to a reduction of the complexity. In the following, we estimate $P_M$.

Let $P(\mathbf{p}_u \in \mathbf{M}, \mathbf{p}_v \in \mathbf{M})$ denote the probability that both points are sampled from $\mathbf{M}$. Thus, the probability of recognizing $\mathbf{M}$ in a single iteration is

$$P_M = KP(\mathbf{p}_u \in \mathbf{M}, \mathbf{p}_v \in \mathbf{M}), \tag{2.2}$$

where $K$ is the fraction of oriented point pairs for which the descriptors are saved in the model hash table. Using conditional probability and the fact that $P(\mathbf{p}_u \in \mathbf{M}) = m/n$ we get

$$P_M = (m/n)KP(\mathbf{p}_v \in \mathbf{M} | \mathbf{p}_u \in \mathbf{M}). \tag{2.3}$$

$P(\mathbf{p}_v \in \mathbf{M} | \mathbf{p}_u \in \mathbf{M})$ is the probability to sample $\mathbf{p}_v$ from $\mathbf{M}$ given that $\mathbf{p}_u \in \mathbf{M}$. We can assume that $\mathbf{p}_v$ is not independent of $\mathbf{p}_u$ because it is sampled uniformly from the set $\mathbf{L}$ consisting of the scene points which lie on the sphere with center $\mathbf{p}_u$ and radius $d$, where $d$ is the pair width used in the offline phase. Under the assumptions that the visible object part has an extent larger than $2d$ and that the reconstruction is not too sparse, $\mathbf{L}$ contains points from $\mathbf{M}$. Thus, $P(\mathbf{p}_v \in \mathbf{M} | \mathbf{p}_u \in \mathbf{M}) = |\mathbf{L} \cap \mathbf{M}|/|\mathbf{L}|$ is well-defined and greater than zero. $|\mathbf{L} \cap \mathbf{M}|/|\mathbf{L}|$ depends on the scene, i.e., it depends on the extent and the shape of the visible object part. Estimating $C = |\mathbf{L} \cap \mathbf{M}|/|\mathbf{L}|$ by, e.g., $1/4$ (this is what we use in our implementation) accounts for up to 75% outliers and scene clutter. Thus, we get for $P_M$ as a function of $n$ (the number of scene points)

$$P_M(n) = (m/n)KC. \tag{2.4}$$

Again, approximating the denominator in (2.1) by its Taylor series $\ln(1-P_M(n)) = -P_M(n)+O(P_M(n)^2)$ we get for the number of iterations

$$N(n) \approx \frac{-\ln(1 - P_S)}{P_M(n)} = \frac{-n \ln(1 - P_S)}{mKC} = O(n). \tag{2.5}$$

This proves that the number of iterations depends linearly on the number of scene points. Furthermore, it is guaranteed that the model instances will be recognized with the desired probability $P_S$.

**(ii)** The number of pairs per hash table cell depends on the number of models as well on the number of points of each model. An algorithm is considered to scale well with the number of models if its runtime is less than the sum of the runtime needed for the recognition of each model separately. In other words, an algorithm should need less time than it is needed for a sequential matching of each model to the scene. The use of the model hash table ensures this in the case of our method. For almost all real world objects it holds that a hash table cell does not store pairs from all models. Furthermore, not all pairs originating from a model end up in the same hash table cell.

**(iii)** The acceptance function $\mu$ runs in $O(l)$ time, where $l$ is the number of model points. Note that $\mu$ does not depend on the number of scene points since back projecting a model point in the range image is performed in constant time.

## 2.3 Experimental Results for Scene Occlusions

**Comparison with spin images and tensor matching** In the first test scenario, we compare the recognition rate of our algorithm with the spin images and the tensor matching approaches on occluded real scenes. We test our method on the same 50 data sets as used in other approaches mentioned in the appendix. This allows for a precise comparison without the need of re-implementing neither of the two algorithms. The models of the four toys to be recognized are shown in the upper row of Fig. 2.8. Each test scene contains the toys (not necessary all four of them) in different positions and orientations. Each scene is digitized with a laser range finder from a single viewpoint which means that the back parts of the objects are not visible. Furthermore, in most scenes the toys are placed such that some of them occlude

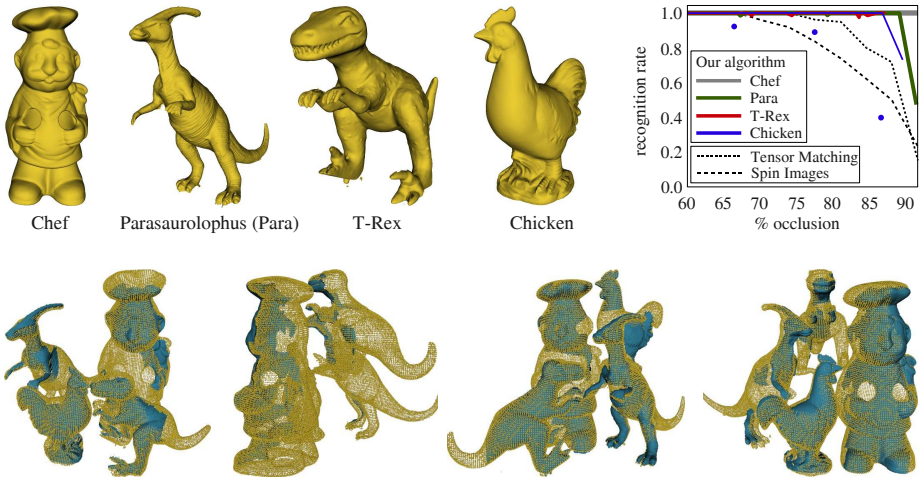



Figure 2.8: (Upper left) The models used in the comparison test case. (Upper right) The continuous lines indicate the recognition rate of our algorithm for each object as a function of its occlusion. The dashed lines give the recognition rate of the spin images and the tensor matching approaches on the same scenes. Note that our method outperforms both algorithms. The chef is recognized in all trials, even in the case of occlusion over 91%. The blue dots represent the recognition rate in the three chicken test scenes in which our method performs worse than the other algorithms. This is due to the fact that in these scenes only the chicken's back part is visible which contains strongly varying normals which makes it difficult to compute a stable aligning transform. (Lower row) Four (out of 50) test scenes and the corresponding recognition results. The recognized models are rendered as yellow point clouds and superimposed over the scenes which are rendered as blue meshes. These are challenging examples since only small parts of the objects are visible.

others which makes the visible object parts even smaller. The lower row of Fig. 2.8 shows exemplary four (out of 50) test scenes with the corresponding recognition results obtained with our algorithm. Since our algorithm is a probabilistic one we run 100 recognition trials on each scene and compute the recognition rate for each object represented in the scene in the following way. We visually inspect the result of each of the 100 trials. If object $\mathbf{A}$ was recognized $n$ times ($0 \leq n \leq 100$) then the recognition rate for $\mathbf{A}$ is $n/100$. Since the occlusion of every object in each scene is known we report the recognition rate for each object as a function of its occlusion. The occlusion for an object model is given by $1 - \frac{\text{area of visible model surface}}{\text{total area of model surface}}$. The results of the tests and the comparison with the spin images and the tensor matching approaches are summarized in the upper right part of Fig. 2.8.

**Noisy and Sparse Scenes** In the second scenario, we run tests under varying noisy conditions. The models to be recognized are the same as in the last test case and the scene is the third one in the lower row of Fig. 2.8. Next, several versions of the scene are computed by degrading it by zero-mean Gaussian noise with different variance values $\sigma$. Again, we perform 100 recognition trials for each noisy scene and compute the recognition rate, the mean number of false positives and the mean RMS error as functions of $\sigma$. For a point set $\mathbf{P}$, a (rigidly) transformed copy $\mathbf{Q}$ and a (rigid) transform $T$ the RMS error measures how close each point $\mathbf{p}_i \in \mathbf{P}$ comes to its corresponding point $\mathbf{q}_i \in \mathbf{Q}$ after transforming $\mathbf{Q}$ by $T$. Thus RMS measures the quality of $T$. It is given by

$$\text{RMS}(T) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|\mathbf{p}_i - T(\mathbf{q}_i)\|^2}, \tag{2.6}$$

where $N$ is the number of points in $\mathbf{P}$. Since we know the ground truth location of each model in the test scene the RMS error of the rigid transform computed by our method can be easily calculated. The results of all noise tests are summarized in Fig. 2.9(a) – (c). Typical recognition results and four of the noisy scenes are shown in Fig. 2.9(d).

Next, we demonstrate the ability of our method to deal with data sets corrupted by noise which is not artificially generated but originates in scan device imprecision. We use a low-cost light section based scanner which gives sparser and noisier data sets. The models used in this test scenario are shown in Fig. 2.10. Typical recognition results of our method are shown in Fig. 2.7 and Fig. 2.11.

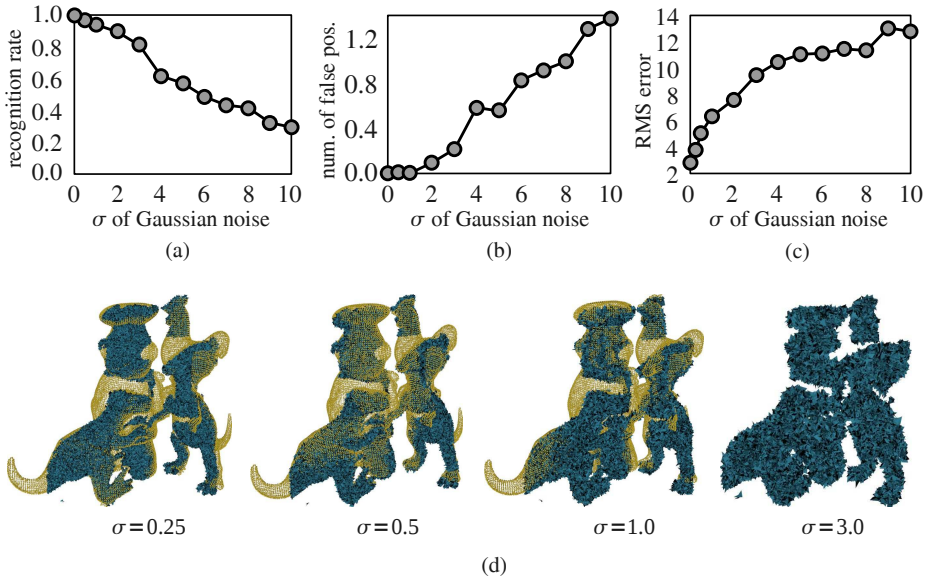



Figure 2.9: (a) - (c) Recognition rate, mean number of false positives and mean RMS error as functions of the $\sigma$ of Gaussian noise. One $\sigma$ unit equals 1% of the bounding box diagonal length of the scene. The RMS units are in millimeters. (d) Typical recognition results for noise degraded data sets.

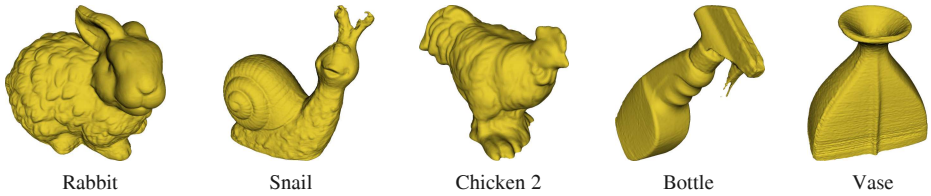



Figure 2.10: The models used for object recognition in scenes reconstructed with a low-cost light intersection based device.

**Runtime** In the last test scenario, we experimentally verify the two main claims regarding the time complexity of our algorithm, namely that it needs less time than it is required for a sequential matching of each model to the scene and that it has a linear complexity in the number of scene points.

First, we measure the runtime dependency on the number of models. The models used in this test case are the ones shown in Fig. 2.8 and Fig. 2.10 and the scene is the leftmost one in Fig. 2.11. The recognition time for each object (when it is the only one loaded in the hash table) is reported in Fig. 2.12(a). In Fig. 2.12(b), the computation time of our algorithm as a function of the number of models loaded in the hash table is compared with the time needed for a sequential matching of each model to the scene. The difference in the performance is obvious.

Second, we measure how the runtime depends on the number of scene points. There are eleven different data sets involved in this test case — a subset from the scenes used in the comparison test case. It is important to note that we do *not* take a single data set and down/up-sample it to get the desired number of points. Instead we choose eleven *different* scenes with varying scene extent, number of points and number of objects. This suggests that the results will hold for arbitrary scenes. We report the results of this test in Fig. 2.12(c).

### 2.3.1 Deformable Registration

We implemented also a new method for fast and robust deformable registration of 3D shapes. Our algorithm is applicable to all kind of shape representations which consist of a finite set of points (which we will call particles) with a neighborhood structure. Examples include range images, meshes and volumetric grids, just to name a few. In contrast to most existing methods which convert the registration problem to a minimization of a cost function, we formulate a new ordinary differential equation (ODE) which models the deformation of a source shape towards a target shape. Integrating the ODE with a simple numerical

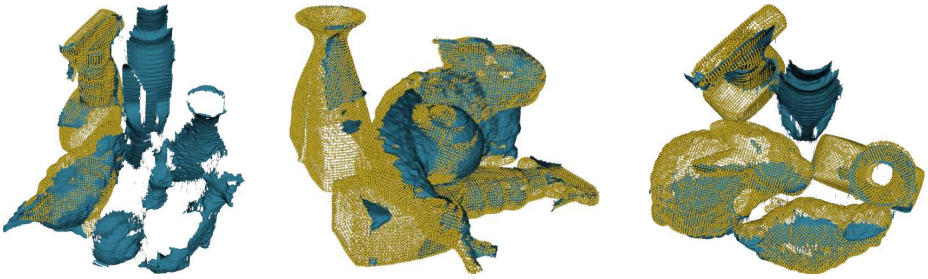
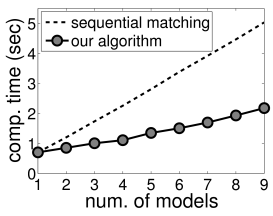


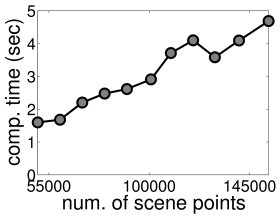
Figure 2.11: Typical recognition results obtained with our method for three test scenes. The scenes are shown as blue meshes and the recognized model instances are rendered as yellow point clouds and superimposed over the meshes. Some of the scenes contain unknown objects (the left and the right one). Note that the scene reconstruction contains only small portions of the objects.

| model | comp. time (sec) |
|---|---|
| Chef | 0.568 |
| Para | 0.533 |
| T-Rex | 0.5 |
| Chicken | 0.522 |
| Rabbit | 0.536 |
| Snail | 0.546 |
| Chicken 2 | 0.551 |
| Bottle | 0.577 |
| Vase | 0.566 |

(a)



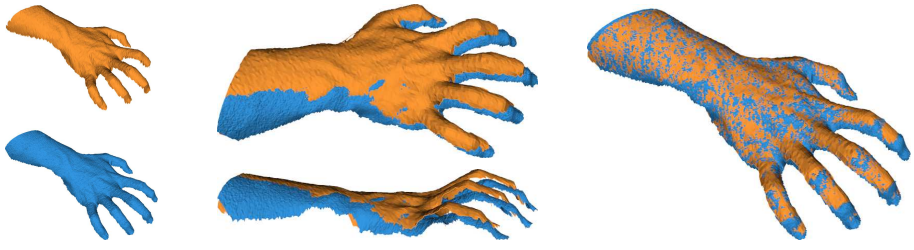

(b)




(c)

Figure 2.12: (a) Recognition time for each model. (b) Computation time for a simultaneous recognition of multiple objects (solid line) compared to a sequential matching of each model to the scene (dashed line). The runtime in the case of the sequential matching is the sum of the times reported in (a) for each model. (c) Linear time complexity in the number of scene points for the simultaneous recognition of 9 models.

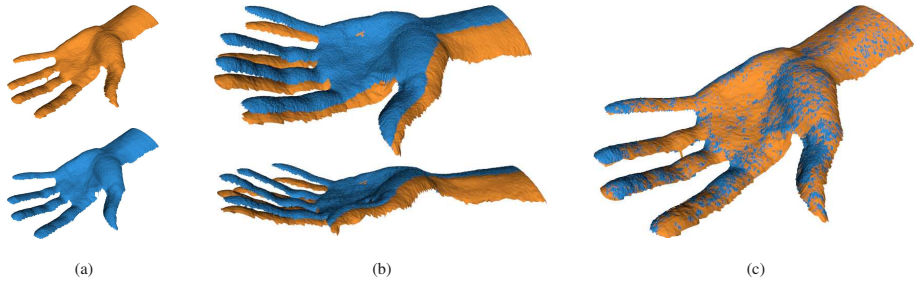
scheme avoids the repeated solution of (usually very) large linear systems necessary for the minimization of non-linear functions. The ODE consists of two terms. The first one causes the deformation by pulling the source shape particles towards positions on the target shape which are determined by nearest-point search. The second term regularizes the deformation by drawing the particles towards locally-defined rest positions. These are determined by rigidly matching the undeformed neighborhood of each source particle to its current (deformed) neighborhood. The numerical solution of the ODE defines a trajectory for each particle from its initial position to its end position on the target shape. We experimentally demonstrate the efficiency of our method in terms of processing time on a variety of real range images. We used hands in our example because they allow the easiest controlled deformation. The system is designed to work with arbitrary object and was also tested on objects from the project.



(a) (b) (c)

Figure 2.13: Range images representing the back part of the same hand in two different poses. The data sets were obtained with a fast 3D geometry scanner (see paper in the appendix). (a) The source shape is shown in yellow (top) and the target shape is shown in blue (bottom). (b) Initial pose of the shapes as captured with the scanner. (c) The result of our deformable registration.

We developed an efficient algorithm for deformable registration of 3D shapes. In contrast to many recent methods, our approach is not based on optimization. The minimization of high-dimensional, nonlinear cost functions is computationally very demanding since it involves the repeated solution of large linear systems. Instead, we rely on a simple and effective numerical integration scheme and solve a system



(a) (b) (c)

Figure 2.14: Registering two range images representing the front part of the same hand in two different poses. The data sets are publicly available. (a) The source shape is the yellow mesh (top) and the target shape is the blue one (bottom) (b) Initial pose of the shapes as captured from the scanner. (c) The result of our deformable registration.

of ODEs which models the non-rigid motion of a source shape towards a target shape. The ODE we proposed consists of an attraction and a regularization term. The attraction term is based on closest-point computations and the regularizer is inspired from deformation modeling techniques recently proposed in the computer graphics community. Furthermore we provided important implementation details for the case of range image registration. We have proven that the proposed method performs well on noisy and incomplete data.

This method can be used in the project to generalize shapes in the Atlas. The method can detect similarities and deformations of them providing hints how to manipulate a novel object based on the stored handling data for the generic representation in the Atlas.

## 2.4 Prediction and Verfication of Physical Object Properties

We started work on evaluation of physical attributes of objects at TUM that cannot be observed without a physical interaction with the object.

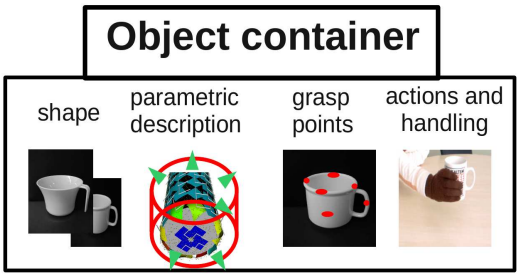



Figure 2.15: The Object Container stores geometric, physical, and handling properties.

The Fig. 2.15 will be extended by physical properties of the object like mass, center of mass, inertia, handling forces for grasp points etc. We try to estimate the parameters through an initial excitation of the object through the robot that observes the response of the object this response is compared with a prediction from the Open Dynamics Engine that allows us to estimate the physical parameters of the object from observation. The Object Container stores the possible ranges for the parameters to be estimated in this active interaction step and reduces the parameter space for the search for the physical attributes of the model.

Our first experiments allow us to estimate the mass and mass distribution properties of an object based on simple excitation of the object which do not require any grasping.

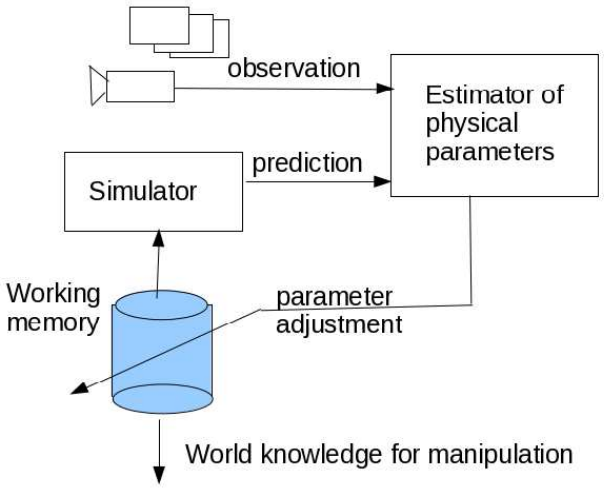



Figure 2.16: Physical object properties in the Object Container are adjusted based on observations of real world interaction.

# Appendix A

# List of Publications

Susanne Petsch and Darius Burschka. Estimation of Spatio-Temporal Object Properties for Manipulation Tasks from Observation of Humans. IEEE International Conference on Robotics and Automation (ICRA 2010), Anchorage, 2010

Susanne Petsch and Darius Burschka. Representatuion of Manipulation-Relevant Object Properties and Actions for Suprise-Driven Exploration. In submission to IROS 2011

Chavdar Papazov and Darius Burschka. An Efficient RANSAC for 3D Object Recognition in Noisy and Occluded Scenes. In Proceedings of the 10th Asian Conference on Computer Vision (ACCV'10), November 2010. (oral presentation; acceptance rate: 4.7

Oliver Ruepp, Robert Bauernschmit, and Darius Burschka. Towards On-Line Intensity-Based Surface Recovery from Monocular Images, British Machine Vision Conference, BMVC 2010.

Chavdar Papazov and Darius Burschka. Stochastic Global Optimization for Robust Point Set Registration. Invited paper submitted to Computer Vision and Image Understanding (CVIU) Journal.

Oliver Ruepp and Darius Burschka. Fast recovery of weakly textured surfaces from monocular image sequences. In Proceedings of the 10th Asian Conference on Computer Vision (ACCV'10).

Matthew Johnson-Roberson, Jeannette Bohg, Gabriel Skantze, Joakim Gustafson, Ralf Carlsson, Babak Rasolzahdeh and Danica Kragic. Enhanced Visual Scene Understanding through Human-Robot Dialogue. In submission to IROS 2011

# Estimation of Spatio-Temporal Object Properties for Manipulation Tasks from Observation of Humans

Susanne Petsch and Darius Burschka

***Abstract*— We propose a system for vision-based estimation of manipulation-relevant properties of objects in natural scenes based on observation of human actions. The system consists of an a-priori *(Atlas)* knowledge about known generic objects in the scene and classifies the scene into mission relevant objects and background geometry that is important only for collision avoidance. We present the structure of our system consisting of an *Atlas* representation and a *Working Memory* storing the current knowledge about the scene, the manipulated objects and actions applied to them in the local environment.**

**We present experimental results how the system maintains the information in the database and show the quality of the results that can be obtained with our system.**

## I. Motivation

Cognitive systems need to be capable of identifying the mission relevance and of learning the model description of objects by themselves during a joint action with a human operator. Most generally, a model of context specifies the entities to observe, the properties to measure and the relations to detect according to [17]. Dey [6] proposed an operational model for context aware perception. In this model, a situation is defined as a configuration of entities and relations relative to a task. The task serves to determine which entities and relations are of interest and should be observed. We transfer these findings into our environment representations, which allows to decouple complex object recognition loops from the low level 3D reconstruction.

Sensation and perception are key components of cognitive systems. Cognition can be defined as "generation of knowledge on the basis of perception, reasoning, learning and prior-models". Perception is the main source of information for reasoning and learning capabilities. Scene classification is an important task in cognitive systems. It helps in sensor-based 3D model generation to discriminate between objects interesting for missions (*foreground*) and *background* objects relevant merely for localization and obstacle avoidance.

We aim to develop a system that defines its actions as a response to the perception under consideration of its knowledge about the current action context. A cognitive system is one that is capable of interacting with humans and other systems in an environment and that is capable to respond to an unexpected event that we will refer to as a *surprise* in the following text. Our system uses the *surprise* to control the learning about the scene and to trigger its own actions as responses to the external stimuli in the environment. We use this to allow the system to deal with a possible high complexity of the scene. Our system observes a human operator who identifies the mission relevant objects through a direct interaction with them (manipulation). This way, our system does not need to identify and to learn about all objects in the scene but only about the objects that were used by the human. These objects define the *foreground* layer of our representation while the geometrical model of the entire scene remains as a global three-dimensional structure in a *background* layer. Only a contact of a human hand with an object followed by a change of its position renders the action as something that the system should know about (Fig. 1).

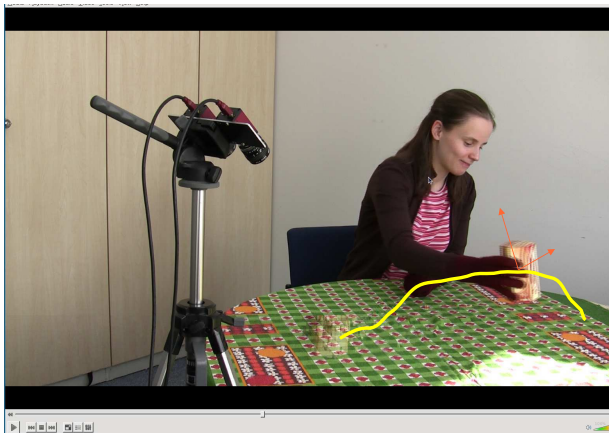


Fig. 1: System observes human actions and completes the internal knowledge representation for objects relevant for a manipulation task.

The target selection task is a challenging part of the system and can be implemented as a manual or automatic process. Examples in 2D image space are described in [14], [13] in more detail. Interesting targets like single standing objects in the scene need to be separated from the supporting planes of the floor and walls that are merely relevant for collision avoidance.

Single standing objects are categorized as *foreground*. They need to be separated from the environment structure (*background*) first. In an additional step, the remaining *foreground* objects are classified according to their shape, extension and movement relative to the scene. The *background* structures are used in a subsequent classification process to

This work was supported by the European Communitys Seventh Framework Programme FP7/2007-2013 under grant agreement 215821 (GRASP project)

Susanne Petsch and Darius Burschka are with the Machine Vision and Perception Group, Department of Informatics, Technische Universität München. 85748 Garching, Germany
{petsch|burschka}@in.tum.de

classify the scene structure according to the criteria described above.

We consider the visual and haptic perception as the stimuli generating the input for our cognitive processing. This multi-modal sensor input will allow to extract the initial information about *foreground objects* in the scene, to classify them, and to match them to already known representations in the *Atlas (long-term memory)* (Fig. 2). The visual observation is a first step to acquire an initial guess about the object properties from its appearance.

The paper is structured as follows: in the next section we present details of our approach. We present the way how the a-priori and working knowledge about the actual environment is represented and how the processing of the robot is implemented. In Section III, we present our experimental results showing the different steps of the processing chain. We conclude in Section IV with an evaluation of the current system and present our future work in this area.

### A. Related Work

Modayil and Kuipers [10], [11] developed a method where a learning agent can autonomously learn about object models, by detecting, tracking, and characterizing clusters of foreground pixels in the sensory stream. Their agent is a mobile robot that receives a stream of sensory information from a laser range-finder. Grauman and Darrel [7] learned feature masks for object categories by embedding sets of unordered image features into a space where they cluster according to their partial-match correspondences. Weber et al. [16] focused on learning object models that are represented as flexible constellations of rigid parts. Savarese and Fei-Fei [12] proposed a model to represent and learn generic 3D object categories by linking together diagnostic parts of the objects from different viewing points. All these methods learn models for particular objects or object categories from a database of static images under different viewpoints and different backgrounds. Our approach works in 3D space providing a more robust segmentation and registration performance.

In the field of object tracking, Comaniciu et al. [5] proposed a kernel-based tracking algorithm where an object is represented by an ellipsoidal region in the image and the mean-shift tracker maximizes the appearance similarity iteratively. Isard and Blake [8] presented a particle filter based tracking algorithm where object shape is represented by B-splines. Yilmaz et al. [19] proposed a contour-based tracking method using the color and texture models in a band around the objects boundary. Tran and Davis presented a robust object tracking method using regional affine invariant features [15]. In our approach, we use our previously presented 6DoF system VGPS that tracks the structure in monocular images and provides in real-time all six motion parameters.

## II. Approach

The robot needs to know about the geometric and physical properties of the object to perform a successful manipulation. Hypotheses about the possible grasp points for the robotic manipulator need to be generated based on the shape and the physical properties like mass and friction of an object. The properties that we currently consider as important for a successful manipulation are: mass, center of gravity, shape to find appropriate surfaces for a successful grasp with a given manipulator, and allowed actions that can be applied to an object. Not all of these properties are observable with a camera and, therefore, we use an additional information database *Atlas* in our system (Fig. 2) to represent the "experience" (*a-priori information*) of the system.

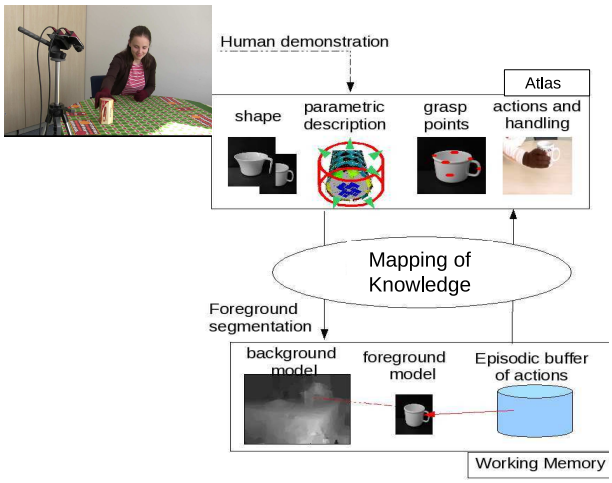



Fig. 2: The system moves the knowledge from a-priori database (Atlas) and instantiates it in the Working Memory representing the actual setup of the manipulation task.

We use for the knowledge representation in the Atlas an analogy to the cognitive capabilities of the human brain and its different strategies, how to store and process the information in the most efficient way. The brain does not store memories in one unified structure. Instead, different types of memory are stored in different regions of the brain. *Long-term memory* in the brain is memory that can last as little as a few days or as long as decades. It differs structurally and functionally from *working memory* or short-term memory, which stores items for only a short time. Working memory (also referred to as short-term memory, depending on the specific theory) is a theoretical construct within cognitive psychology that refers to the structures and processes used for temporarily storing and manipulating information. There are numerous theories as to both the theoretical structure of working memory as well as to the specific parts of the brain responsible for working memory. Baddeley and Hitch (1974) introduced and made popular the multi-component model of working memory [1].

We follow the structure suggested by Baddeley with the long-term memory and the short-term memory maintained by the central executive (Mapping of the Knowledge in Fig. 2). Our system consists of two databases storing a-priori knowledge about the world (*the Atlas*) corresponding to the long-term memory and a *Working Memory* representing

the current visual an spatial representation of the world (visuospatial sketchpad). In this layer, the episodic buffer is implemented as a system storing the typical actions applied to a mission relevant object.

The two layers (Fig. 2) have the following representation:

- **Atlas Representation (Experience of the System)** - this information represents a-priori knowledge given to the system from an expert or representations of the environment collected in previous operations in the same or similar environment. An important difference of the proposed system to many other systems suggested before is that it is supposed to interact with its environment in a cognitive way. This means that the system does not operate based on a set of pre-defined rules but it tries to learn from its own actions and actions of other agents in the environment (human or other robots). The information stored in the Atlas represents a generic knowledge about a class of object.
- **Working Memory**- Working memory is a theoretical construct within cognitive psychology that refers to the structures and processes used for temporarily storing and manipulating information. In our system, the *experience* needs to be grounded to a given environment. We expect to operate in highly complex environments, where the system must not try to analyze all elements of the scene as it is often the case in other current manipulation systems but it needs to focus its *attention* on mission relevant objects whose properties need to be explored for a successful interaction with the world.

An important novelty in the presented system is that the objects are represented not only with their spatial and physical properties (shape, mass, friction) but include also temporal handling information which is essential for the system to handle the object with the same constraints regarding its orientation relative to the gravity vector and accelerations in the translational and rotational motions as presented by the human. The following processing chain allows us to extract this information from the visual system of the robot.

Our system depicted in Fig. 3 contains the entire processing chain for the visual interpretation of a human action. It starts with the detection of candidates for mission-relevant objects in the world using in our first implementation a simple Supporting Plane Removal algorithm presented already in [2], [4]. In the next step, we use our Vision Interaction Cues (VICs) approach [18] to speed up the processing of human actions. In our system, the human triggers any new knowledge acquisition by presenting new actions to the system. Each cluster segmented in the initial segmentation step defines an *interaction space* where gestures are actually analyzed. It is only necessary to do it if the hand is in the vicinity of a given object. Once a grasp gesture at a given cluster is detected the system starts tracking the 6DoF pose of the object to understand the action performed by the user. It stores the corresponding trajectory for later analysis until the object is released. In a final step, a registration step is performed to match the given cluster to the known

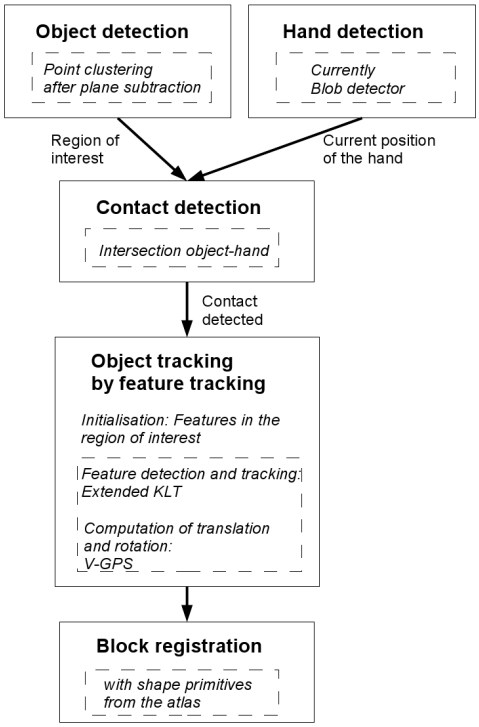



Fig. 3: Overview of the approach. The boxes represent the single modules of the system. Each box contains a dashed box, where the implementations can be found, which are used for the realization of the modules here.

geometries in the *Atlas* using the shape representations stored in there. The system has the choice to use direct shape registration for known objects or parametric shape analysis to categorize the shape to a specific generic class representation in the *Atlas*.

### *A. Knowledge Representation*

We can tell from Fig. 2 that the *Atlas* contains several distinctive object representations that provide information which is important for the recognition of an object (geometric shape for direct 3D shape registration, and parametric shape description for generalized object class representation) and additional information which is important to initialize parameters which are not observable by the system. These additional parameters are mass, center of gravity and friction leading to specific grasp point representation, and actions that are known to be associated with a given object (e.g., motion constraints on cups or glasses that my contain water).

This a-priori information (*experience*) from the *Atlas* needs to be mapped on the current environment representation surrounding the robot, which is stored in the *Working Memory*. The *Working Memory* contains the geometric shape description as well which is now complete in opposite to the current sensor reconstruction that usually provides only a partial view due to occlusions in the scene. The registration step to the *Atlas* information allows a completion here. Additionally, now the system is able to store also the texture information representing the appearance of an actual instance

of an object in the scene. Now we know not only that there is e.g., a cup, but we also know that this is a cup with a specific texture or logo on it. We move the initial hypothesis about the grasping points and actions from the *Atlas* to the *Working Memory*. Finally, we get also hypotheses about the mass range, center of gravity position, friction and stiffness of the object as an initial guess for the first interaction of the robot with the object. This information is provided as a container for other processing steps and not considered in this paper.

### B. Action Representation

An important novelty in our object description is the representation of the temporal changes to the object. We decided to use an object-centric representation of actions. We consider the robot and the human as agents that can imply changes to the state of an object. We are interested in this context only in three phases of the change depicted in Fig. 4

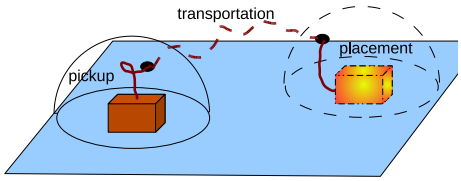



Fig. 4: Three phases defining an action: the type of pickup, the way the transportation is done, and the placement of an object.

The pickup and the placement is mostly concerned with the grasp type performed by the human operator and not part of this paper. This is an information, which is important for an emulation of the grasp by the robot and requires a hand gesture recognition which is provided by a project partner. In this paper, we are interested in the analysis of the transportation phase of the action. It is important for us in this stage of the project, how free the motion of the object can be (which enforces constraints of coupling between the joints of the robot to ensure a specific orientation relative to, e.g., the gravitational vector) is and where the object is usually placed in the scene. We found it not necessary to safe any actual trajectories presented by the human since our focus is on a detailed description of object properties here. For an object it is not important which way it took through the environment but only how it was handled (speeds, orientations) and where it was picked up and placed. This is the information that we need to extract from the vision system.

### C. Scene Clustering

In order to detect the object in the plane, a plane-subtraction is applied first, which is described in [2], [4]. The approach uses the fact that there is a homography between the $(u, v, D)$ coordinates of the disparity image ($[u, v]$-image coordinates and disparity D) and the corresponding Cartesian coordinates from the 3D scene. According to [2], the planar surface $P_r$ can be represented as

$$P_r : a_r x + b_r y + c_r z = d_r. \tag{1}$$

It is shown in [4], that the equivalent disparity plane is given by

$$D(u,v) = \begin{pmatrix}\rho_1\\ \rho_2\\ \rho_3\end{pmatrix} \cdot \begin{pmatrix}u\\ v\\ 1\end{pmatrix} = n_r^* \cdot \begin{pmatrix}u\\ v\\ 1\end{pmatrix} \tag{2}$$

with the disparity $D(u, v)$ at image coordinates $(u, v)$, $\rho_1 = \frac{a_r}{k}$, $\rho_2 = \frac{b_r}{k}$, $\rho_3 = \frac{c_r}{k}$, $k = \frac{d_r}{B}$ and baseline $B$.

The next step is the search for the planar candidates. These candidates depend on the gradient of the disparity-map: A high gradient or low gradients with different directions refer to the border of a planar plane, whereas pixel with low gradients of the same direction form a plane. The biggest area with low gradients of the same direction is assumed to be a part of the plane. This area is used for the estimation of the normal vector $n_r^*$ of the plane. The vector $n_r^*$ is estimated according to (6) in [4]:

$$\begin{pmatrix}\sum u_i \cdot D_i\\ \sum v_i \cdot D_i\\ \sum D_i\end{pmatrix} = \begin{pmatrix}\sum u_i^2 & \sum u_i v_i & \sum u_i\\ \sum u_i v_i & \sum v_i^2 & \sum v_i\\ \sum u_i & \sum v_i^2 & \sum 1\end{pmatrix} \cdot n_r^* \tag{3}$$

The direction vector $n_r^*$ enables a comparison between the observed disparity of a pixel and the expected disparity of the plane at the position of the pixel according to the direction vector $n_r^*$ of the plane. If the difference between both is higher than a certain threshold, the pixel is assumed to not belong to the plane. All pixel, which belong to the plane, are deleted in the disparity-map. Consequently the objects, which are placed on the plane, remain in the disparity-map. An example of the plane subtraction is given in Fig. 5.

Fig. 5: Results of plane subtraction. *Left column:* Original color image. *Middle column:* Disparity image of the color image on its left. *Right column:* Remaining object in the disparity image after the plane subtraction.

For the further processing the outer bounding box of the object as the biggest connected component is taken as region of interest. Any other representation could be applied as well.

### D. Parsing of Human Action

The manipulation of the objects is going to be parsed as follows.

The first step is the detection of the contact of the object and the hand, which will take the object. The position of the object is detected as described before. Since the position of the object is not changing until its contact with the hand, the computation of the position of the object has not to be computed again. The position of the hand can be determined in different ways, a blob-detector is used here. Therefore the

color-image is split in HSV-planes and appropriate thresholds are applied. Just the pixel with the color of the hand remain in the image. If the hand touches the region of interest, a contact is detected. Otherwise the procedure for the contact detection is repeated until a contact is detected.

After the detection of the contact between the hand and the region of interest (the object), the tracking of the features of the object is initialized. The features are selected in the region of the object. If an outer bounding box is used as region of interest, there will be features, which are not on the object and cannot be used for the tracking of the object. The positions of these features do not contain disparity after the plane-subtraction and the features can be deleted. The valid features on the object are used for the tracking of the manipulated object. The contact between the hand and the object is assumed to be lost, when the object and its features are not moving any more. Therefore the tracking of the objects features is finished when all features stop moving. The extended KLT [9] is used here for feature detection and tracking. An example of the contact detection and the tracked features is given in Fig. 6.

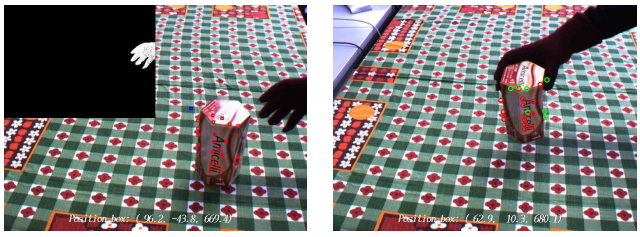


Fig. 6: Contact detection and object tracking. *Left:* The tracking is initialized after the contact detection between the hand and the region of interest. The small red boxes are the valid features, whereas the blue ones are the deleted features, which are not on the box. The top left corner of the image shows the position of the hand, when the contact occurs. *Right:* Example of features during the tracking. The tracked features are shown in red, the assumed position of the lost features are drawn in green.

The recorded trace of the tracked features of the manipulated object enables the computation of its rotation and translation. V-GPS is used for the computation of the rotation and translation [3]. The computed angles and the translation during the movement determine the possible movements of the objects. Additionally the trace of lost features can be reconstructed.

## III. Results

In this section the results of the experiments are presented. The used sequences (seq.) have different motion properties, shown in Table I. The movement was either a straight line or an arbitrary motion, the object was either tilted or not. All movements were tested with two different boxes. The scene was recorded with a Firewire Marlin FO46C camera. The following settings were used: image size = 780x582 pixel (width x height). OpenCV, XVision, extended KLT [9] and V-GPS [3] were used in the algorithm, which was running on a Linux system.

TABLE I: Properties of the used sequences

| Seq.: | Box 1 | Box 2 | Movement | Rotation | # Images |
|---|---|---|---|---|---|
| 1 | x | | line | | 705 |
| 2 | x | | line | x | 740 |
| 3 | x | | arbitrary | | 1055 |
| 4 | x | | arbitrary | x | 1035 |
| 5 | | x | line | | 725 |
| 6 | | x | line | x | 870 |
| 7 | | x | arbitrary | | 860 |
| 8 | | x | arbitrary | x | 1600 |

### A. Clustering of Object Candidates on a Table

The first part of the experiment is the plane subtraction, as described in II-C, in order to get the position of the object as the region of interest. A sliding-average window was used to get a fill holes in the disparity-map, although that results in smoother transitions between an object and the plane. Just one sequence (seq. 6) required the modification of two additional parameters (a larger kernel for the expected size of the ROI and a higher number of neighbors considered for the comparison of the direction of the gradient) because of a too smooth transition between the object and the table, which included the box as a candidate region for background subtraction.

Fig. 7 shows the result of seq. 6. The result of seq. 3 has already been presented in Fig. 5. The ROI is successfully detected for all sequences, the size of the all ROI is given in table II. The boxes used for the experiments have different sizes (box 2 is larger than box 1), therefore, the algorithm computes correctly a larger ROI for box 2.

Fig. 7: Results of plane subtraction (Seq. 6). *Left column:* Original color image. *Middle column:* Disparity image of the color image on its left. *Right column:* Remaining object in the disparity image after the plane subtraction.

### B. Tracking of Human-Induced Motions on the Objects

After the detection of the ROI, the manipulated object is tracked as described in II-D. The tracking is initialized when a contact between the hand and the object is detected. The feature tracking is implemented with extended KLT tracker [9]. The rotation and translation are computed with V-GPS approach [3]. The rotation and translation is also used for the reinitialization of lost features, since the assumed position of the lost feature can be computed from the estimated rotation and translation of the object. The initialization of the tracker and features during the tracking have already been shown in Fig. 6. An example for the

used feature set and the object trajectory is shown in Fig. 8. The trajectory is computed from the position of the tracked features using V-GPS. All sequences show that the trace of a tracked arbitrary feature in the sequence is similar to its projected 3D trajectory.

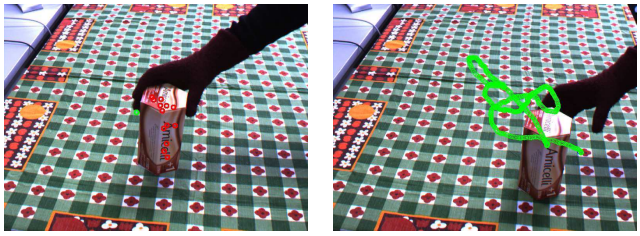


Fig. 8: Tracking of the object (6DoF from monocular) (Seq. 3). *Left:* Example of feature set used for tracking. The tracked features are shown in red, the predicted position of the lost features is drawn in green. *Right:* the object trajectory is shown in green.

Table II contains also the number of features at the beginning and at the end of the sequence. Additionally, the number of features, which were tracked during the whole sequence without a reinitialization, can be seen. The number of features which are tracked during the whole sequence without any reinitialization decreases with the length of the sequence (seq. 3,4,8) and the influence of rotation (seq. 2,4,6,8). The table shows also that the reinitialization of lost features is successful, especially seq. 8.

TABLE II: ROI and number of tracked features

| Seq.: | Size ROI (pixel) | # features: | start | end | whole seq. |
|---|---|---|---|---|---|
| 1 | 27.945 | | 20 | 15 | 7 |
| 2 | 25.488 | | 22 | 13 | 4 |
| 3 | 27.800 | | 14 | 11 | 3 |
| 4 | 20.088 | | 11 | 8 | 2 |
| 5 | 39.026 | | 10 | 9 | 8 |
| 6 | 48.830 | | 16 | 9 | 5 |
| 7 | 52.398 | | 41 | 37 | 30 |
| 8 | 52.832 | | 34 | 21 | 7 |

### C. Analysis of the Trajectories

The calculated information about the rotation and translation of the manipulated object during tracking enables the computation of several properties of the manipulation. As already described in III-B, the computation of the object trajectory is possible. Furthermore, the rotation of the object can be computed at each step as well as the speed of the object along the trajectory. Fig. 9 shows the rotation, translation and speed of the object in seq. 7. The orientation of the vertical axis of the object is drawn every 50 steps. We assume, that at the beginning of the trajectory the object orientation is aligned with the calculated normal vector of the table since we do not use any model information for the object. Moreover, Fig. 9 depicts the shape of the speed curve during manipulation. The speed at each position is the average of the past 50 steps in Cartesian coordinates in the 3D Scene. The speed increases during the task.

Fig. 9: Rotation, translation and speed of the object (Seq. 7). *Left:* The development of the rotated and translated normal vector of the table is shown in green. *Right:* The development of the speed along the trajectory is shown in different colors: green = no movement, yellow = slow movement, red = movement, blue = fast movement.

Fig. 10 shows the rotation, the translation and the speed of the manipulation in other trials. The rotation of the object is visible for seq. 4 and 8 while seq. 5 does not contain any rotation of the object. It is a translation along a straight line. Besides the shown rotations and translations of some sequences, the translation and (if applied) rotation of the objects have been drawn for all sequences. Fig. 10 contains also the shape of the speed during the manipulation of the object. The manipulation of the object in seq. 5 along a straight line shows clearly the increasing speed after the pickup, the (in average) constant speed during the transportation and the decreasing speed before the placement.

The results of the angle analysis between the original position of the object and its rotated position during the manipulation are in Table III. The magnitude of the average angle along the trajectory indicates if the object needs to be kept in a vertical orientation or can be tilted during manipulation. As it can be seen, Seq. 2, 6 and 8, which contain rotations, have a clearly higher average angle than seq. 1, 5 and 7 without rotations. In seq. 3 and 4 the system switched to wrong features during tracking resulting in a bias in angle estimates. As table II shows, the number of tracked features in seq. 3 and 4 is really low, therefore, it is obvious that the computation of the rotation and translation, which is based on these features and their number, is challenging for the complex movements in seq. 3 and 4. The fact that there is a small average angle for seq. 1, 5 and 7, which do actually not contain rotations, is also caused by the human operator, since it is hardly possible to move an object without any rotation at all. The results for the maximum angles between the original position and the rotated position (table III) show a similar result: The maximum angle of seq. 2, 6 and 8, which contain rotations, is much higher than for seq. 1, 5 and 7 without rotations. The remaining angle between the original position and the final position should be close to zero, since the object is placed on the table again. The results in table III show, that there is a relatively high remaining angle in seq. 2 and 6. These sequences have a small number of constantly tracked features, similar to seq. 3 and 4. The remaining angle

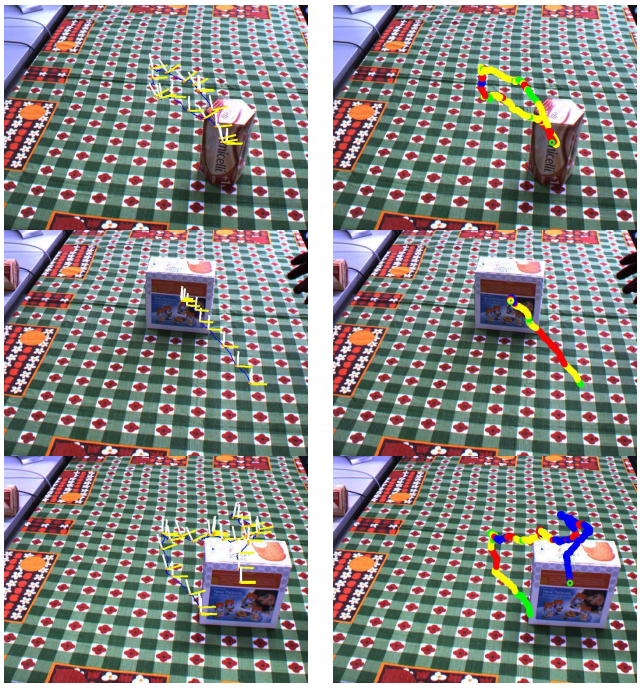


Fig. 10: Rotation and translation of the object in different sequences (Seq. 4, 5, 8). *Left column:* The drawn coordinate system shows the computed rotation and translation of the tracked object. The object trajectory is drawn in yellow. *Right column:* The development of the speed of the object trajectory is shown in different colors: green = no movement, yellow = slow movement, red = movement, blue = fast movement.

of seq. 7 reaches with 0.36 nearly zero. This sequence has the highest number of constantly tracked features among all sequences, therefore it can be concluded that the number of constantly tracked features influences the performance.

TABLE III: Analyzed angles of the sequences

| Seq.: | Average | Maximum | Remaining angle (end) |
|---|---|---|---|
| 1 | 4.28 | 11.11 | 5.54 |
| 2 | 10.36 | 31.99 | 19.69 |
| 3 | 10.08 | 20.72 | 4.57 |
| 4 | 6.60 | 14.23 | 1.83 |
| 5 | 4.40 | 10.87 | 4.86 |
| 6 | 11.30 | 21.50 | 20.07 |
| 7 | 5.47 | 11.28 | 0.36 |
| 8 | 10.33 | 25.99 | 4.08 |

## IV. Conclusions and Future Work

The initial representation developed in the current system is the our testbed how to represent knowledge in a manipulation system and how to define action representations that are necessary for a successful surprise detection. The detection accuracy is already sufficient and will be improved through usage of a bifocal setup in the near future, where the object is observed with a long focal length camera that will allow an even better spatial resolution.

Our next goal is to focus more on the representation of actions in the local environment and to include them in the predictions of the system. We started already work on registration of generic shape descriptions that will allow a classification of objects to a global category. This will allow to provide a-priori suggestion about the manipulation capabilities of an object which may still be unknown to the system.

## References


[1] A.D. Baddeley and G.J. Hitch. Working Memory. *New York: Academic Press, vol. 8*, 1974.

[2] D. Burschka and G. Hager. Scene Classification from Dense Disparity Maps in Indoor Environments. In *Proc. ICPR*, 2002.

[3] D. Burschka and G. Hager. V-GPS – Image-Based Control for 3D Guidance Systems. In *Proc. of IROS*, pages 1789–1795, October 2003.

[4] D. Burschka and G.D. Hager. Vision-Based 3D Scene Analysis for Driver Assistance. *ICRA*, 2005.

[5] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based Object Tracking. *Transactions on Pattern Analysis and Machine Intelligence, vol. 25*, pages 564–577, 2003.

[6] A.K. Dey. Understanding and Using Context. In *Personal and Ubiquitous Computing*, volume 5(1), pages 4–7, 2001.

[7] K. Grauman and T. Darell. Unsupervised learning of categories from sets of partially matching image features. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June, 2006.

[8] M. Isard and A. Blake. CONDENSATION - Conditional Density Propagation for Visual Tracking. *International Journal of Computer Vision, vol. 29, no. 1*, pages 5–28, 1998.

[9] E. Mair, K. Strobl, M. Suppa, and D. Burschka. Efficient Camera-Based Pose Estimation for Real-Time Applications. *IROS*, 2009, to appear.

[10] J. Modayil and B. Kuipers. Bootstrap learning for object discovery. *IROS*, 2004, vol. 1.

[11] J. Modayil and B. Kuipers. Autonomous Shape Model Learning for Object Localization and Recognition. *IEEE Int. Conf. on Robotics and Automation*, 2006.

[12] S. Savarese and F. Li. 3D Generic Object Categorization, Localization and Pose Estimation. *IEEE International Conf. in Computer Vision (ICCV)*, 2007.

[13] C. Schmid, R. Mohr, and C. Bauckhage. Comparing and evaluating interest points. *Proc. of International Conference on Computer Vision*, 1998.

[14] S. Simhon and G. Dudek. Selecting targets for local reference frames. *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 2840–2845, 1998.

[15] S. Tran and L. Davis. Robust Object Trackinng with Regional Affine Invariant Features. *IEEE 11th International Conference on Computer Vision*, pages 1–8, 2007.

[16] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. *Proc. ECCV, vol. 1*, pages 18–32, 2000.

[17] T. Winograd. Architecture of Context. In *Human Computer Interaction*, volume 16, pages 401–419.

[18] Guangqi Ye, Jason Corso, Darius Burschka, and Gregory D. Hager. VICs: A Modular Vision-Based HCI Framework. In *Proceedings of 3rd International Conference on Computer Vision Systems*, pages 257–267, 2003.

[19] A. Yilmaz, X. Li, and M. Shah. Contour-based Object Tracking with Occlusion Handling in Video Acquired using Mobile Camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 11*, pages 1531–1536, 2004.

# Representation of Manipulation-Relevant Object Properties and Actions for Surprise-Driven Exploration

Susanne Petsch and Darius Burschka

***Abstract*— We propose a framework for sensor-based estimation of manipulation-relevant object properties and for abstraction of the known actions in a learning setup from observation of humans. The introduced descriptors consist of a representation constraining the motion of the object during the manipulation task and an action graph spanning between the typical places where the object is placed. This framework allows to abstract the strongly varying actions of a human operator and monitors for unexpected new actions (surprise) that require a modification of the knowledge stored in the system. The usage of a general and object-centric structure enables the transfer of knowledge not only to the same situation, but also to similar environments. Furthermore the information can be derived from different sensing modalities.**

**The proposed system consists of the manipulation-relevant properties, which are directly related to the object (Object Container), and the actions performed with the object in its environment (Functionality Map with Location Areas). We present experimental results on real human actions, showing the quality of the results that can be obtained with our system.**

## I. Motivation

A robot should be able to learn unsupervised through observation of human actions in its environment. Unfortunately, humans do not follow exact trajectories while performing repetitive manipulation tasks in an environment. The system needs to be able to abstract the manipulation actions to focus only on information which is necessary to imitate the manipulation or to cooperate with the human in the given environment. A mismatch between the expectation of the observer system and a current human action, which we will call a surprise event in the following text, should occur only in situations, when the stored information needs to be actually refined or modified. The important examples here are the cases when in the object transport phase suddenly motion constraints are changed (e.g., a cup carried always upright is now tilted arbitrarily) or when an object is suddenly places on an unexpected place, e.g. cup on the floor. This usually is an indication that the physical properties of the object (e.g., level of the liquid in the object) or their function (not a drinking cup but a dirty dish) changed. This needs to be considered in the internal representation of the manipulation system. The robot has to be able to detect new information efficiently in its known environment (surprise detection).

This work was supported by the European Communitys Seventh Framework Programme FP7/2007-2013 under grant agreement 215821 (GRASP project)

Susanne Petsch and Darius Burschka are with the Machine Vision and Perception Group, Department of Informatics, Technische Universität München. 85748 Garching, Germany
{petsch|burschka}@in.tum.de

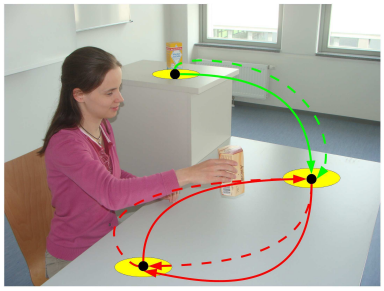


Fig. 1: The system creates an abstract map of different functionalities in the environment.

Our aim is to define a model that allows to map different physical properties of the object to modifications in the way how an object is carried and that efficiently abstracts the known actions applied to a given object to be able to correctly predict the often strongly varying transport trajectories and goals. This second property of the system allows to reason about changes in the function of a specific object in a given environment, e.g. a tool is not used for a specific adjustment but just put away. It is important to consider, that the robot might face different sources of information. One source is of course the observation of human e.g. with a vision system, which can be useful in a household. But there are also other sources of knowledge like a data base, which provides knowledge in another form (e.g. the exact trajectories of actions in a chemical laboratory). Therefore, the system needs to be able to deal with different representations of knowledge.

The model for the stored properties and actions need to abstract from an exact Cartesian motion. An a-priori knowledge about an object class is stored in an *Atlas* introduced in [18]. It contains already known properties of the objects as well as a-priori knowledge about the objects handling. Of course different handling properties might occur for the same object depending on the context. The correct alternative alternatives has to be selected based on the observation. This property may change over time. This modifications are triggered by a mismatch between the expectation and the observation of the human action. The general knowledge can be mapped into the current context and stored in the Working Memory, in order to use information in the current scene.

It is important to consider, that not only the object itself (e.g. its properties or physical states) is defining the way how it is manipulated, but also the locality of the object can change its function in an environment. Different locations in

the environment are used for different actions, which have certain properties. For example washing the dishes is normally done in the sink and not on the flat table without any water source in the neighborhood. The conclusion is, that we need a collection of object properties (Object Container), but also a map of the environment (Functionality Map), which provides the functionalities in this environment (see Fig. 1). It is important to notice, that we are not interested in the exact reconstruction of the environment in the sense of navigation, but in an abstract representation of the functionalities in the environment.

We split the representation of the knowledge about the human actions into an object-centric representation reflecting the physical properties of an object stored in an *Object Container* and a *Functionality Map* representing possible transport relations between places in the environment. While the *Object Container* is linked only to the object, the *Functionality Map* is anchored in to the geometric model of the environment. This framework allows us to limit unexpected events (surprise events) that cannot be explained with the current knowledge to situations where the physical state or the function of an object changed. The system is insensitive to variations in human actions.

Mismatches to predictions based on the information stored in Object Container or the Functionality Map signal new, unknown events, which require an update of the stored information. As already mentioned, the robot should be able to deal with several sources of information. Since our system is built on properties of the object and object's function, we are not relying on a trajectory in a certain representation like x,y,z-coordinates, or on a certain colored pattern of an object. The properties in our system have to be powerful enough to provide information for a manipulation. At the same time, they have to be generic and extractable from different sources.

The goal of this paper is the presentation of a system, which provides the manipulation-relevant knowledge in a manner that the described requirements can be met.

The paper is structured as follows: the detailed approach is presented in the next section. First the manipulation-relevant object knowledge and the Functionality Map of the environment are described, followed by the presentation of the knowledge extraction. The results of the experiments with real human actions are described in Section III. We end with the conclusions and future work in Section IV.

### A. Related Work

A lot of work exists in the field of imitation learning. In [1], HMMs are used for imitation learning of arm movements in manipulation tasks for humanoid robots, in order to achieve a human-like reproduction of the motions. It is important to point out the difference between our approach and non-object-centric approaches. Such approaches are for example the imitation/ learning of motor skills or the imitation of movements with Dynamic Movement Primitives (DMP), which encode the trajectories themselves directly [9], [16]. This paper does not aim to encode the trajectories as, e.g., DMP or the model in [12]. Calinon *et. al* use imitation learning, in order to learn control strategies [3]. Moreover, approaches related to Reinforcement Learning [22] are used for imitation learning, using the observations of humans as reward [23], [2]. In contrast to the described work in the field of imitation learning, our aim goes beyond imitation learning. A more general representation of object properties and their functionality in the environment is provided, in order to get a further understanding of the knowledge in different tasks and environments.

The intention in imitation tasks is addressed by Jansen and Belpaeme [10]. They train their agent in a grid with blocks in a computer simulation. In contrast, this work deals with more complex, real-world environments and the system needs much less training instances than the one presented in [10]. A real-world example of capturing the user's intention about sequential task constraints is presented in [15]. Their system reasons about the existence of sequential dependencies of operations, in contrast to a further understanding of the object's functionality itself in this paper.

In order to achieve a further understanding of the object's functionality, the object's motion has to be analyzed. The work in [13] takes into account the effect on the object. The object properties and its state are also of interest in this paper, but the difference between their approach and ours is to get information about the manipulation properties directly related to the object and, furthermore, to the objects functionality in the environment. Function from motion is analyzed in [5] for "primitive motions", which are translations or rotations relative to the main axes of primitive objects. Our approach goes further to more and more general manipulation-relevant object properties. In [24], functional roles of objects like "pour out" have been explicitly introduced. These roles do not refer to the object's properties, which are directly observable during manipulation. It is important to distinguish the analysis of the object's functionality through the observation of humans in this paper from reasoning about shape descriptions for object functions [21].

Once again, it should be to pointed out, that the reconstruction or the analysis of the environment, like [14], is not of interest, since it focuses on the objects and their functionalities in the environment. The relative/ absolute position of objects to each other have been used [14] for the consideration of the environment in manipulation properties. In [4], a perceptual space (for the color and shape object properties) and a situation space (for the displacement of the objects in the scene) are introduced. In contrast, the object properties in this paper are beyond the pure visual current appearance of the object, since the manipulation-relevant object properties are of interest. Furthermore, the Functionality Map is not dealing with the object's relative position to each other, but it aims to understand the objects and their functionalities in the environment. Additionally, an analysis or a semantic labeling of the whole scene (e.g. [6], [17]) is not our aim. In contrast, just the shape and the position of the relevant manipulated object is determined as possible object properties in the object's environment.

## II. Approach

An overview of the system is given in Fig. 2. It illustrates the *Object Container* with the object properties and the representation of the actions in the *Functionality Map*. The Functionality Map contains the action properties. It represents a graph spanning *Location Areas* between which the actions are performed.

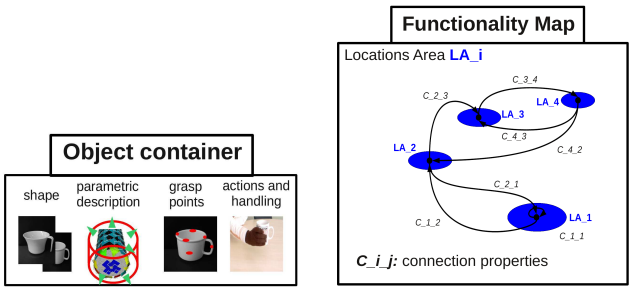



Fig. 2: Object Container and Functionality Map. The Object Container stores the object properties. The Functionality Map is an abstract representation of the manipulation-relevant operating areas in the environment.

### A. Manipulation-Relevant Object Properties

Since we are interested in a general knowledge of the object properties, we do not want to list all the simple records of a trajectory (traj.) in x,y,z-coordinates, but the abstract handling properties. The properties we consider as important for each object are the variation of orientation, the maximal allowed acceleration, the type of grasp allowing a successful grasp with a given manipulator, the mass and the center of gravity. Some of these properties are not observable with e.g. a pure vision-based system or a pure tracking system. Therefore, the already described information database Atlas [18], which contains the "experience" (*a-priori information*), is used to provide initial information. The other properties need to be extracted using for example a vision system.

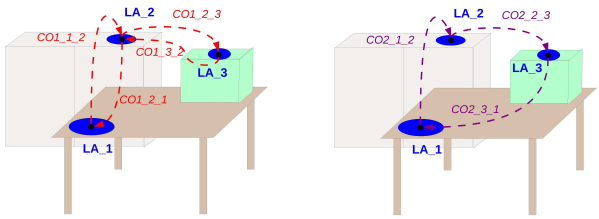
The handling properties themselves are constraints, which limit the handling possibilities of the object in a certain situation. For example an observed rotation of a manipulated object indicates that the object has not to be kept in a fix orientation. Consequently, the object-orientation does not put a constraint on the planning of manipulation in this context.

Our object-centric representation shows here the advantage. The representation of a constraint does not rely on a specific Cartesian position of space but considers only the changes/derivatives of them. Every derivative removes the constant offset from a descriptor letting the resulting parameter float in the coordinate frame of the local area. Possible variations in the actual Cartesian position of the traj. do not modify these descriptors.

### B. Functionality Map of the Environment

The first component of the Functionality Map are the Location Areas. These areas are the locations in the 3D space, where a manipulation sequence for a given object can start or end. We define explicitly location *areas* and not single locations, since an object is usually placed in a certain area and not on one certain point in space. These observed Location Areas have observed connections between them representing the edges of our Functionality Map. A Location Area does not necessarily mean that the manipulated object is standing on a surface. A hand-over step (e.g. changing hands) can also establish a Location Area. The established Location Area is, therefore, not necessarily connected to a surface, but to an area in space.

The connections between different Location Areas are the next component of the Functionality Map. A connection exists between two Location Areas if an action has been performed directly between both areas without visiting another Location Area inbetween. It is important to consider that a connection is directed here. Therefore a connection from A to B is different from the connection in its opposite direction from B to A. A connection itself consists of the different manipulation properties of the actions, which are performed on this connection. Of course the properties can depend on different factors. The first factor are the objects themselves. The other factor are the different manipulation alternatives that can occur for each object. Therefore the system needs to store the different properties, which can occur for the each object and their manipulation alternatives. Two exemplary objects of a Functionality Map can be seen in Fig. 3




(a) Part of the Functionality Map, showing the information of one object

(b) Part of the Functionality Map, showing the information of another object

Fig. 3: Functionality Map of the environment for two exemplary objects.

The properties, which are stored in the Functionality Map, are the following:

- **pushed object vs. lifted object** - An object can be manipulated by lifting or by pushing it. A pushed object needs just to be pushed in the desired direction, whereas lifting an object requires much more effort (e.g. knowledge about the way of grasping, the objects weight).
- **arbitrary movement vs. constrained traj.** - The shape of the traj. between two Location Areas has either an arbitrary shape or it is a movement with a goal. An action with a goal connects the Location Areas in a direct manner while avoiding deviations. In contrast an arbitrary movement has not such a directed shape. Consequently the movement with a goal sets a constraint on the possible traj., whereas an arbitrary movement does not.

- **connection relevance** - The connection relevance shows the probability of a connection property, based on the observed actions.
- **velocity constraints during pick-up** introduced in [18], are used: - The three phases defining an action, the pick-up, the transportation and the placement phase. The maximal speed during the pick-up phase is stored as velocity constraint in the Functionality Map. It is an indicator of the difficulty to pick up the object. grasp taxonomy
- **grasp taxonomy** - The grasp type is mainly important for the pick-up and placement phase of the manipulation and not part of this paper. The grasp taxonomy we consider for the system is summarized in [7].
- **grasp approach vector** - The grasp approach vector is similarly to the grasp type mainly important for the pick-up and placement phase of the manipulation and not part of this paper. The grasp direction is the direction, from which the object is grasped in the object-centric point of view.

The assignment of the properties to the Object Container or the Functionality Map depends on the type of the property. A property, which is related to the function in the environment, is assigned to the Functionality Map. For example the velocity constraint during the pick-up is part of the Functionality Map, since the possible velocity constraint depends on the environment of the object (e.g. obstacles). In contrast, a property, which is directly related to the object and the state of the object in the context, is a part of the Object Container. An example for such a property is the maximal allowed acceleration for an object in a certain state (e.g. no high acceleration for a filled cup) and a possible constraint on the variation of orientation during manipulation.

### C. Knowledge Extraction

The presented Object Container and Functionality Map need to be filled with information. For example a scene can be observed with a system, which provides a 6DoF-traj. of the manipulated objects.

*1) Object Container:* The properties for the Object Container, which we want to define in this paper, are the maximal acceleration value and the variation of observed orientation of the object during the manipulation.

*a) Maximal Acceleration:* The maximal acceleration value is calculated from the change of two speed values following each other. The speed is computed using two consecutive samples.

*b) Orientation:* The observed orientation is determined from the given 6DoF-traj.. For the constraints in the manipulation task just the orientation change around the vertical axis is of interest. The aim is to distinguish a movement with rotation from a movement without rotation. We use Hidden Markov Models (HMMs) [19] for classification. HMMs are statistical classifiers, which use an observation sequence for the estimation of the underlying state-sequence. Here, discrete HMM with $\lambda = (A, B, \Pi)$ are chosen. They comprise a transition probability matrix $A$, an observation symbol probability distribution matrix $B$ and an initial state distribution $\Pi$.
HMM are used for the classification task because of their ability of generalization. They are statistical classifiers, which are able to detect the underlying state sequence of the given observation sequence and they take into account knowledge of the past (previous state) in the sequential input.

First the preprocessing takes place until a codebook of the rotation information is built. An overlapping window of 400 ms with a 200 ms overlap (according to [11]) is applied on sequential input. The deviation of angle at the start position to the current angle is computed around each axis of rotation in this window. Depending on the object and the way of recording its traj., different amounts of deviations can occur for different objects. Since we need a relative amount of change for each object, the deviation is normalized for each object with its maximum deviation occurring in all movements of the object.
After this preprocessing the collected data of the rotation information is clustered with the K-mean algorithm [8] independent of its time of occurrence, resulting in a 64 symbol rotation information codebook.

Then two HMM (each with 10 states) are built for the classification of the occurrence of rotation of the object ($\lambda_r$) or not ($\lambda_{noR}$). The transition and emission probabilities for each model are calculated with the maximum likelihood estimation using the labeled training sequences.

For the evaluation, the system observes test sequences, which are preprocessed as described above. The corresponding symbols in each codebook are assigned by the k-nearest-neighbors-method. To evaluate the classification performance of the trained HMMs, the maximum log likelihood $log\ P(o_{test}|\lambda_i)$ of a given model $\lambda_i$ is computed for each test sequence with observations $o_{test}$ similar to [20]:

$$\lambda_r^* = \ arg\ max[log\ P(o_{test}|\lambda_{noR}), log\ P(o_{test}|\lambda_r)]\ . \quad (1)$$

*2) Functionality Map:*

*a) Location Areas:* The possible Location Areas have to be determined first. Therefore the available traj. are split up in single sequences, which consist of the movement of the object between two consecutive stops. A stop is a part of the traj., in which the x,y,z-coordinates do not change. The collected 3D-points of the stops are clustered and the resulting cluster-centers are the centers of the Location Areas. If objects of different heights have different height values for the stop points on the same surface, a projection on the corresponding plane makes the clustering more convenient.It is possible, that the system detects two Location Areas, which coincide in fact, but appear randomly as two. Since these Location Areas are close to each other and have the same connection properties, they can be fused.

*b) Connection Properties:* The next step is the determination of the connections between the detected Location Areas and the corresponding properties of the connections for each object. The properties, we are using in this paper, are the distinction of a pushed vs. a lifted object, an arbitrary movement vs. a movement with a goal, the velocity constraints

during the pick-up phase and the connection relevance of a movement property on a certain connection. If possible, the grasp type of the manipulation is determined.

*Pushed Object vs. Lifted Object:* An object is pushed, if it is in contact with its background during the whole manipulation.

*Arbitrary Movement vs. Movement with Constrained Traj.:* A Principle Component Analysis PCA (with rescaling) is performed for the distinction of an arbitrary movement and a movement with a constrained traj.. The PCA is done on a 4.8 s window with a 2.4 s overlap, the resulting principal components are normalized. Now we want to determine the arbitrary movement, which has no main direction of motion, but the movement is relatively large in all directions. Therefore we are especially interested in the third component, since it shows the direction of the smallest motion. If this motion has a high amplitude, the whole motion has a relatively high amplitude in all directions, since even the direction of the smallest motion is high. Consequently it is an arbitrary movement. We define the smallest motion as "high" in two cases. The first case is a comparison with the main direction of motion (= the first PCA-component): If the magnitude of the first and the third component are relatively "close" to each other, there is hardly any main direction of the movement and the movement is arbitrary. "Close" means, that the component of the smallest movement is multiplied with a factor (multiplication factor arbitrary-movement), which determines, how many times the component of the largest movement is maximally allowed to be larger than the component of the smallest movement. The second case of a "high" smallest motion is occurring, when the third component is higher than a threshold (arbitrary-movement-threshold). The arbitrary-movement-threshold has to be chosen in the magnitude of the third PCA-component of the arbitrary movements. If all the described criteria are not met, the direction of the smallest motion is not high and the movement is a movement with a constrained traj..

*Velocity constraints during the pick-up:* The pick-up phase is defined manually here with 50 samples from the starting position. As already described, the speed is computed for two samples following each other.

*Connection relevance:* The connection relevance can easily be determined by dividing the number of occurrences of a certain movement property on a connection by the number of all movements on this connection.

*Grasp Type:* In the current implementation, grasp type is determined by manual labeling.

## III. Results

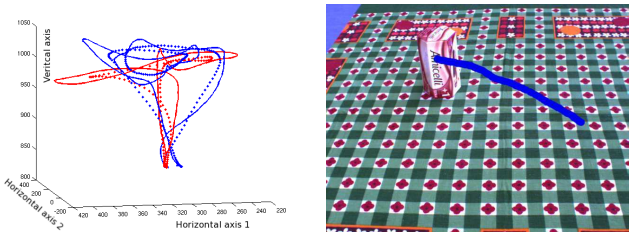
The proposed system is tested on sequences (seq.) of real human actions. Firstly, we test our system on tracking data (subsection III-A). The tracking data provides directly the 6DoF-traj. of the tracked markers, which are placed on top of the manipulated objects (obj.). Afterward in subsection III-B the system is evaluated on data, which is observed with a vision system.

TABLE I: *Left*: Description of the movements with a constrained traj. and arbitrary movements. A pushed obj. is not lifted from a plane. *Right*: Further description of the four movements with a constrained traj. (seq. 1-4, 5-8, 9-12, 13-16): The obj. is first lifted from the starting position on the table to a higher position on a box. Then it is moved back. Afterward, it is moved into the corner of the table, which is on the same level of height like the starting position, and moved back.

| Seq.: | Movement | Rotation | Seq.: | Start Pos. | End Pos. |
|---|---|---|---|---|---|
| 1-2 | constr.: line | | 1 | table-start | box |
| 3-4 | constr.: pushed | | 2 | box | table-start |
| 5-8 | constr.: curve | | 3 | table-start | corner |
| 9-12 | constr.: line | x | 4 | corner | table-start |
| 13-16 | constr.: curve | x | | | |
| 17 | arbitrary | | | | |
| 18 | arbitrary | x | | | |

### A. Basic Results on Tracking Data

The tracking data is recorded with a marker-based IR tracking system[1] at 50 Hz. The data is first preprocessed: Each sample, which does not correspond to an at least minimal movement of 0.005 m/s, is deleted. Furthermore, the traj. are smoothed with a 1.4 s moving-average-window, in order to eliminate high-frequency noise, which can especially occur at the beginning of the movement. After this basic preprocessing, the seq. vary between 4.3 s and 17.72 s, the average is 7.45 s (example in Fig. 4a).




(a) Tracking seq.: cup (b) Vision seq.: box 2

Fig. 4: Traj. of movements. The arbitrary movements of the tracking data (in mm) for the cup are shown in red (without rotation) and in blue (with rotation) (line = original movements, dotted line = result of the basic preprocessing). The traj. of a pushed object (seq. 1) is shown for the vision data.

The seq. are recorded with four different obj.: a milk carton, a spoon, a cup and a vase. The cup is grasped twice: at the handle and at the cylindrical part from the side. This leads to five "different" obj. for the test. For each obj., there are 18 different actions of a person, shown in Table I. The implementation is done in Matlab (Statistics Toolbox: PCA, HMM, K-mean algorithm. Bioinformatics Toolbox: knn-classification.[2]).

[1]Advanced Realtime Tracking system. Advanced Realtime Tracking GmbH, *url:* http://www.ar-tracking.de/ .

[2]Matlab: Statistics Toolbox, Bioinformatics Toolbox.

TABLE II: Statistical results of the classifications: Accuracy, true positive rate and true negative rate. T = Tracking data, V = Vision data.

| Property: | Accuracy | True pos. rate | True neg. rate |
|---|---|---|---|
| Rotation (T) | 80.0% | 66.7% | 93.3% |
| Pushed Object (T) | 98.9% | 100.0% | 87.5% |
| Arbitrary Movem. (T) | 94.4% | 80.0% | 96.3% |
| Rotation (V) | 77.5% | 93.8% | 66.7% |
| Pushed Object (V) | 95.0% | 87.5% | 96.9% |
| Arbitrary Movem. (V) | 80.6% | 100.0% | 78.6% |

The parameters and the initial values for the knowledge extraction (see Section II-C) are set as follows. The knn-assignment of a new value to a cluster in the rotation information codebook is done with k=3. For the classification as arbitrary movement in the Functionality Map, the multiplication factor arbitrary-movement for the third component is 15, and the arbitrary-movement-threshold is 0.06. Concerning the distinction pushed vs. lifted obj., the height difference to the table is measured along the axis, which is vertical to the table. The obj. is pushed, if its height difference to the table is not changing ($\pm 5\ mm$). The maximal acceleration is computed for a window of 8 ms. Furthermore, the initialization of the cluster for the build-up of the rotation information codebook is set. Otherwise the results of the clustering are not always deterministic, even though they look mostly very similar. The initialization values are chosen between 0 and 1, since the input values are the normalized changes of the angles.

*1) Object Container:* For the rotation classification, a leave-one-out cross validation is made. The results show, that 42 of 45 of the motions without rotation are correctly labeled, and 30 of 45 motions with rotation are correctly classified. Therefore the system performs definitely better than guessing the classification (ground truth: 50%), and performs quite well (see Table II).

The final result of the Object Container can be seen in Table III. In order to make the Object Container more generic, acceleration classes are introduced. The number of acceleration classes is set to three for illustration. Each class represents an approximately equal sized part of the achieved acceleration values within $0.003 - 0.01\ m/s^2$.

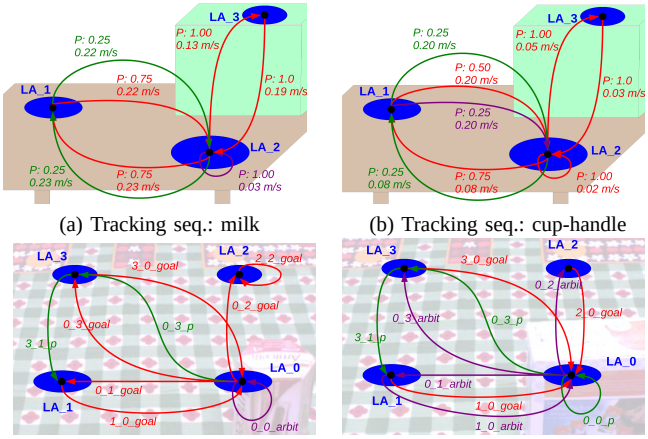
*2) Functionality Map:* The three used location areas have been correctly identified. A leave-one-out cross validation is done for the classification of the (non-)arbitrary movements (at first without the distinction of a pushed or lifted obj.). 8 of 10 arbitrary movements are correctly labeled, and 77 of 80 movements with a constrained traj. are correctly classified. This shows, that the system performs definitely better than guessing (ground truth: 11%). For the true positive rate (see Table II), one has to consider, that there are just 10 arbitrary movements among all 90 seq., leading to a significant influence of every mislabeled arbitrary movement.

The classification of the pushed vs. the lifted obj. is successful for all seq. except one spoon-sequence.

The results of the Functionality Maps show, that the system is able to deal with some misclassifications, since it achieves correct high connection relevances for all obj. except for the cup-handle. The best (= completely correct) results are achieved for the cup and the milk (Fig. 5a). The worst result is the Functionality Map of the cup-handle, since it contains the highest number of misclassifications (two misclassifications) among all Functionality Maps. In Fig. 5b, The misclassified arbitrary movements (red self-loop LA_2) and the misclassified movement with a constrained traj. (magenta connection from LA_1 to LA_2) are drawn. The results for the other two obj. show just one misclassification.

TABLE III: Result: Object Container. The number of observations per acceleration class and the rotation-classification are shown. Legend: R = motion with rotation, Acc. class = acceleration class: acc. class 1 for $x < 0.006\ m/s^2$, acc. class 2 for $0.006\ m/s^2 \leq x < 0.009\ m/s^2$ and acc. class 3 for $0.009\ m/s^2 \leq x$.

| Acc. class | 1 | | 2 | | 3 | |
|---|---|---|---|---|---|---|
| Objects Tracking | no R | R | no R | R | no R | R |
| Milk | 3 | 3 | 6 | 3 | 3 | 0 |
| Spoon | 3 | 1 | 3 | 4 | 4 | 3 |
| Cup-handle | 2 | 4 | 3 | 8 | 1 | 0 |
| Cup | 5 | 2 | 8 | 1 | 2 | 0 |
| Vase | 10 | 4 | 4 | 0 | 0 | 0 |
| Objects Vision | no R | R | no R | R | no R | R |
| Object 1 | 1 | 1 | 0 | 1 | 2 | 5 |
| Object 2 | 3 | 0 | 0 | 0 | 4 | 3 |
| Object 3 | 2 | 1 | 1 | 1 | 1 | 4 |
| Object 4 | 0 | 2 | 2 | 2 | 1 | 3 |




(a) Tracking seq.: milk (b) Tracking seq.: cup-handle

(c) Vision seq.: obj. 1 (d) Vision seq.: obj. 2

Fig. 5: Result: Functionality Maps. Red arrow = movement with a constrained traj., green arrow = pushed obj., magenta arrow = arbitrary movement.

## B. Results from a Vision System

The vision data is recorded with a Firewire Marlin FO46C camera at 30 Hz and an image size of 640x480 pixel (width x height). The traj. are acquired as described in [18]. The C++ Implementation of Hidden Markov Model by Dekang

TABLE IV: **Sequence properties - vision system**. 10 Seq. are recorded with each of the 4 obj.. The start and end positions are the bottom right (br), the bottom left (bl), the top right (tr) and the top left (tl) of a table.

| Seq.: | Movement | Rotation | Start Pos. | End Pos. |
|---|---|---|---|---|
| 1, 11, 21, 31 | constr.: push | | br | tl |
| 2, 12, 22, 32 | constr.: push | | tl | bl |
| 3, 13, 23, 33 | constr.: curve | | bl | br |
| 4, 14, 24, 34 | arbitrary | | br | br |
| 5, 15, 25, 35 | constr.: curve | | br | tl |
| 6, 16, 26, 36 | constr.: curve | | tl | br |
| 7, 17, 27, 37 | constr.: curve | x | br | tr |
| 8, 18, 28, 38 | constr.: curve | x | tr | br |
| 9, 19, 29, 39 | constr.: curve | x | br | bl |
| 10, 20, 30, 40 | constr.: curve | x | bl | br |

Lin [3] is (slightly modified) is used for the implementation of HMMs. The PCA, the K-means algorithm and the knn-classification are done with OpenCV. The properties of the recorded seq. of real human actions are listed in Table IV. An example is shown in Fig. 4b.

Similarly to the tracking data, a basic preprocessing is performed (minimal movement $> 0.01/sample$, 140 sample moving-average-window). Furthermore the first and last 20 samples were cut of, in order to deal with the arbitrary motions at the beginning and at the end of the seq.. The angle-values are smoothed along each dimension separately. The initialization and threshold values are set as for the experiment with the tracking data, except for the arbitrary-movement-threshold (0.06 for the vision data) and the window for the acceleration-computation (16 ms).

*1) Object Container:* The appearance of (non-) rotation is correctly identified for 31 of 40 seq. (Table II). Eight seq. are mislabeled as seq. with rotations. All of them show, that one or both horizontal angles vary during the manipulation. The variations are not as strong as for most of the seq. with rotation, but it is still visible. Table III shows the final result of the Object Container.

*2) Functionality Map:* The Location Areas themselves are successfully determined. The assignment is successful for 77 of 80 positions (96.3%). The missclassifications occur for the end positions of seq. 8, 21 and 33. These misclassifications are mainly caused by the z-components (the depth) of the end positions, which are closer to the wrong Location Areas.

As the statistical measures in Table II show, the result of the distinction between a pushed obj. and an obj., which is lifted for the movement, is remarkable. There is just one seq. mislabeled as pushed obj., and one seq. mislabeled as raised obj.. The performance of the classification as arbitrary movement or as movement with a constrained traj. achieves a true positive rate of 100.0%. Consequently, no arbitrary movement is mislabeled as non-arbitrary movement. Six seq. are misclassified as arbitrary movements instead of movements with constrained traj.. These movements contain small parts with an arbitrary shape. The kind of grasp is analyzed according to [7]. All used grasps are power grasps with an abducted position of the thumb.

[3]Copyright (C) 2003 Dekang Lin, lindek@cs.ualberta.ca, *url:* http://webdocs.cs.ualberta.ca/ lindek/hmm.htm .

TABLE V: Result: Connection properties of a fictional obj., which is built on all seq. of the vision data. The relevance (probability P) of each connection property is shown.

| $COall_{ij}$ | P(constr.) | P(arbit) | P(push) |
|---|---|---|---|
| $COall_{00}$ | 0.00 | 0.75 | 0.25 |
| $COall_{01}$ | 0.60 | 0.40 | 0.00 |
| $COall_{02}$ | 0.75 | 0.25 | 0.00 |
| $COall_{03}$ | 0.43 | 0.14 | 0.43 |
| $COall_{10}$ | 0.86 | 0.14 | 0.00 |
| $COall_{12}$ | 1.00 | 0.00 | 0.00 |
| $COall_{20}$ | 0.67 | 0.33 | 0.00 |
| $COall_{22}$ | 1.00 | 0.00 | 0.00 |
| $COall_{30}$ | 1.00 | 0.00 | 0.00 |
| $COall_{31}$ | 0.00 | 0.00 | 1.00 |

The Functionality Maps of obj. 1 and 4 have just one wrong assignment of an end location each, everything else is correct (see obj. 1 in Fig. 5c). The Functionality Map of obj. 2 suffers mainly from misclassifications as arbitrary movements (see Fig. 5d). One movement of obj. 3 can be seen a outlier, since its connection property, as well as the assignment of its end location, are wrong. Besides one further misclassified connection property, the Functionality Map of obj. 3 is correct.

Most of the connections in the Functionality Maps are correctly established, even though just one seq. is observed for the connection. Taking this into account, the results of the Functionality Maps are already very good, only the Functionality Map of obj. 2 can be improved. If just one manipulation is observed along a connection, each miss-classification leads to another connection property in the resulting Functionality Map. The observation of more seq. enables a better usage of the connection relevance, which is assigned to each connection. In order to give an impression of this possibility, all observations of the four obj. are treated as observations of the same fictional obj.. It should be pointed out, that it is not the intention to sum up all observations in general. The purpose is just to illustrate a Functionality Map, based on some observations per connection.

Of course, the 3 seq. with misassigned end locations result in wrong connections as well. The effect of a higher number of observations per connections becomes clear in Table V. The mislabeled connection properties have a lower relevance than the correctly labeled connections properties ($COall_{00}$, $COall_{01}$, $COall_{02}$, $COall_{03}$, $COall_{10}$, $COall_{20}$). Consequently, the system shows its ability to deal with misclassifications, if more than one observation is made for each connection. It should be pointed out, that the system gets a maximum of four observations per connection property for most of the connections (exception: 8 observations on $COall_{10}$).

## IV. Conclusions and Future Work

The proposed system is developed for abstract representation of manipulation-relevant knowledge of objects. This

system aims to monitor object properties and function in a given environment The experiments on tracking and vision data show that the system can derive the knowledge from different sources, which provide the necessary information for the system. The presented system allows an efficient monitoring scheme for detection of unexpected (surprising) event that require an update of the information in the internal representation. The proposed framework allows to deal with the strong variation in actions performed by a human operator reducing the number of false positive surprise events to a minimum.

The proposed descriptors consisting of an Object Container and a Functionality Map spanning typical object locations in a graph allow a close monitoring for changes in a physical state of the object and its function in a given environment.

The results from the vision system show that a single trajectory of a certain functionality is not enough to avoid with a misclassification. An observation of multiple actions in along a given edge of the Functionality Map (graph) allow us though to estimate the function of the object in the world. Our next goal is to focus more on unknown situations and environments. They provide new information to the system.

## REFERENCES


[1] T. Asfour, P. Azad, F. Gyarfas, and R. Dillmann. Imitation learning of dual-arm manipulation tasks in humanoid robots. *International Journal of Humanoid Robotics*, 5(2):183–202, 2008.

[2] G. Bombini, N. Di Mauro, T. M. A. Basile, S. Ferilli, and F. Esposito. Relational Learning by Imitation. In A. Håkansson et al., editor, *KES-AMSTA 2009*, volume 5559, pages 273–282. Lecture Notes in Artificial Intelligence, Springer-Verlag Berlin Heidelberg, 2009.

[3] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. Billard. Learning and Reproduction Gestures by Imitation. *IEEE Robotics and Automation Magazine*, 17:44 – 54, 2010.

[4] A. Chella, H. Dindo, and I. Infantino. Learning high-level tasks through imitation. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3648–3654, 2006.

[5] Z. Duric, J. A. Fayman, and E. Rivlin. Function from Motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(6):579–591, June 1996.

[6] M. Eich, M. Dabrowska, and F. Kirchner. Semantic Labeling: Classification of 3D Entities Based on Spatial Feature Descriptors. In *IEEE International Conference on Robotics and Automation: Workshop on Best Practice Algorithms in 3D Perception and Modeling for Mobile Manipulation*, Anchorage, USA, 2010.

[7] T. Feix, R. Pawlik, H.-B. Schmiedmayer, J. Romero, and D. Kragic. The generation of a comprehensive grasp taxonomy. In *Robotics, Science and Systems Conference: Workshop on Understanding the Human Hand for Advancing Robotic Manipulation, Poster Presentation*, June 2009.

[8] J.A. Hartigan and M.A. Wong. A k-means clustering algorithm. *Applied Statistics*, 8(1):100–108, 1979.

[9] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Movement Imitation with Nonlinear Dynamical Systems in Humanoid Robots. In *IEEE International Conference on Robotics and Automation*, pages 1398–1403, Washington, DC, USA, 2002.

[10] B. Jansen and T. Belpaeme. A Model for Inferring the Intention in Imitation Tasks. In *The 15th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN'06*, pages 238–243, 2006.

[11] M. Kawato. Trajectory formation in arm movements: Minimization principles and procedures. In *Advances in Motor Learning and Control, ser. Human Kinetics, H. N. Zelaznik, Ed. Human Kinetics Publishers, Chanpaign Illinois*, pages 225–259, 1996.

[12] K.Ogawara, J.Takamatsu, K.Kimura, and K.Ikeuchi. Generation of a task model by intergrating multiple observations of human demonstrations. In *Proceedings of the IEEE Intl. Conf. on Robotics and Automation (ICRA '02)*, pages 1545–1550, May 2002.

[13] V. Krüger, D. L. Herzog, S. Baby, A. Ude, and D. Kragic. Learning actions from observations. *IEEE Robotics and Automation Magazine*, pages 30–43, June 2010.

[14] M. Mitani, M. Takaya, A. Kojima, and K. Fukunaga. Environment Recognition Based on Analysis of Human Actions for Mobile Robot . In *The 18th International Conference on Pattern Recognition (IEEE)*, pages 782–786, 2006.

[15] M. Pardowitz, S. Knoop, R. Dillmann, and R. D. Zöllner. Incremental Learning of Tasks From User Demonstrations, Past Experiences, and Vocal Comments . *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 37(2):322–332, April 2007.

[16] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and Generalization of Motor Skills by Learning from Demonstration. In *IEEE International Conference on Robotics and Automation*, pages 763–768, Kobe, Japan, 2009.

[17] R. Paul and P. Newman. FAB-MAP 3D: Topological Mapping with Spatial and Visual Appearance. In *IEEE International Conference on Robotics and Automation*, pages 2649–2656, Anchorage, USA, 2010.

[18] S. Petsch and D. Burschka. Estimation of Spatio-Temporal Object Properties for Manipulation Tasks from Observation of Humans. In *IEEE International Conference on Robotics and Automation*, pages 192–198, Anchorage, USA, 2010.

[19] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *IEEE*, 77(2):257–286, 1989.

[20] C.E. Reiley and G.D. Hager. Task versus subtask surgical skill evaluation of robotic minimally invasive surgery. In *Medical Image Computing and Computer-Assisted Intervention -MICCAI 2009*, pages 435–442, 2009.

[21] L. Stark, K. Bowyer, A. Hoover, and D. B. Goldgof. Recognizing Object Function Through Reasoning About Partial Shape Descriptions and Dynamic Physical Properties. In *Proceedings of the IEEE*, volume 84, pages 1640–1656, 1996.

[22] R. S. Sutton und A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[23] D. Verma and R. P. N. Rao. Imitation Learning Using Graphical Models. In J. N. Kok et al., editor, *ECML 2007*, volume 4701, pages 757–764. Lecture Notes in Artificial Intelligence, Springer-Verlag Berlin Heidelberg, 2007.

[24] R. D. Zöllner and R. Dillmann. Using multiple probabilistic hypothesis for programming one and two hand manipulation by demonstration. In *IEEE International Conference on Intelligent Robots and Systems*, pages 2926–2931, Las Vegas, Nevada, USA, 2003.

# Stochastic Global Optimization for Robust Point Set Registration

Chavdar Papazov*, Darius Burschka

*Technische Universität München, Department of Computer Science*
*Boltzmannstr. 3, 85748 Garching, Germany*

**Abstract**

In this paper, we propose a new algorithm for pairwise rigid point set registration. The main properties of our method are noise robustness, outlier resistance and global optimal alignment. The problem of registering two point clouds in space is converted to a minimization of a nonlinear cost function. We propose a new cost function based on an inverse distance kernel that significantly reduces the impact of noise and outliers. In order to achieve a global optimal registration without the need of any initial alignment, we develop a new stochastic approach for global minimization. It is an adaptive sampling method which uses a generalized BSP tree and allows for minimizing nonlinear scalar fields over complex shaped search spaces like, e.g., the space of rotations. We introduce a new technique for a hierarchical decomposition of the rotation space in disjoint equally sized parts called spherical boxes. Furthermore, a procedure for uniform point sampling from spherical boxes is presented. Tests on a variety of point sets show that the proposed registration method performs very well on noisy, outlier corrupted and incomplete data. For comparison, we report how two state-of-the-art registration algorithms perform on the same data sets.

*Keywords:* Rigid registration, robust cost function, stochastic global optimization, generalized BSP tree, hierarchical decomposition of $SO(3)$, uniform sampling from spherical boxes

*Corresponding author

*Email addresses:* papazov@in.tum.de (Chavdar Papazov), burschka@in.tum.de (Darius Burschka)

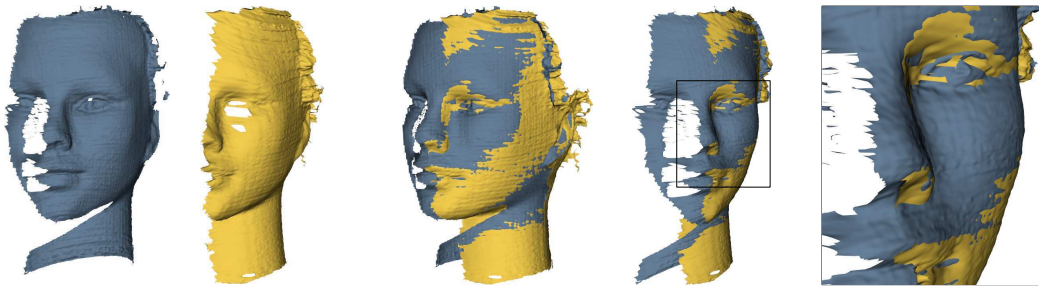



Figure 1: Pairwise rigid point set registration obtained with our method. The input point sets, model and data, are shown in (a) and (b), respectively. Although rendered as meshes no surface information (like, e.g., normals) is used for the registration. Note that the scans are noisy and only partially overlapping. (c), (d) Our registration result (shown from two different viewpoints) obtained without noise filtering, local ICP refinement [1] or any assumptions about the initial pose of the input scans. (e) A closer view of the part marked by the rectangle in (d). Observe the high quality of the alignment.

## 1. Introduction and Related Work

Point set registration is a fundamental problem in computational geometry with applications in the fields of computer vision, computer graphics, image processing and many others. The problem can be formulated as follows. Given two finite point sets $\mathbf{M} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subset \mathbb{R}^3$ and $\mathbf{D} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\} \subset \mathbb{R}^3$ find a mapping $T : \mathbb{R}^3 \to \mathbb{R}^3$ such that the point set $T(\mathbf{D}) = \{T(\mathbf{y}_1), \ldots, T(\mathbf{y}_n)\}$ is optimally aligned in some sense to $\mathbf{M}$. $\mathbf{M}$ is referred to as the model point set (or just the model) and $\mathbf{D}$ is termed the data point set. Points from $\mathbf{M}$ and $\mathbf{D}$ are called model points and data points, respectively.

If $T$ is a rigid transform, i.e., $T(\mathbf{x}) = R\mathbf{x} + \mathbf{t}$ for a rotation matrix $R$ and a translation vector $\mathbf{t}$, we have to solve a rigid point set registration problem. This special case is of major importance for the tasks of object recognition, tracking, localization and mapping, and object modeling, just to name a few. The problem is especially hard when no initial pose estimation is available, the point sets are noisy, corrupted by outliers and incomplete. In Figure 1, a model and a data set are shown before and after rigid registration.

### *1.1. Rigid Point Set Registration*

Algorithms for the registration problem belong to two general classes. One class consists of methods designed to solve the initial pose estimation

problem[1]. These methods compute a (more or less) coarse alignment between the point sets without making any assumptions about their initial position and orientation in space. Classic initial pose estimators are the generalized Hough transform [2], geometric hashing [3] and pose clustering [4]. These algorithms are guaranteed to find the optimal alignment between the input point sets. However, because of their high computational cost and/or high memory requirements, these methods are only applicable to small data sets.

Johnson *et al.* introduced in their work [5] local geometric descriptors, called spin images, and used them for pose estimation and object recognition. The presented results are impressive, but no tests with noisy or outlier corrupted data were performed. Gelfand *et al.* [6] developed a local descriptor which performs well under artificially created noisy conditions, but still, defining robust local descriptors in the presence of significant noise and a large amount of outliers remains a difficult task.

A more recent approach to the initial pose estimation problem is the robust 4PCS algorithm introduced by Aiger *et al.* [7]. It is an efficient randomized generate-and-test approach. It computes for an appropriate quadruple $\mathbf{B}$ (called a basis) of nearly coplanar points from the model set $\mathbf{M}$ the optimal rigid transform between $\mathbf{B}$ and each of the potential bases in the data set $\mathbf{D}$ and chooses the best one. In order to achieve high probability for success, the procedure is repeated several times for different bases $\mathbf{B} \subset \mathbf{M}$. Note, however, that the rigid transform, found by the algorithm, is optimal only for the two bases (i.e., for eight points). In contrast to this, the rigid transform we compute is optimal for all points of the input sets and thus we expect to achieve higher accuracy than the 4PCS algorithm. This is further validated in the experimental results in Section 5 of this paper.

Since the accuracy of the pose computed by the above mentioned methods is insufficient for many applications, an additional pose refinement step needs to be performed. The pose refining algorithms represent the second class of registration approaches. The most popular one is the Iterative Closest Point (ICP) algorithm. Since its introduction by Chen and Medioni [8], and Besl and McKay [1], a variety of improvements has been proposed in the literature. A good summary as well as new results in acceleration of ICP algorithms have been given by Rusinkiewicz and Levoy [9]. A major drawback of ICP and all its variants is that they assume a good initial guess for the pose of the

[1]Pose = position (translation) + orientation (rotation).

data point set (with respect to the model). This pose is improved in an iterative fashion until an optimal rigid transform is found. The quality of the solution depends heavily on the initial guess. A further disadvantage of the methods compared by Rusinkiewicz and Levoy [9] is that they use local surface features like surface normals which cannot be computed very reliably in the presence of noise.

### *1.2. Optimization Based Point Set Registration*

Solving the registration problem by minimizing a cost function with a general-purpose optimization algorithm has already been introduced in the literature. Depending on the choice of either a global or a local optimization procedure the corresponding registration approach belongs to the class of initial pose estimators or pose refining methods, respectively.

Breuel [10] and Olsson *et al.* [11] used deterministic branch-and-bound methods to minimize globally the sum of squared distances between corresponding points in $\mathbf{M}$ and $\mathbf{D}$. Although these methods are guaranteed to find the globally optimal solution, the computational cost seems to be very high since in [10] only planar rigid transforms (with three degrees of freedom) were considered and the experimental results provided in [11] were based on point sets consisting of not more than 60 points. Furthermore, in [11], a pointwise correspondence between the point sets has to be known in advance which is very seldom the case in a real world setting.

Mitra *et al.* [12], Pottmann *et al.* [13] and Fitzgibbon [14] also formulated the registration problem as a minimization of a geometric cost function. For its minimization, however, a local optimization method is used. This results in the already mentioned strong dependence on a good initial transform estimation.

### *1.3. Stochastic Optimization*

Stochastic optimization has received considerable attention in the literature over the last three decades. Much of the work has been devoted to the theory and applications of simulated annealing (SA in the following) as a minimization technique [15, 16, 17]. A comprehensive overview of this field is given in [18]. A major property of SA algorithms is their "willingness" to explore regions around points in the search space at which the objective function takes values greater than the current minimum [19]. This is what makes SA algorithms able to escape from local minima and makes them suitable for global minimization. A known drawback of SA algorithms is the

fact that they waste a lot of iterations generating candidate points, evaluating the objective function at these points, and finally rejecting them [18]. In order to reduce the number of rejections, Bilbro and Snyder [20] select candidate points from "promising" regions of the search space, i.e., from regions in which the objective function is likely to have low values. They achieve this by adapting a k-d tree to the objective function each time a new candidate point is accepted. If, however, the current point is rejected, the tree remains unchanged. This is a considerable waste of computation time since the information gained by the (expensive) evaluation of the objective function is not used. In contrast to this, our algorithm adapts a generalized BSP tree at every iteration and thus uses all the information collected during the minimization. Furthermore, the use of a generalized BSP tree allows for a minimization over complex shaped spaces and not only over rectangular regions as in the case of [20].

### *1.4. Contributions and Overview*

Our registration algorithm aims to solve the initial pose estimation problem robustly in the case of noisy, outlier corrupted and incomplete point sets. Our main contributions are (i) a noise and outlier resistant cost function, (ii) a stochastic approach for its global minimization, (iii) a technique for a hierarchical rotation space decomposition in disjoint parts of equal volume and (iv) a procedure for uniform sampling from spherical boxes. The work presented here is a significant extension of the concept introduced in [21].

The rest of the paper is organized as follows. In Section 2, we define the task of aligning two point sets as a nonlinear minimization problem and define our cost function. In Section 3, a stochastic approach for global minimization is presented. In Section 4, we motivate the choice of the rotation space parametrization we use in combination with our minimization approach and introduce a technique for a hierarchical rotation space decomposition. Furthermore, a procedure for uniform sampling from spherical boxes is described. Section 5 presents experimental results obtained with our registration algorithm as well as comparisons with two state-of-the-art registration methods. The paper ends with some conclusions in Section 6.

## 2. Registration as a Minimization Problem

Consider, we are given a model point set $\mathbf{M} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subset \mathbb{R}^3$ and a data point set $\mathbf{D} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\} \subset \mathbb{R}^3$. Suppose, we have a continuous

function $S : \mathbb{R}^3 \to \mathbb{R}$, called the model scalar field, which attains small values at the model points $\mathbf{x}_j,\ j \in \{1, \ldots, m\}$ and increases with increasing distance between the evaluation point and the closest model point. Our aim is to find a rigid transform $T : \mathbb{R}^3 \to \mathbb{R}^3$ of the form $T(\mathbf{x}) = R\mathbf{x} + \mathbf{t}$ for a rotation matrix $R$ and a translation vector $\mathbf{t} \in \mathbb{R}^3$ such that the functional

$$\mathcal{F}(T) = \sum_{i=1}^{n} S(T(\mathbf{y}_i)), \quad \mathbf{y}_i \in \mathbf{D} \tag{1}$$

is minimized. This definition of $\mathcal{F}$ is based on the following idea common for most registration algorithms: we seek a rigid transform that brings the data points as close as possible to the model points.

### *2.1. Definition of the Model Scalar Field*

Given the model point set $\mathbf{M} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, we want our model scalar field $S : \mathbb{R}^3 \to \mathbb{R}$ to attain its minimal value at the model points, i.e.,

$$S(\mathbf{x}_j) = s_{\min} \in \mathbb{R}, \quad \forall \mathbf{x}_j \in \mathbf{M}, \tag{2}$$

and to attain greater values for all other points in $\mathbb{R}^3$, i.e.,

$$S(\mathbf{x}) > s_{\min}, \quad \forall \mathbf{x} \in \mathbb{R}^3 \setminus \mathbf{M}. \tag{3}$$

Define

$$d_{\mathbf{M}}(\mathbf{x}) = \min_{\mathbf{x}_j \in \mathbf{M}} \|\mathbf{x} - \mathbf{x}_j\| \tag{4}$$

to be the distance between a point $\mathbf{x} \in \mathbb{R}^3$ and the set $\mathbf{M}$, where $\|\cdot\|$ is the Euclidean norm in $\mathbb{R}^n$. If we set

$$S(\mathbf{x}) = d_{\mathbf{M}}(\mathbf{x}), \tag{5}$$

we get an unsigned distance field which is implicitly used by ICP [1]. It is obvious that this choice for $S$ fulfills both criteria (2) and (3).

Mitra *et al.* [12] and Pottmann *et al.* [13] considered in their work more sophisticated scalar fields. They assumed that the model point set $\mathbf{M}$ consists of points sampled from an underlying surface $\Phi$. The scalar field $S$ at a point $\mathbf{x} \in \mathbb{R}^3$ is defined to be the squared distance from $\mathbf{x}$ to $\Phi$. In this context, $S$ is called the squared distance function to the surface $\Phi$. We refer to [12] for details on computing the squared distance function and its approximation for point sets.

The version of $S$ given in (5) and the ones used by Mitra *et al.* [12] and Pottmann *et al.* [13] are essentially distance fields. This means that $S(\mathbf{x})$ approaches infinity as the point $\mathbf{x}$ gets infinitely far from the point set. This has the practical consequence that a registration technique which minimizes a cost function based on an unbounded scalar field will be sensitive to outliers in the data set. This is because data points lying far away from the model point set will have great impact on the sum in (1) and thus will prevent the minimization algorithm from converging towards the right alignment. A similar problem arises in the case of model and data sets with low overlap. In this case, there will be a lot of data points which have no corresponding model points and vice versa. The distance between such a data point and the closest model point will be large and thus will deteriorate the sum in (1). A simple way to overcome this is just to exclude data points which are too far away from the model set. However, this strategy introduces discontinuities in the cost function which cause a problem for many optimization methods.

Fitzgibbon presented in his work [14] a more convenient way to alleviate these difficulties which does not lead to a discontinuous cost function. He proposed to use either of the following two robust kernels:

$$S(\mathbf{x}) = \log\left(1 + \frac{\left(d_{\mathbf{M}}(\mathbf{x})\right)^2}{\sigma}\right) \quad \text{(Lorentzian kernel) or} \tag{6}$$

$$S(\mathbf{x}) = \begin{cases} \left(d_{\mathbf{M}}(\mathbf{x})\right)^2 & \text{if } d_{\mathbf{M}}(\mathbf{x}) < \sigma \\ 2\sigma d_{\mathbf{M}}(\mathbf{x}) - \sigma^2 & \text{otherwise} \end{cases} \quad \text{(Huber kernel).} \tag{7}$$

However, we still have $\lim_{d_{\mathbf{M}}(\mathbf{x})\to\infty} S(\mathbf{x}) = \infty$ for both kernels as in the case of (5). Thus a cost function based on (6) or (7) will still be sensitive to outliers. We further validate this in the experimental results presented in Section 5 of the paper.

To avoid this sensitivity, we propose to use a bounded scalar field satisfying (2) and (3) and having the additional property

$$\lim_{d_{\mathbf{M}}(\mathbf{x})\to\infty} S(\mathbf{x}) = 0. \tag{8}$$

We set

$$S(\mathbf{x}) = -\varphi\left(d_{\mathbf{M}}(\mathbf{x})\right), \tag{9}$$

where $\varphi : \mathbb{R}^* \to \mathbb{R}^*$, for $\mathbb{R}^* = \{x \in \mathbb{R} : x \geq 0\}$, is a strictly monotonically

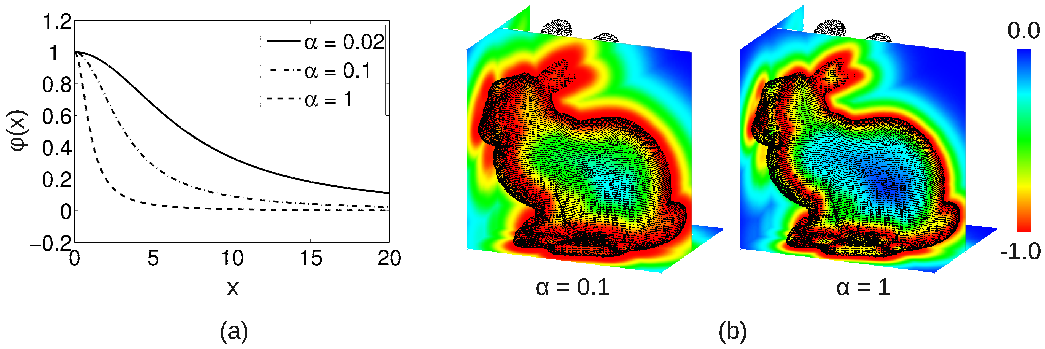



Figure 2: (a) The inverse distance kernel (defined in (12)) for three different $\alpha$ values. (b) The model scalar field $S_\alpha^{\mathbf{M}}(\mathbf{x})$ (defined in (13)) based on the inverse distance kernel from (a) for $\alpha = 0.1$ and $\alpha = 1$. In this example, the Stanford bunny is used as the model set. $S_\alpha^{\mathbf{M}}(\mathbf{x})$ is visualized by evaluating it at a number of points lying on the three planes and visualizing the scalar values using a standard color mapping technique.

decreasing continuous function with

$$\max_{x \in \mathbb{R}^*} \varphi(x) = \varphi(0) \quad \text{and} \tag{10}$$

$$\lim_{x \to \infty} \varphi(x) = 0. \tag{11}$$

In our implementation, we use an inverse distance kernel of the form

$$\varphi(x) = \frac{1}{1 + \alpha x^2}, \quad \alpha > 0 \tag{12}$$

because it is computationally efficient to evaluate and can be controlled by a single parameter $\alpha$ (see Figure 2(a)). This results in the following model scalar field:

$$S_\alpha^{\mathbf{M}}(\mathbf{x}) = -\frac{1}{1 + \alpha \left(d_{\mathbf{M}}(\mathbf{x})\right)^2}, \quad \alpha > 0. \tag{13}$$

It is easy to see that (2), (3) and (8) hold. Different values for $\alpha$ in (13) lead to different scalar fields. The greater the value the faster $S_\alpha^{\mathbf{M}}(\mathbf{x})$ convergences to zero as $d_{\mathbf{M}}(\mathbf{x}) \to \infty$ (see Figure 2(b)). In Section 2.2, we will discuss how to choose a suitable value for $\alpha$ and why this particular form of $S_\alpha^{\mathbf{M}}(\mathbf{x})$ leads to an outlier robust cost function.

### *2.2. Cost Function Definition*

The group of all rigid transforms in $\mathbb{R}^3$ is called the special Euclidean group and is denoted by $SE(3)$. At the beginning of Section 2, we formulated

the rigid point set registration problem as a functional minimization problem over $SE(3)$. Using a parametrization of $SE(3)$, the functional $\mathcal{F}$ in (1) can be converted to a real-valued scalar field $F : \mathbb{R}^6 \to \mathbb{R}$ of the form

$$F(\varphi, \psi, \theta, x, y, z) = \sum_{i=1}^{n} S_\alpha^{\mathbf{M}}(R_{\varphi,\psi,\theta}\mathbf{y}_i + (x, y, z)), \tag{14}$$

where $\mathbf{y}_1, \ldots, \mathbf{y}_n$ are the data points, $S_\alpha^{\mathbf{M}}$ is the model scalar field defined in (13), $R_{\varphi,\psi,\theta}$ is a rotation matrix parametrized by $\varphi, \psi, \theta$ and $(x, y, z) \in \mathbb{R}^3$ is a translation vector. In order to achieve good optimization performance, it is very important to choose the right parametrization of the rotation group. We employ an axis-angle based parametrization which is especially well suited for our branch and "stochastic bound" minimization method. Furthermore, we introduce a new technique for a hierarchical decomposition of the rotation space in spherical boxes and describe a procedure for uniform sampling from them. Since the advantages of these techniques are best seen in the context of our minimization algorithm we postpone the detailed discussion to Section 4 after the introduction of the minimization method in Section 3.

A global minimizer $\mathbf{x}^* \in \mathbb{R}^6$ of $F$ defines a rigid transform that brings the data points as close as possible to the model points. What makes the proposed cost function robust to outliers is the fact that outlier data points have a marginal contribution to the sum in (14) depending on $\alpha$. More precisely, given a positive real number $d$, we can compute a value for $\alpha$ such that $|S_\alpha^{\mathbf{M}}(\mathbf{x})|$ is less than an arbitrary $\delta > 0$, if $\mathrm{d}_{\mathbf{M}}(\mathbf{x}) > d$ holds. In this way, the contribution of an outlier point to the sum in (14) can be made arbitrary close to zero and $F$ will behave like an outlier rejector. However, too large values for $\alpha$ will lead to the rejection of data points which do not have exact counterparts in a sparsely sampled model set, but still are not outliers. In our implementation we set

$$d = \frac{1}{4}\min\{bbox_x(\mathbf{M}), bbox_y(\mathbf{M}), bbox_z(\mathbf{M})\}, \tag{15}$$

$$\delta = 0.1, \tag{16}$$

where $bbox(\mathbf{M})$ denotes the bounding box of the model point set and $bbox_s(\mathbf{M})$, $s \in \{x, y, z\}$ is the extent of the bounding box along the $x, y$ or $z$ axis. Using the absolute value of the right side of (13) and solving for $\alpha$ yields

$$\alpha = \frac{1 - \delta}{\delta d^2}. \tag{17}$$

The cost function given in (14) is nonlinear and nonconvex. This results in a large number of local minima of $F$ over the search space. Using a local optimization procedure—common for many registration methods in the literature—will lead in most cases to a local minimizer of $F$ and thus will not give the best alignment between model and data. To avoid this, we employ a new stochastic approach for global minimization described in the next Section of this paper.

## 3. Stochastic Adaptive Search for Global Minimization

Our stochastic minimization approach is inspired by the simulated annealing (SA) method of Bilbro and Snyder [20]. The main difference between their work and a typical SA algorithm is the way how the minimizer candidates are generated. As we already mentioned in Section 1.3, SA algorithms are known to waste many iteration in sampling candidate points from the search space, evaluating the cost function at these points and finally rejecting them [18]. In order to reduce the number of rejections, Bilbro and Snyder [20] sampled the points from a distribution which is modified iteratively during the minimization such that its modes are built around minimizers of the cost function. They achieved this by building a k-d tree and sampling the candidates from those leaves of the tree which cover "promising" regions of the search space, i.e., regions in which the cost function is likely to attain low values. Although this leads to fewer candidate rejections and thus saves computation time the method in [20] still has two drawbacks. First, the candidate points are sampled directly from the tree leaves which are n-dimensional boxes of the form $[a_1, b_1] \times ... \times [a_n, b_n]$, where $[a_i, b_i] \subset \mathbb{R}$ is a closed interval. This strategy is based on the implicit assumption that the search space can be covered efficiently by such boxes. This, however, is not the case if we have a more complex shaped space, e.g., the space of rotations (see Section 4). Second, the k-d tree used in [20] is updated only if the generated candidate is accepted. In the case of a rejection, the tree remains unchanged. This is a waste of computation time since the information gained by the expensive cost function evaluation is not used.

We account for the first drawback by formulating our minimization algorithm using a more general spatial data structure, namely, a generalized binary space partitioning tree (we will call it a G-BSP tree in the following). As opposed to the classic BSP trees (see, e.g., [22]), we do not require that the subspaces represented by the tree nodes are convex sets. Thus we can

minimize efficiently over more complex shaped search spaces like, e.g., the space of rotations (see Section 4). To avoid the second drawback, i.e., to use all the information gained by the cost function evaluation, we update the tree at every iteration—even in the cases of bad minimizer candidates. This apparently minor modification leads to a rather different algorithm (than [20]) and enables a faster rejection of the regions in which the cost function is likely to have high (i.e., poor) values and thus speeds up the convergence.

### *3.1. Generalized BSP Trees*

A binary space partitioning tree (BSP tree) is a spatial data structure which decomposes the real space $\mathbb{R}^n$ in a hierarchical manner. At each subdivision stage, the space is subdivided by a (hyper)plane in two disjoint parts of arbitrary size. Thus the resulting decomposition consists of arbitrarily shaped convex polygons [22]. Each node of the tree has exactly two or zero child nodes. A node with zero children is called a leaf. If we drop the assumption that the space subdivision is performed by planes we get a generalized BSP tree (G-BSP tree). This results in a decomposition made up of subspaces of arbitrary shape.

### *3.2. Problem Definition*

Given a set $\mathbf{X}$ (called the search space) and a function $f : \mathbf{X} \to \mathbb{R}$ our aim is to find a global minimizer of $f$, i.e., an $\mathbf{x}^* \in \mathbf{X}$ such that

$$f(\mathbf{x}^*) \leq f(\mathbf{x}) \quad \forall \mathbf{x} \in \mathbf{X}. \tag{18}$$

The following assumptions about $\mathbf{X}$ should hold:

- $\mathbf{X} \subset \mathbb{R}^n$ is a bounded set of positive volume (Lebesgue measure in $\mathbb{R}^n$).
- There is an algorithm of acceptable complexity which can build a G-BSP tree for $\mathbf{X}$ such that each two subsets of $\mathbf{X}$ at the same level of the tree are of equal volume (have the same Lebesgue measure in $\mathbb{R}^n$).
- $\mathbf{X}$ is simple enough for sampling algorithms of acceptable complexity to be able to sample uniformly from the G-BSP tree nodes, i.e., from the subsets of $\mathbf{X}$ represented in the G-BSP tree.

Furthermore, the cost function $f$ is required to be bounded and defined at each $\mathbf{x} \in \mathbf{X}$.

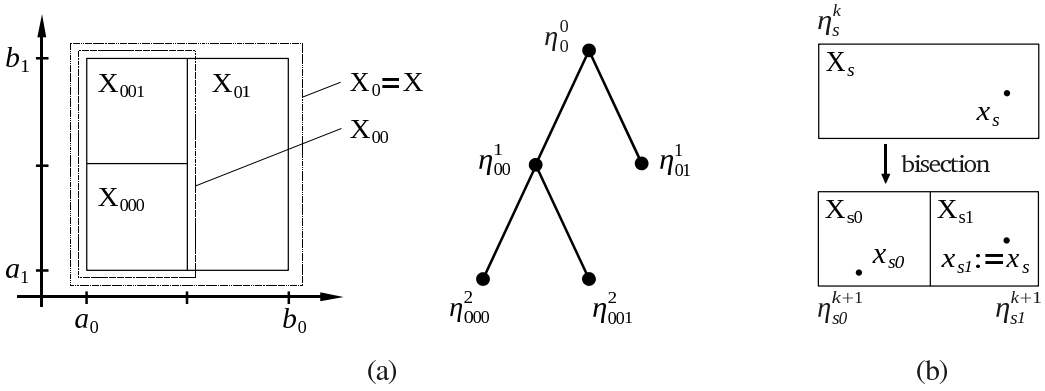



Figure 3: (a) An example of a two-dimensional G-BSP tree and a rectangular search space $\mathbf{X}$. In this case, the G-BSP tree is a two-dimensional k-d tree. (b) Expanding the leaf $\eta_s^k$. In this example, after the bisection of $\eta_s^k$, the point $\mathbf{x}_s$ lies in the box $\mathbf{X}_{s1}$, hence $\eta_{s1}^{k+1}$ adopts the pair $(\mathbf{x}_s, f(\mathbf{x}_s))$ from $\eta_s^k$. For the other child, $\eta_{s0}^{k+1}$, a point $\mathbf{x}_{s0}$ is sampled uniformly from $\mathbf{X}_{s0}$ and the objective function is evaluated at that point.

### *3.3. Overall Algorithm Description*

We use a G-BSP tree to represent the n-dimensional search space $\mathbf{X}$. The root $\eta_0^0$ is at the 0th level of the tree and represents the whole space $\mathbf{X}_0 = \mathbf{X}$. $\eta_0^0$ has two children, $\eta_{00}^1$ and $\eta_{01}^1$, which are at the next level. They represent the subsets $\mathbf{X}_{00}$ and $\mathbf{X}_{01}$, respectively, which are disjoint, have equal volume and their union equals $\mathbf{X}_0$. In general, a node $\eta_s^k$ (where $k \geq 0$ and $s$ is a binary string of length $k+1$) is at the $k$th level of the tree and has two children, $\eta_{s0}^{k+1}$ and $\eta_{s1}^{k+1}$, which are at the next, $(k+1)$th, level. The volume of $\eta_s^k$ is $1/2^k$ of the volume of $\mathbf{X}$. This concept is easily visualized in the case $n = 2$ and $\mathbf{X}$ and its subsets being rectangles (see Figure 3(a)).

During the minimization, the G-BSP tree is built in an iterative fashion beginning at the root. The algorithm adds more resolution to promising regions in the search space, i.e., the tree is built with greater detail in the vicinity of points in $\mathbf{X}$ at which the objective function attains low values. The overall procedure can be outlined as follows:

1. Initialize the tree (see Section 3.4) and set an iteration counter $j = 0$.
2. Select a "promising" leaf according to a probabilistic selection scheme (see Section 3.5).
3. Expand the tree by bisecting the selected leaf. This results in the creation of two new child nodes. Evaluate the objective function at a point which is uniformly sampled from the subset of one of the two children (see Section 3.6).

4. If a stopping criterion is not met, increment the iteration counter $j$ and go to step 2, otherwise terminate the algorithm (see Section 3.7).

### *3.4. Initializing the Tree*

For every tree node $\eta_s^k$ the following items are stored: (i) a set $\mathbf{X}_s \subset \mathbf{X}$ and (ii) a pair $(\mathbf{x}_s, f(\mathbf{x}_s))$ consisting of a point $\mathbf{x}_s$, uniformly sampled from $\mathbf{X}_s$, and the corresponding function value $f(\mathbf{x}_s)$. The tree is initialized by storing the whole search space $\mathbf{X}$ and a pair $(\mathbf{x}_0, f(\mathbf{x}_0))$ in the root.

### *3.5. Selecting a Leaf*

At every iteration, the search for a global minimizer begins at the root and proceeds down the tree until a leaf is reached. In order to reach a leaf, we have to choose a concrete path from the root down to this leaf. At each node, we have to decide whether to take its left or right child as the next station. This decision is made probabilistically. For every node, two numbers $p_0, p_1 \in (0, 1)$ are computed such that $p_0 + p_1 = 1$. Arriving at a node, we choose to descend via either its left or right child with probability $p_0$ or $p_1$, respectively. We make these left/right decisions until we reach a leaf.

*Computing the Probabilities $p_0$ and $p_1$* The idea is to compute the probabilities in a way such that the "better" child, i.e., the one with the lower function value, has greater chance to be selected. We compute $p_0$ and $p_1$ for each node $\eta_s^k$ based on the function values associated with its children $\eta_{s0}^{k+1}$ and $\eta_{s1}^{k+1}$. Let $f_{s0}$ and $f_{s1}$ be the function values associated with $\eta_{s0}^{k+1}$ and $\eta_{s1}^{k+1}$, respectively. The following criterion should be fulfilled:

$$f_{s0} < f_{s1} \quad \Leftrightarrow \quad p_0 > p_1. \tag{19}$$

If $f_{s0} < f_{s1}$ we set

$$p_0 = (t+1)/(1+2t), \quad p_1 = t/(1+2t), \tag{20}$$

for a parameter $t \geq 0$. For $t \to \infty$ we get $p_0 = p_1 = \frac{1}{2}$ and our minimization algorithm becomes a pure random search. Setting $t = 0$ results in $p_0 = 1$ and $p_1 = 0$ and makes the algorithm deterministically choosing the "better" child of every node which leads to the exclusion of a large portion of the search space and in most cases prevents the algorithm from finding a global minimizer. For $f_{s1} < f_{s0}$ we set

$$p_0 = t/(1+2t), \quad p_1 = (t+1)/(1+2t). \tag{21}$$

*Updating the Probabilities* From the discussion above it becomes evident that $t$ should be chosen from the interval $(0, \infty)$. For our algorithm the parameter $t$ plays a similar role as the temperature parameter for a simulated annealing algorithm [15] so we will refer to $t$ as temperature as well. Like in simulated annealing, the search begins at a high temperature level (large $t$) such that the algorithm samples the search space quite uniformly. The temperature is decreased gradually during the minimization process so that promising regions of the search space are explored in greater detail. More precisely, we update $t$ according to the following cooling schedule:

$$t = t_{\max} \exp(-vj), \tag{22}$$

where $j \in \mathbb{N}$ is the current iteration number, $t_{\max} > 0$ is the temperature at the beginning of the search (for $j = 0$) and $v > 0$ is the cooling speed which determines how fast the temperature decreases.

### *3.6. Expanding the Tree*

After reaching a leaf $\eta_s^k$, the set $\mathbf{X}_s$ associated with it gets bisected in two disjoint subsets $\mathbf{X}_{s0}$ and $\mathbf{X}_{s1}$ of equal volume. The corresponding child nodes are $\eta_{s0}^{k+1}$ and $\eta_{s1}^{k+1}$, respectively. In this way, we add more resolution in this part of the search space. Next, we evaluate the new children, i.e., we assign to the left and right one a pair $(\mathbf{x}_{s0}, f(\mathbf{x}_{s0}))$ and $(\mathbf{x}_{s1}, f(\mathbf{x}_{s1}))$, respectively.

Note that the parent of $\eta_{s0}^{k+1}$ and $\eta_{s1}^{k+1}$, namely, the node $\eta_s^k$, stores a pair $(\mathbf{x}_s, f(\mathbf{x}_s))$. Since $\mathbf{X}_s = \mathbf{X}_{s0} \cup \mathbf{X}_{s1}$ and $\mathbf{X}_{s0} \cap \mathbf{X}_{s1} = \emptyset$ it follows that $\mathbf{x}_s$ is contained either in $\mathbf{X}_{s0}$ or in $\mathbf{X}_{s1}$. Thus we set

$$(\mathbf{x}_{s0}, f(\mathbf{x}_{s0})) = (\mathbf{x}_s, f(\mathbf{x}_s)) \text{ if } \mathbf{x}_s \in \mathbf{X}_{s0} \text{ or} \tag{23}$$

$$(\mathbf{x}_{s1}, f(\mathbf{x}_{s1})) = (\mathbf{x}_s, f(\mathbf{x}_s)) \text{ if } \mathbf{x}_s \in \mathbf{X}_{s1}. \tag{24}$$

To compute the other pair, we sample a point uniformly from the appropriate remaining set ($\mathbf{X}_{s0}$ or $\mathbf{X}_{s1}$) and evaluate the function at this point (see Figure 3(b) for the case $n = 2$ and $\mathbf{X}$ and its subsets being rectangles).

*Updating the Tree* During the search we want to compute the random paths from the root down to a certain leaf such that promising regions—leaves with low function values—are visited more often than non-promising ones. Thus, after evaluating a new created leaf, we propagate its (possibly very low) function value as close as possible to the root. This is done by the following updating procedure. Suppose that the parent point $\mathbf{x}_s$ is contained in the set $\mathbf{X}_{s1}$ belonging to the new created child $\eta_{s1}^{k+1}$. Therefore, we randomly

generate $\mathbf{x}_{s0} \in \mathbf{X}_{s0}$, compute $f(\mathbf{x}_{s0})$ and assign the pair $(\mathbf{x}_{s0}, f(\mathbf{x}_{s0}))$ to the child $\eta_{s0}^{k+1}$. Updating the tree consists of ascending from $\eta_{s0}^{k+1}$ (via its ancestors) to the root and comparing at every parent node $\eta_u^j$ the function value $f(\mathbf{x}_{s0})$ with the function value of $\eta_u^j$, i.e., with $f(\mathbf{x}_u)$. If $f(\mathbf{x}_{s0}) < f(\mathbf{x}_u)$ we update the current node by setting $(\mathbf{x}_u, f(\mathbf{x}_u)) = (\mathbf{x}_{s0}, f(\mathbf{x}_{s0}))$ and proceed to the parent of $\eta_u^j$. The updating procedure terminates if we reach the root or no improvement for the current node is possible.

Note that if $f(\mathbf{x}_{s0})$ is the lowest function value found so far, it will be propagated to the root, otherwise it will be propagated only to a certain level $l \in \{1, \ldots, k+1\}$. This means, that every node contains the minimum function value (and the point at which $f$ takes this value) found in the subset associated with this node. Since the root represents the whole search space, it contains the point we are interested in, namely, the point at which $f$ takes the lowest value found up to the current iteration.

### *3.7. Stopping rule*

We break the search if the following two criteria are fulfilled. (i) The leaf $\eta_s^k$ selected in the current iteration has a volume which is smaller than a user predefined value $\delta_v > 0$. (ii) The absolute difference between the minimal function value found so far and the function value computed in the current iteration is less than a user specified $\delta_f > 0$.

The first condition accounts for the desired precision of the solution and the second one assures that the algorithm makes no significant progress any more.

## 4. Processing in the Space of Rigid Transforms

As already mentioned in Section 2.2, the choice of a parametrization of $SE(3)$ (the group of rigid transforms) is an important issue since different parametrizations lead to different optimization performance. We decompose $SE(3)$ into a translational and a rotational part. While parametrizing translations is straightforward special care is needed when dealing with rotations since the geometry of the rotation space is more complex than the geometry of $\mathbb{R}^3$. In the following, we concentrate on the rotation space.

In view of our branch and "stochastic bound" minimization method, three specific problems have to be solved. (i) We need to parametrize rotations. (ii) We have to hierarchically decompose the rotation space in disjoint parts of equal volume. In other words, a G-BSP tree has to be computed in which

the nodes are representing equally sized parts of the rotation space. (iii) We need to sample points (i.e., rotations) uniformly from each leaf of the G-BSP tree. These issues are discussed separately in the next three subsections.

### *4.1. Parametrizing Rotations*

There are many ways how to parametrize 3D rotations. Discussing all of them is far beyond the scope of this paper. An excellent introduction to this topic is included in the books by Kanatani [23] and A. Watt and M. Watt [24] in the context of computer vision and computer graphics, respectively. The set of all $3 \times 3$ rotation matrices is a group (under matrix multiplication) which is referred to as $SO(3)$. A parametrization of $SO(3)$ is a mapping $R : \mathbf{U} \rightarrow SO(3)$, where $\mathbf{U}$ is a subset of $\mathbb{R}^3$ since every rotation has three degrees of freedom.

Parametrizing rotation matrices using Euler angles is probably the most widely used technique which is, however, inefficient in conjunction with our minimization method. This is due to the fact that Euler angles are a redundant representation of rotations. In order to represent all elements in $SO(3)$ the following range, $\mathbf{E}$, for the three Euler angles is needed: $\mathbf{E} = [0, 2\pi) \times [0, 2\pi) \times [0, \pi]$. However, the corresponding parametrization $R : \mathbf{E} \rightarrow SO(3)$, which is given in [23], is not one-to-one. There are infinitely many combinations of Euler angles (within the range $\mathbf{E}$) which lead to the same rotation matrix (see [24]). A minimization method like ours which considers the whole search space will waste computation time exploring regions in $\mathbf{E}$ which should be completely ignored because they do not lead to "new" rotation matrices. The same applies to deterministic branch-and-bound methods (see, e.g., [25]).

In order to avoid this difficulty, we employ a redundant-free rotation space parametrization based on the axis-angle representation of $SO(3)$. According to Euler's theorem (see [23]), each rotation in $\mathbb{R}^3$ can be represented by an axis specified by a unit vector $\mathbf{n}$ and an angle $\theta$ of rotation around it. $\mathbf{n}$ can itself be parametrized using spherical coordinates $\varphi$ and $\psi$:

$$\mathbf{n} = (\sin(\psi)\cos(\varphi),\ \sin(\psi)\sin(\varphi),\ \cos(\psi)). \tag{25}$$

Figure 4(a) visualizes this concept. In order to represent all rotation matrices, we need to consider the following range for the spherical coordinates $(\varphi, \psi)$ and the rotation angle $\theta$:

$$(\varphi, \psi, \theta) \in [0, 2\pi) \times [0, \pi] \times [0, \pi) = \mathbf{A}. \tag{26}$$

The parametrization $R : \mathbf{A} \rightarrow SO(3)$, which can be found in [23], is a one-to-one mapping between $\mathbf{A}$ and $SO(3)$.

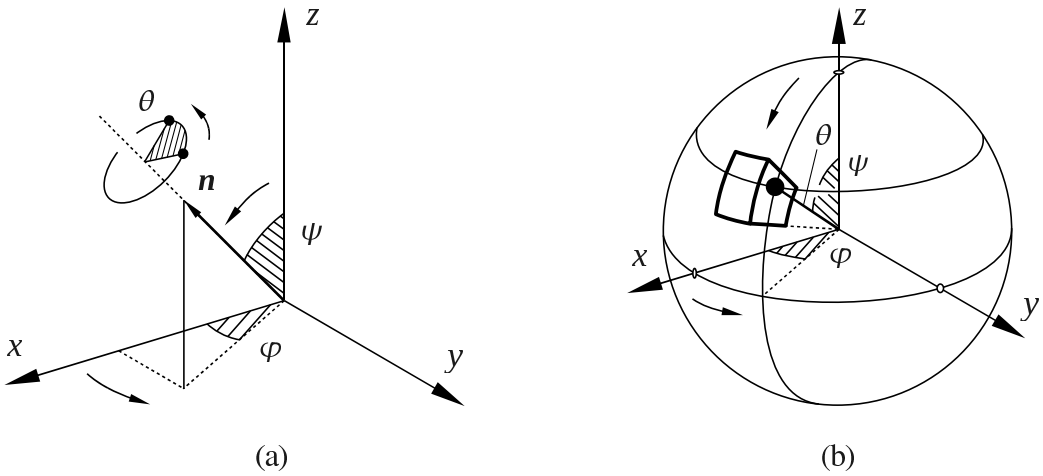



Figure 4: (a) The axis-angle based parametrization of $SO(3)$. The two bold dots in the figure represent a point before and after rotation by the angle $\theta$ around the axis defined by the unit vector $\mathbf{n}$, which is itself parametrized using spherical coordinates $(\varphi, \psi)$. (b) The rotation space represented as the open ball in $\mathbb{R}^3$ with radius $\pi$. The spherical coordinates $(\varphi, \psi)$ of the point (shown as a bold dot) define the rotation axis and the distance to the origin gives the angle of rotation $\theta$. The bold lines depict a spherical box.

### *4.2. Hierarchical Decomposition of the Rotation Space*

According to the axis-angle representation and to (26), it is possible to express the set of rotations by the open ball in $\mathbb{R}^3$ with radius $\pi$ which we will denote by $\mathbf{B}^3(\pi)$ (see Figure 4(b)). Thus a straightforward way to decompose the rotation space is to enclose $\mathbf{B}^3(\pi)$ in the cube $\mathbf{C}^3(\pi) = [-\pi, \pi]^3$ and to divide $\mathbf{C}^3(\pi)$ into smaller cubes by simply bisecting the $x$, $y$ or $z$ axis. Hartley and Kahl [25] used this technique in conjunction with a deterministic branch-and-bound minimization method to estimate the essential matrix and to solve the relative camera pose problem. However, if combined with our minimization algorithm, this technique leads to two problems. First, the sub-cubes of $\mathbf{C}^3(\pi)$ which do not lie within $\mathbf{B}^3(\pi)$ have to be ignored since the rotations they represent are included in other cubes within $\mathbf{B}^3(\pi)$. This gives rise to nodes in the corresponding G-BSP tree which have only one "legal" child. Second, the sub-cubes of $\mathbf{C}^3(\pi)$ which are partially intersecting $\mathbf{B}^3(\pi)$ represent a smaller region of the rotation space than sub-cubes at the same tree level which are fully enclosed in $\mathbf{B}^3(\pi)$. Thus the minimization algorithm will prefer rotations which are close to the boundary of $\mathbf{B}^3(\pi)$.

We solve these two problems by changing the shape of the building blocks

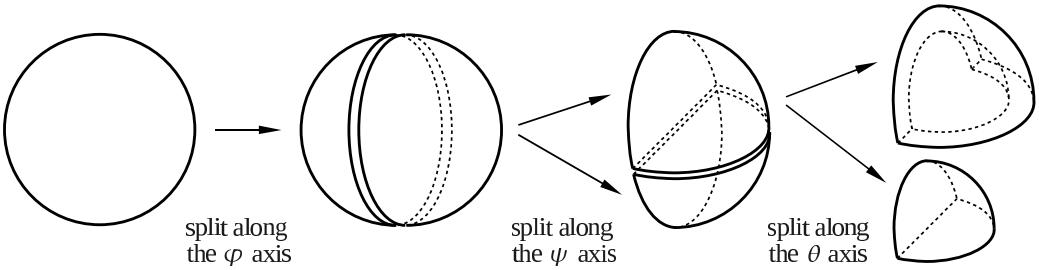



Figure 5: Decomposing the rotation space (represented as $\mathbf{B}^3(\pi)$) into spherical boxes of equal volume. In this example, only one spherical box at each splitting step is further decomposed.

of the decomposition. Since we are dealing with a three-dimensional ball the most natural shape is the shape of a spherical box (see Figure 4(b)). In ball coordinates, we define a spherical box $\mathbf{S}^3$ to be a point set of the form

$$\mathbf{S}^3 = \{(\varphi, \psi, \theta) : (\varphi, \psi, \theta) \in [\varphi_1, \varphi_2) \times [\psi_1, \psi_2) \times [\theta_1, \theta_2)\}, \tag{27}$$

where $[\varphi_1, \varphi_2) \times [\psi_1, \psi_2)$ is the range of the spherical coordinates and $[\theta_1, \theta_2)$ limits the distance of the points to the origin. Decomposing the rotation space means to hierarchically subdivide $\mathbf{B}^3(\pi)$ into disjoint spherical boxes of equal volume (see Figure 5). Note that the volume of $\mathbf{S}^3$ is given by

$$vol_{\mathbf{S}^3}(\varphi_1, \varphi_2, \psi_1, \psi_2, \theta_1, \theta_2) = \int_{\varphi_1}^{\varphi_2} \int_{\psi_1}^{\psi_2} \int_{\theta_1}^{\theta_2} \theta^2 \sin\psi d\theta d\psi d\varphi \tag{28}$$

$$= (\varphi_2 - \varphi_1)(\cos\psi_1 - \cos\psi_2)\frac{\theta_2^3 - \theta_1^3}{3}. \tag{29}$$

Our aim is to consecutively cut $\mathbf{S}^3$ along the $\varphi$, $\psi$ or $\theta$ axis such that the resulting pieces have the same volume. Since $vol_{\mathbf{S}^3}$ depends in a different way from each of the ball coordinates $\varphi$, $\psi$ and $\theta$ we get a different rule for cutting along each axis. We are looking for

$$\varphi \in (\varphi_1, \varphi_2), \quad \psi \in (\psi_1, \psi_2), \quad \theta \in (\theta_1, \theta_2) \tag{30}$$

such that

$$vol_{\mathbf{S}^3}(\varphi_1, \varphi) = vol_{\mathbf{S}^3}(\varphi, \varphi_2), \tag{31}$$

$$vol_{\mathbf{S}^3}(\psi_1, \psi) = vol_{\mathbf{S}^3}(\psi, \psi_2), \tag{32}$$

$$vol_{\mathbf{S}^3}(\theta_1, \theta) = vol_{\mathbf{S}^3}(\theta, \theta_2), \tag{33}$$

where, for the sake of clarity, $vol_{\mathbf{S}^3}$ is expressed as a function of two variables only, namely, the ones defining the interval which is currently cut. Using (29) to solve the equations (31)–(33) leads to

$$\varphi = \frac{\varphi_1 + \varphi_2}{2}, \quad \psi = \arccos\left(\frac{\cos\psi_1 + \cos\psi_2}{2}\right), \quad \theta = \sqrt[3]{\frac{\theta_1^3 + \theta_2^3}{2}}. \tag{34}$$

Thus we fully specified how to hierarchically decompose the space of rotations in disjoint equally sized parts such that a G-BSP tree can be built. Furthermore, the shape of the parts is optimally tailored to our minimization algorithm.

### *4.3. Uniform Sampling from Spherical Boxes*

Our method for sampling points uniformly from a spherical box is grounded on the following basic result from Statistics called the inverse probability integral transform. Since it is proved in many textbooks (like, e.g., in [26]) we state it here without a proof.

**Theorem 1.** *Let $F$ be a cumulative distribution function (c.d.f.) on $\mathbb{R}$ and let $U$ be a random variable uniformly distributed in $[0, 1]$. Then the random variable $X = F(U)^{-1}$ has c.d.f. $F$.*

Based on this result we perform the uniform sampling from a spherical box $\mathbf{S}^3 = [\varphi_1, \varphi_2) \times [\psi_1, \psi_2) \times [\theta_1, \theta_2)$ in three steps:

1. Sample a $\varphi$ uniformly from $[\varphi_1, \varphi_2)$.
2. Sample a $\psi$ from $[\psi_1, \psi_2)$ according to a c.d.f. $F_2$ such that the point in $\mathbb{R}^3$ with spherical coordinates $(\varphi, \psi)$ is uniformly distributed on the spherical patch $\mathbf{S}^2 = [\varphi_1, \varphi_2) \times [\psi_1, \psi_2)$.
3. Sample a $\theta$ from $[\theta_1, \theta_2)$ according to a c.d.f. $F_3$ such that the point in $\mathbb{R}^3$ with ball coordinates $(\varphi, \psi, \theta)$ is uniformly distributed in the spherical box $\mathbf{S}^3$.

Step 1 is easy to perform. In step 2, we need to compute the area of a spherical patch (of the unit 2-sphere) as a function of an interval $[\varphi_1, \varphi_2) \times [\psi_1, \psi_2)$:

$$area_{\mathbf{S}^2}(\varphi_1, \varphi_2, \psi_1, \psi_2) = \int_{\varphi_1}^{\varphi_2} \int_{\psi_1}^{\psi_2} \sin\psi d\psi d\varphi \tag{35}$$

$$= (\varphi_2 - \varphi_1)(\cos\psi_1 - \cos\psi_2). \tag{36}$$

Thus the c.d.f. we need in step 2 is given by

$$F_2(\psi) = \frac{area_{\mathbf{S}^2}(\varphi_1, \varphi_2, \psi_1, \psi)}{area_{\mathbf{S}^2}(\varphi_1, \varphi_2, \psi_1, \psi_2)} \tag{37}$$

$$= \frac{\cos\psi_1 - \cos\psi}{\cos\psi_1 - \cos\psi_2}, \tag{38}$$

Analogously, we see that the c.d.f. in step 3 is given by

$$F_3(\theta) = \frac{vol_{\mathbf{S}^3}(\varphi_1, \varphi_2, \psi_1, \psi_2, \theta_1, \theta)}{vol_{\mathbf{S}^3}(\varphi_1, \varphi_2, \psi_1, \psi_2, \theta_1, \theta_2)} \tag{39}$$

$$= \frac{\theta^3 - \theta_1^3}{\theta_2^3 - \theta_1^3}, \tag{40}$$

where (40) follows from (29). Note that both $F_2$ and $F_3$ can easily be inverted and we can use Theorem 1 to sample according to $F_2$ and $F_3$ and hence uniformly from the spherical box $\mathbf{S}^3$.

*4.4. Computing the Search Space and the G-BSP Tree*

Now since all details regarding the parametrization and decomposition of $SO(3)$ and the sampling from spherical boxes are given, we define the search space $\mathbf{X}$ and specify how to build the corresponding G-BSP tree. We set

$$\mathbf{X} = \mathbf{A} \times bbox(\mathbf{M}), \tag{41}$$

where $\mathbf{A}$ is, according to (26), the domain of the axis-angle based parametrization of $SO(3)$ and $bbox(\mathbf{M})$ (the bounding box of the model $\mathbf{M}$) represents the translational part of the search space. Since $bbox(\mathbf{M})$ is a rectangular box of the form $[x_1, x_2] \times [y_1, y_2] \times [z_1, z_2] \subset \mathbb{R}^3$ it can easily be broken up into smaller boxes of the same size by simply bisecting it along the $x$, $y$ or $z$ axis.

The root $\eta_0^0$ of the G-BSP tree represents the whole set $\mathbf{X}$. The child nodes of the root, namely, $\eta_{00}^1$ and $\eta_{01}^1$, represent the subsets $\mathbf{X}_0$ and $\mathbf{X}_1$, respectively, resulting from cutting the 0th interval of $\mathbf{X}$—which is $[0, 2\pi)$ in (26)—using the rule $(34)_1$. In general, a node $\eta_s^k$ (where $k \geq 0$ and $s$ is a binary string of length $k+1$) is at the $k$th level of the tree, represents a subset $\mathbf{X}_s$ of the 6D search space and has two children, $\eta_{s0}^k$ and $\eta_{s1}^k$. The child nodes represent the sets $\mathbf{X}_{s0}$ and $\mathbf{X}_{s1}$, respectively, which are computed by cutting the ($k$ mod 6)th interval of $\mathbf{X}_s$ according to (34) if $0 \leq k \bmod 6 \leq 2$ (rotational part) or by dividing it in the middle if $3 \leq k \bmod 6 \leq 5$ (translational part).

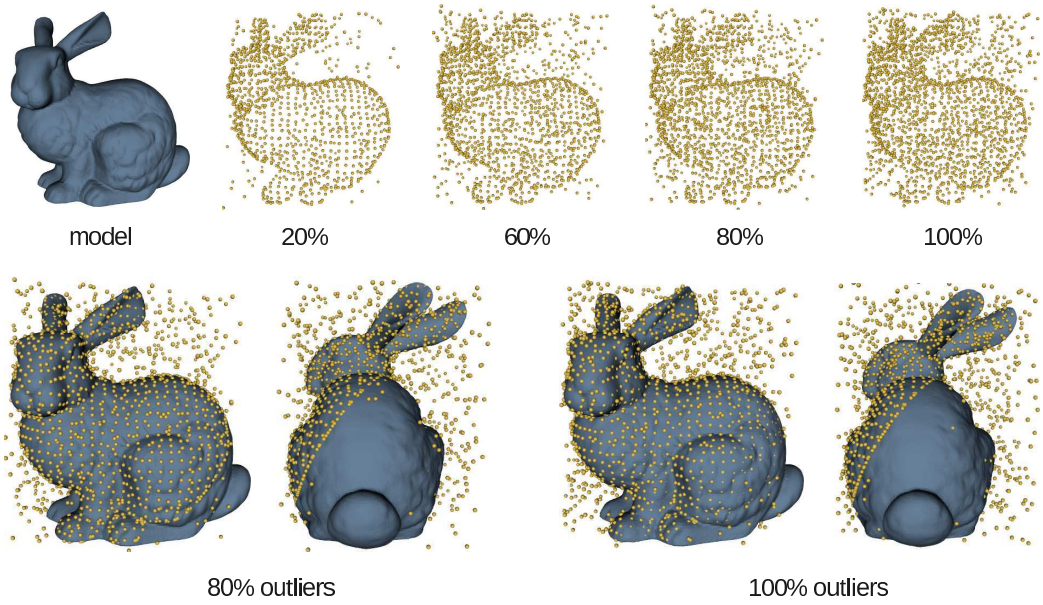



Figure 6: (Top row) The model set is shown as a blue mesh (note that only the mesh vertices are used for the registration). The outlier corrupted data sets are rendered as yellow point clouds. The number of outliers as percentage of the original number of input points is shown below each figure. Further note that the data sets are incomplete and sparsely sampled compared to the model. (Bottom row) Typical registration results obtained with our algorithm using the model scalar field (13) based on the inverse distance kernel (12). Observe the high quality of the alignment even in the presence of a significant amount of outliers.

## 5. Experimental Results

In this Section, we test our registration method on a variety of point sets. Since our algorithm is a probabilistic one, it computes each time a (slightly) different result. In order to make a statistical meaningful statement about its performance, we run 100 registration trials for each pair of inputs and report the mean performance values. We measure the success rate and the accuracy under varying amount of noise and outliers in the input sets. The success rate gives the percentage of registration trials in which a transform which is close to the global optimal one is found. The accuracy is measured using the RMS error (see [6]). The type of noise added to some of the model and data sets is Gaussian and the outliers are simulated by drawing points from a uniform distribution within the bounding box of the corresponding point set. We also measure the number of cost function evaluations and the computation time

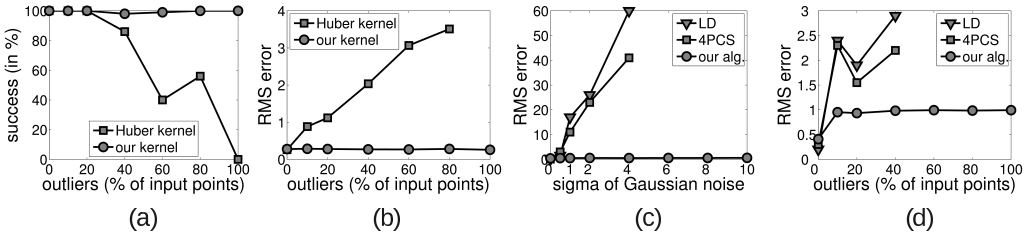



Figure 7: (a) The success rate as a function of the percentage of outliers in the data sets shown in Figure 6. The success rate of the registration is shown when using the inverse distance kernel (12) (our kernel) and the Huber kernel (7). Note that our kernel leads to an almost constant success rate of 100% even in the presence of a very large amount of outliers whereas at the level of 100% outliers the registration completely fails if the Huber kernel is used. (b) The RMS error between the ground truth pose for each data set and the estimated pose is shown as a function of the percentage of outliers. Only the successful trials are used for computing the RMS error. Note that our kernel leads to much more precise registration results which are almost independent of the amount of outliers. (c), (d) We compare our method with the robust 4PCS algorithm [7] and a local descriptor based approach (LD). A combination of a spin-image based descriptor and integral invariants are used as local descriptors (see [7]). Note that the graphs corresponding to LD and 4PCS end by $\sigma = 4.0$ and 40% outliers. This is because the authors in [7] did not test their methods on point sets with more noise or outliers whereas we did. Observe that our algorithm is quite insensitive to noise and outliers and it outperforms both other methods. The alignment error is measured using the RMS error between the model and the data after registration. One unit corresponds to 1% of the bounding box diagonal length of the model set.

for varying cooling speed $v$ (defined in (22)). We compare the robustness of our method using two different kernels in the cost function. Furthermore, we report how two state-of-the-art registration approaches perform on the same point sets. In the following, we describe each test scenario in detail.

First, the success rate and the accuracy of our method are tested with two different kernels, namely, the inverse distance kernel (12) used in our cost function and the Huber kernel (7) used in [14]. The point sets used in this test together with some typical registration results are shown in Figure 6. Note that outliers are added to the data set only and it is a subset of the model. This case occurs in real world scenarios in which one has a complete (relatively clean) model of an object and wants to align it to a low quality data set which only partially represents the object (due to visibility issues like, e.g., occlusion and scene clutter). As already mentioned in Section 2.1, we expect a registration method which minimizes a cost function based on

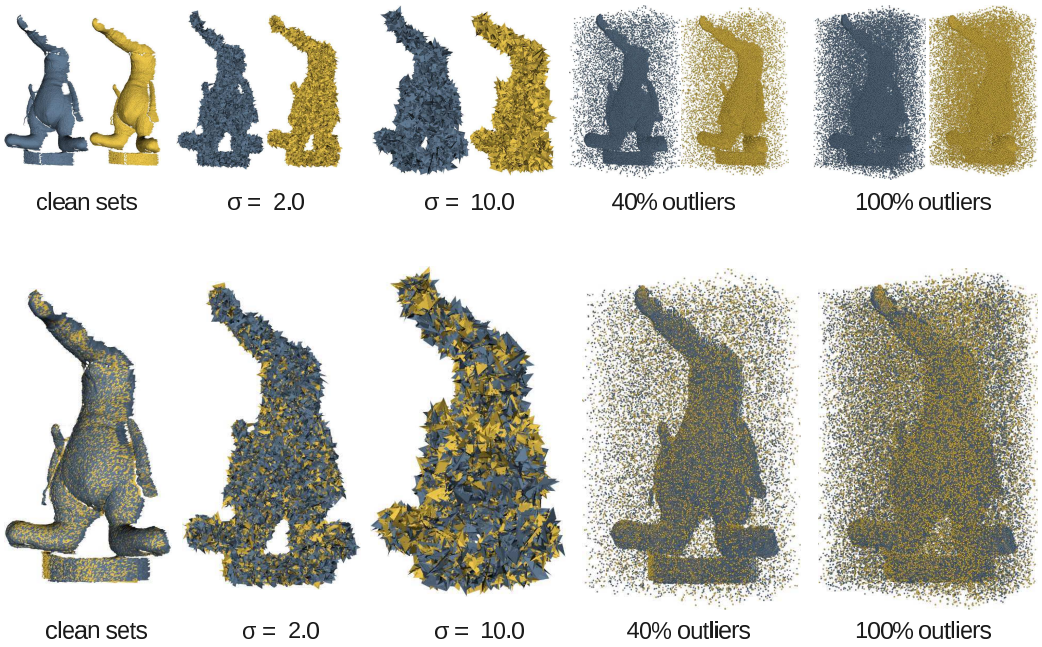



Figure 8: Registration of partially overlapping noisy and outlier corrupted point sets. The models are shown in blue whereas the data sets in yellow. (Top row) Partial scans of the Coati model degraded by noise or outliers. The $\sigma$ of the Gaussian noise or the amount of outliers as percentage from the original number of input points is indicated below each figure. One $\sigma$ unit equals 1% of the bounding box diagonal length of the corresponding point set. (Bottom row) Typical registration results computed with our algorithm. The results are obtained without any preprocessing of the input, ICP refinement [1] or assumptions about the initial pose of the point sets.

the (unbounded) Huber kernel to have difficulties with outlier corrupted data sets. This is confirmed by the results of this test case which are summarized in the Figures 7(a) and 7(b).

In the second test case, we align two partially overlapping parts of the Coati model under varying conditions. This time, noise and outliers are added to both the model and the data set. This situation occurs in practice when building a complete object model out of multiple partially overlapping scans. We compare our results with the ones reported in [7] which are obtained with the robust 4PCS algorithm and a state-of-the-art local descriptor based approach. We perform the tests on the same point sets which are used in [7]. This allows for a precise comparison without the need of re-implementing neither of the two algorithms. The model and data sets

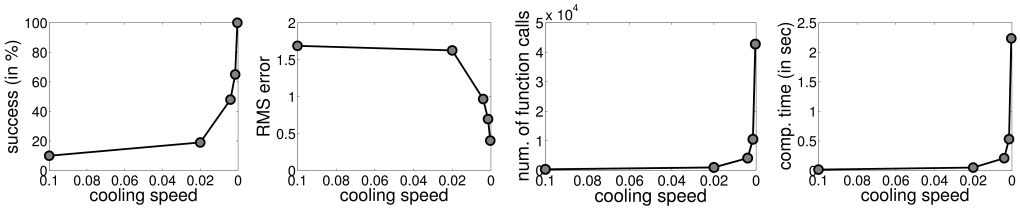



Figure 9: From left to right: success rate, RMS error, number of cost function evaluations and computation time of our registration algorithm as a function of the cooling speed $v$ (defined in (22)). Model and data used in this test case are copies of the outlier-free version of the data set shown in the top row of Figure 6. One RMS error unit equals 1% of the bounding box diagonal length of the point set. All tests presented in this paper were performed on a 3GHz laptop and the algorithm is implemented in C++.

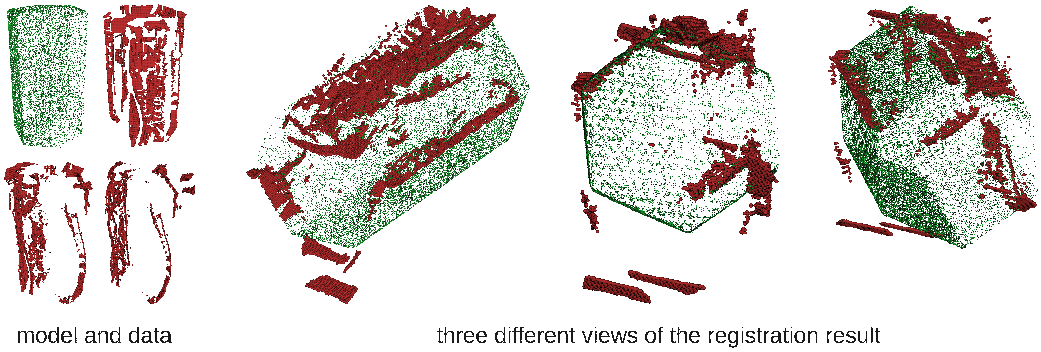



Figure 10: (Left) The complete model of a box (shown in green) and three views of the very low quality data set (shown in red). The data was obtained by a correlation based stereo algorithm under poor lighting conditions. (Right) Our method robustly achieves the visually right alignment. The high amount of noise and outliers which almost completely destroy the shape of the object makes this a challenging example.

together with some typical registration results obtained with our method are shown in Figure 8. In the Figures 7(c) and 7(d) we plot our results together with the ones reported in [7].

Next, we measure the performance of our algorithm for varying cooling speed $v$ defined in (22). We report the results in Figure 9. Our algorithm achieves a success rate of 100% and an RMS error below 0.5 for less than 2.5 seconds.

Finally, we demonstrate the ability of our method to deal with partially overlapping and very sparsely sampled point sets corrupted by noise and outliers which are not artificially generated but originate in scan device im-

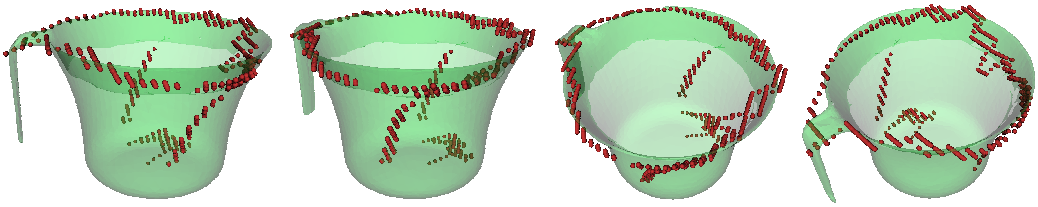


Figure 11: Registration result in the case of a noisy and very sparsely reconstructed data set (shown by the red "curve") and a complete noise-free model (transparent green mesh). Note that in this case the state-of-the-art integral volume descriptor (used in [6]) will fail since the curve which represents the data set does not enclose a volume in $\mathbb{R}^3$. Local descriptors which use surface normals like, e.g., spin images [5] will fail as well since in general the normal of a curve which lies on a surface does not match the surface normal.

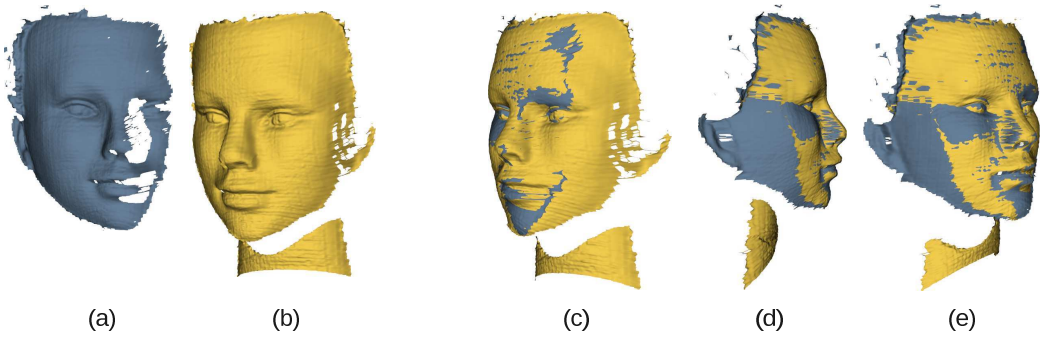



Figure 12: Registration of noisy point sets with low overlap. Although rendered as meshes only points are used for the registration. Note that the input scans, (a) and (b), represent different parts of the face and the model set, shown in (a), contains no parts of the neck. (c) – (e) A typical registration result obtained with our method shown from three different viewpoints.

precision. In Figure 10, we show that our method successfully computes the right registration even in the case of an extremely degraded data set which represents only a subset of the model. Figure 11 illustrates the stability of our algorithm when dealing with very sparsely sampled data sets. Figures 1 and 12 show typical registration results for partially overlapping points sets.

Note that our registration method could lead to incorrect results for a class of shapes for which several almost equally good alignments exist and the registration ambiguity can be dissolved by small scale features only. An example of such a shape is a large cup with a small handle. In this case, the corresponding point sets lead to a cost function with several local minima which are almost as "good" as the global one (see Figure 13).

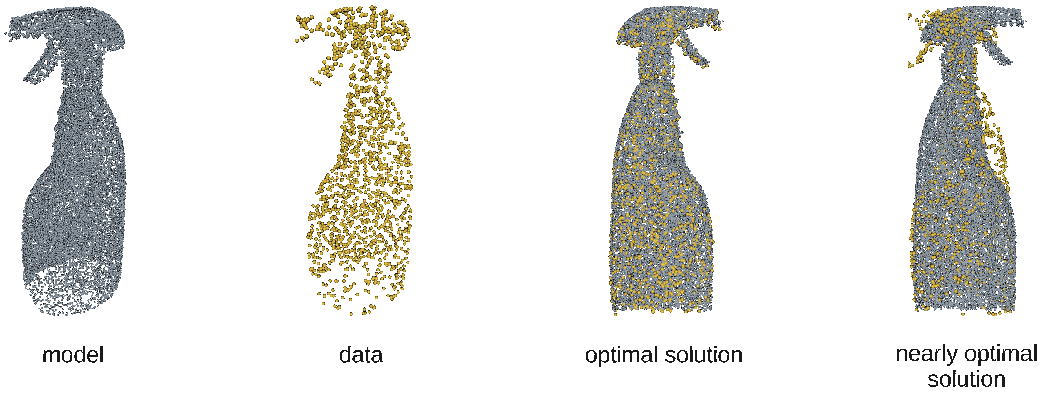



Figure 13: Point sets leading to a cost function which has two almost equally low minima. The nearly optimal solution differs from the optimal one by a rotation of the data set by 180° about the axis which corresponds to the upright orientation of the bottle.

## 6. Conclusions

We introduced a new technique for pairwise rigid registration of point sets. Our method is based on a noise robust and outlier resistant cost function which itself is based on an inverse distance kernel. One of the main messages of the paper is that a registration method which minimizes an objective function based on an unbounded kernel will be sensitive to outliers in the point sets. This was fully validated by comparisons between our kernel and the Huber kernel which were presented in the experimental part of the paper.

A further property of our algorithm is that it does not rely on any initial estimation of the globally optimal rigid transform. This was achieved by employing a new stochastic algorithm for global optimization. In order to minimize efficiently over complex shaped search spaces like the space of rotations we generalized the BSP trees and introduced a new technique for hierarchical rotation space decomposition. Furthermore, we derived a new procedure for uniform point sampling from spherical boxes.

Tests on a variety of point sets showed that the proposed method is insensitive to noise and outliers and can cope very well with sparsely sampled and incomplete data sets. Comparisons showed that our algorithm outperforms a recently proposed generate-and-test approach and a state-of-the-art local descriptor based method in terms of accuracy and robustness.

## References


[1] P. Besl, N. McKay, A Method for Registration of 3-D Shapes, IEEE Trans. PAMI 14 (1992).

[2] Y. Hecker, R. Bolle, On Geometric Hashing and the Generalized Hough Transform, IEEE Trans. on Systems, Man, and Cybernetics 24 (1994).

[3] H. Wolfson, I. Rigoutsos, Geometric Hashing: an Overview, Computational Science & Engineering, IEEE 4 (1997) 10–21.

[4] G. Stockman, Object Recognition and Localization via Pose Clustering, Computer Vision, Graphics, and Image Processing 40 (1987) 361–387.

[5] A. Johnson, M. Hebert, Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes, IEEE Trans. PAMI 21 (1999) 433–449.

[6] N. Gelfand, N. Mitra, L. Guibas, H. Pottmann, Robust Global Registration, Eurographics Symposium on Geometry Processing (2005) 197–206.

[7] D. Aiger, N. Mitra, D. Cohen-Or, 4-Points Congruent Sets for Robust Pairwise Surface Registration, ACM Trans. Graph. 27 (2008).

[8] Y. Chen, G. Medioni, Object Modeling by Registration of Multiple Range Images, Robotics and Automation, Proceedings., IEEE International Conference on 3 (1991) 2724–2729.

[9] S. Rusinkiewicz, M. Levoy, Efficient Variants of the ICP Algorithm, 3DIM (2001) 145–152.

[10] T. M. Breuel, Implementation techniques for geometric branch-and-bound matching methods, Computer Vision and Image Understanding 90 (2003) 258–294.

[11] C. Olsson, F. Kahl, M. Oskarsson, Branch-and-Bound Methods for Euclidean Registration Problems, IEEE Trans. PAMI 31 (2009) 783–794.

[12] N. Mitra, N. Gelfand, H. Pottmann, L. Guibas, Registration of Point Cloud Data from a Geometric Optimization Perspective, Symposium on Geometry Processing (2004) 23–32.

[13] H. Pottmann, Q.-X. Huang, Y.-L. Yang, S.-M. Hu, Geometry and Convergence Analysis of Algorithms for Registration of 3D Shapes, International Journal of Computer Vision 67 (2006) 277–296.

[14] A. W. Fitzgibbon, Robust registration of 2D and 3D point sets, Image Vision Comput. 21 (2003) 1145–1153.

[15] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, Equation of State Calculations by Fast Computing Machines, The Journal of Chemical Physics 21 (1953) 1087–1092.

[16] V. Cerny, Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm, Journal of Optimization Theory and Applications 45 (1985) 41–51.

[17] S. Kirkpatrick, C. Gelatt, M. Vecchi, Optimization by Simmulated Annealing, Science 220 (1983) 671–680.

[18] P. Pardalos, E. Romeijn (Eds.), Handbook of Global Optimization 2, Nonconvex Optimization and Its Applications, Kluwer Academic Publishers, 2002.

[19] D. Bulger, G. Wood, Hesitant Adaptive Search for Global Optimisation, Math. Program. 81 (1998) 89–102.

[20] G. Bilbro, W. Snyder, Optimization of Functions with Many Minima, IEEE Trans. on Systems, Man, and Cybernetics 21 (1991) 840–849.

[21] C. Papazov, D. Burschka, Stochastic Optimization for Rigid Point Set Registration, in: Advances in Visual Computing, 5th International Symposium, Proceedings, volume 5875 of *Lecture Notes in Computer Science*, Springer, 2009, pp. 1043–1054.

[22] H. Samet, The Design and Analysis of Spatial Data Structures, Addison-Wesley, 1990.

[23] K. Kanatani, Group-Theoretical Methods in Image Understanding, Springer Series in Information Sciences, Springer, 1990.

[24] A. Watt, M. Watt, Advanced Animation and Rendering Techniques, Addison-Wesley, 1992.

[25] R. I. Hartley, F. Kahl, Global Optimization through Rotation Space Search, International Journal of Computer Vision 82 (2009) 64–79.

[26] N. Madras, Lectures on Monte Carlo Methods, American Mathematical Society, 2002.

# An Efficient RANSAC for 3D Object Recognition in Noisy and Occluded Scenes

Chavdar Papazov and Darius Burschka

Technische Universität München (TUM), Germany
email: {papazov, burschka}@in.tum.de

**Abstract.** In this paper, we present an efficient algorithm for 3D object recognition in presence of clutter and occlusions in noisy, sparse and unsegmented range data. The method uses a robust geometric descriptor, a hashing technique and an efficient RANSAC-like sampling strategy. We assume that each object is represented by a model consisting of a set of points with corresponding surface normals. Our method recognizes multiple model instances and estimates their position and orientation in the scene. The algorithm scales well with the number of models and its main procedure runs in linear time in the number of scene points. Moreover, the approach is conceptually simple and easy to implement. Tests on a variety of real data sets show that the proposed method performs well on noisy and cluttered scenes in which only small parts of the objects are visible.

## 1 Introduction

Object recognition is one of the most fundamental problems of computer vision. In recent years, advances in 3D geometry acquisition technology have led to a growing interest in object recognition techniques which work with three-dimensional data. Referring to [1], the 3D object recognition problem can be stated as follows. Given a set $\mathcal{M} = \{\mathbf{M}_1, \ldots, \mathbf{M}_q\}$ of models and a scene $\mathbf{S}$ are there transformed subsets of some models which match a subset of the scene? The output of an object recognition algorithm is a set $\{(\mathbf{M}_{k_1}, T_1), \ldots, (\mathbf{M}_{k_r}, T_r)\}$ where $\mathbf{M}_{k_j} \in \mathcal{M}$ is a recognized model instance and $T_j$ is a transform which aligns $\mathbf{M}_{k_j}$ to the scene $\mathbf{S}$. In this paper, we discuss a special instance of this problem which is given by the following assumptions.

(i) Each model $\mathbf{M}_i$ is a finite set of oriented points, i.e., $\mathbf{M}_i = \{(\mathbf{p}, \mathbf{n}) : \mathbf{p} \in \mathbb{R}^3, \mathbf{n} \text{ is the normal at } \mathbf{p}\}$.
(ii) Each model is representing a non-transparent object.
(iii) The scene $\mathbf{S} = \{\mathbf{p}_1, \ldots, \mathbf{p}_s\} \subset \mathbb{R}^3$ is a range image.
(iv) The transform $T_j$ which aligns $\mathbf{M}_{k_j}$ to $\mathbf{S}$ is a rigid transform.

Even under these assumptions the problem remains hard because of several reasons: it is a priori not known which objects are in the scene and how they are oriented; the scene points are typically corrupted by noise and outliers; the

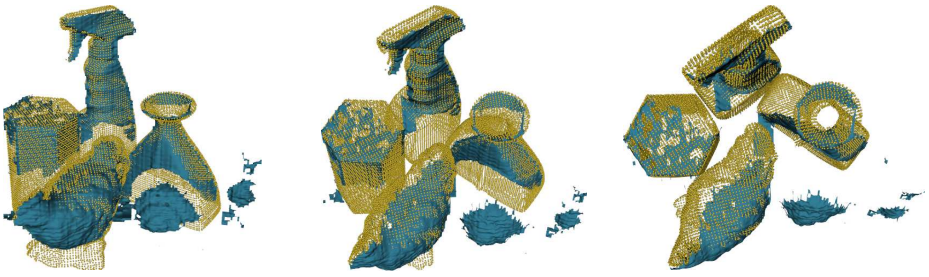


**Fig. 1.** Three views of a typical recognition result obtained with our method. The scene is shown as a blue mesh and the four recognized model instances are rendered as yellow point clouds and superimposed over the scene mesh (see Section 4 for details).

objects are only partially visible due to scene clutter, occlusions and scan device limitations.

**Contributions and Overview** In this paper, we introduce an efficient algorithm for 3D object recognition in noisy, sparse and unsegmented range data. We make the following contributions: (i) We use a hash table for rapid retrieval of pairs of oriented model points which are similar to a sampled pair of oriented scene points. (ii) A new efficient RANSAC-like sampling strategy for fast generation of object hypotheses is introduced. (iii) We provide a complexity analysis of our sampling strategy and derive the number of iterations needed to recognize model instances with a predefined success probability. (iv) A new measure for the quality of an object hypothesis is presented. (v) We use a non-maximum suppression to remove false positives and to achieve a consistent scene explanation by the given models.

The rest of the paper is organized as follows. After reviewing previous work in Section 2, we describe our algorithm in Section 3. Section 4 presents experimental results. Conclusions are drawn in the final Section 5 of the paper.

## 2 Related Work

Object recognition should not be confused with object classification/shape retrieval. The latter methods only measure the similarity between a given input shape and shapes stored in a model library. They do not estimate a transform which maps the input to the recognized model. Moreover, the input shape is assumed to be a subset of some of the library shapes. In our case, however, the input contains points originating from multiple objects and scene clutter.

There are two major classes of 3D object recognition methods. One class consists of the so-called voting methods. Well-known representatives are the generalized Hough transform [2] and geometric hashing [1]. The generalized Hough transform has a favorable space and time complexity of $O(nk^3)$, where $n$ is the number of scene points and $k$ is the number of bins for each dimension of the discretized rotation space. Unfortunately, the method scales bad with the num-

ber of models since one has to match sequentially each one of them to the scene. The geometric hashing approach [1] allows for a simultaneous recognition of all models without the need of sequential matching. However, it tends to be very costly since its space complexity is $O(m^3)$ and its worse case time complexity is $O(n^4)$, where $m$ and $n$ are the number of model and scene points, respectively. A more recent voting approach is the tensor matching algorithm [3]. It performs well on complex scenes but the authors did not present tests on noisy and sparse data sets.

The correspondence based methods belong to the second class of object recognition approaches. First, correspondences between the models and the scene are established usually using local geometric descriptors. In the second step, the aligning rigid transform is calculated based on the established correspondences. There is a vast variety of descriptors which can be used in a correspondence based object recognition framework. A list includes, without being nearly exhaustive, spin images [4], local feature histograms [5], 3D shape context, harmonic shape context [6] and integral invariants [7]. In [8], classic 2D image descriptors were extended to the domain of 2-manifolds embedded in $\mathbb{R}^3$ and applied to rigid and non-rigid matching of meshes. Intrinsic isometry invariant descriptors were developed in [9] and shown to be effective for the matching of articulated shapes. All correspondence based algorithms rely heavily on the assumption that the models to be recognized have distinctive feature points, i.e., points with rare descriptors. In many cases, however, this assumption does not hold. A cylinder, for example, will have too many points with similar descriptors. This results in many ambiguous correspondences between the model and the scene and the recognition method degenerates to a brute force search.

In our recognition approach, we combine a robust descriptor, a hashing technique and an efficient RANSAC variant. A similar strategy was proposed in [10]. In contrast to [10], where a hash table is used only for fast indexing into a large collection of geometry descriptors of *single* model points, we use a hash table to store descriptors of *pairs* of oriented model points (called doublets). This not only enables us to efficiently determine the model doublets which are similar to a sampled scene doublet but also allows for a very easy computation of the aligning rigid transform since it is uniquely defined by two corresponding doublets. Furthermore, in [10], a complex scene preprocessing is performed before running the actual object recognition: (i) multiple views of the scene are registered in order to build a more complete scene description and (ii) a scene segmentation is executed to separate the object from the background. In contrast to this, our method copes with a single view of the scene and does not require any segmentation. Moreover, the scenes used in all tests presented in [10] contain a single object and some background clutter. In this paper, we deal with the more challenging problem of object recognition and pose estimation in scenes which contain multiple object instances plus background clutter.

Before we describe our algorithm in detail, we briefly review the surface registration technique presented in [11] and include a short discussion on RANSAC [12] since both are of special relevance to our work.

**Fast Surface Registration** [11] To put it briefly, the task of rigid surface registration is to find a rigid transform which aligns two given surfaces. Let $\mathbf{S}$ be a surface given as a set of oriented points. For a pair of oriented points $(\mathbf{u}, \mathbf{v}) = ((\mathbf{p}_u, \mathbf{n}_u), (\mathbf{p}_v, \mathbf{n}_v)) \in \mathbf{S} \times \mathbf{S}$, a descriptor $f : \mathbf{S} \times \mathbf{S} \to \mathbb{R}^4$ is defined by

$$f(\mathbf{u}, \mathbf{v}) = \begin{pmatrix} f_1(\mathbf{u}, \mathbf{v}) \\ f_2(\mathbf{u}, \mathbf{v}) \\ f_3(\mathbf{u}, \mathbf{v}) \\ f_4(\mathbf{u}, \mathbf{v}) \end{pmatrix} = \begin{pmatrix} \|\mathbf{p}_u - \mathbf{p}_v\| \\ \angle(\mathbf{n}_u,\, \mathbf{n}_v) \\ \angle(\mathbf{n}_u,\, \mathbf{p}_v - \mathbf{p}_u) \\ \angle(\mathbf{n}_v,\, \mathbf{p}_u - \mathbf{p}_v) \end{pmatrix}, \tag{1}$$

where $\angle(\mathbf{a}, \mathbf{b})$ denotes the angle between $\mathbf{a}$ and $\mathbf{b}$. In order to register two surfaces $\mathbf{S}_1$ and $\mathbf{S}_2$, oriented point pairs $(\mathbf{u}, \mathbf{v}) \in \mathbf{S}_1 \times \mathbf{S}_1$ and $(\tilde{\mathbf{u}}, \tilde{\mathbf{v}}) \in \mathbf{S}_2 \times \mathbf{S}_2$ are sampled uniformly and the corresponding descriptors $f(\mathbf{u}, \mathbf{v})$ and $f(\tilde{\mathbf{u}}, \tilde{\mathbf{v}})$ are computed and stored in a four-dimensional hash table. The hash table is continuously filled in this way until a collision occurs, i.e., until a descriptor of a pair from $\mathbf{S}_1 \times \mathbf{S}_1$ and a descriptor of a pair from $\mathbf{S}_2 \times \mathbf{S}_2$ end up in the same hash table cell. Computing the rigid transform which best aligns (in least square sense) the colliding pairs gives a transform hypothesis for the surfaces.

According to [11], this process is repeated until a hypothesis is good enough, a predefined time limit is reached or all combinations are tested. Non of these stopping criteria is well-grounded: the first two are ad hoc and the last one is computationally infeasible. In contrast to this, we compute the number of iterations required to recognize model instances with a user-defined success probability. Furthermore, a direct application of the above described registration technique to 3D object recognition will have an unfavorable computational complexity since it will require a sequential registration of each model to the scene.

**RANSAC** [12] can be seen as a general approach for model recognition. It works by uniformly drawing minimal point sets from the scene and computing a transform which aligns the model with the minimal point set.[1] The score of the resulting hypothesis is computed by counting the number of transformed model points which lie within a certain $\epsilon$-band of the scene. After a given number of trials, the model is considered to be recognized at the locations defined by the hypotheses which achieved a score higher than a predefined threshold. In order to recognize the model with a probability $P_S$ we need to perform

$$N = \frac{\ln(1 - P_S)}{\ln(1 - P_M)}, \tag{2}$$

trials, where $P_M$ is the probability of recognizing the model in a single iteration.

The RANSAC approach has the advantages of being conceptually simple, very general and robust against outliers. Unfortunately, its direct application to the 3D object recognition problem is computationally very expensive. In order to compute an aligning rigid transform, we need at least three pairs of corresponding model $\leftrightarrow$ scene points. Under the simplifying assumption that the model is

[1] A minimal point set is the smallest set of points required to uniquely determine a given type of transform.

completely contained in the scene, the probability of drawing three such pairs in a single trial is $P_M(n) = \frac{3!}{(n-2)(n-1)n}$, where $n$ is the number of scene points. Since $P_M(n)$ is a small number we can approximate the denominator in (2) by its Taylor series $\ln(1 - P_M(n)) = -P_M(n) + O(P_M(n)^2)$ and get for the number of trials as a function of the number of scene points:

$$N(n) \approx \frac{-\ln(1 - P_S)}{P_M(n)} = O(n^3). \tag{3}$$

Assuming $q$ models in the library the complexity of RANSAC is $O(qn^3)$.

There are many modifications of the classic RANSAC scheme. Some recently proposed methods like ASSC [13] and ASKC [14] significantly improve outlier robustness by using a different score function. However, these variants are not designed to enhance the performance of RANSAC. In [15], an efficient RANSAC-like registration algorithm was proposed. However, it is not advisable to directly apply the method to 3D object recognition since it will require a sequential matching of each model to the scene. In [16], another efficient RANSAC variant for primitive shape detection was introduced. The method is related to ours since the authors also used a localized minimal point set sampling. Their method, however, is limited to the detection of planes, spheres, cylinders, cones and tori.

## 3 Method Description

Like most object recognition methods, ours consists of two phases. The first phase — the model preprocessing — is done offline. It is executed only once for each model and does not depend on the scenes in which the model instances have to be recognized. The second phase is the online recognition which is executed on the scene using the model representation computed in the offline phase.

### 3.1 Model Preprocessing Phase

For a given object model $\mathbf{M}$, we sample all pairs of oriented points $(\mathbf{u}, \mathbf{v}) = ((\mathbf{p}_u, \mathbf{n}_u), (\mathbf{p}_v, \mathbf{n}_v)) \in \mathbf{M} \times \mathbf{M}$ for which $\mathbf{p}_u$ and $\mathbf{p}_v$ are approximately at a distance $d$ from each other. For each pair, the descriptor $f(\mathbf{u}, \mathbf{v}) = (f_2(\mathbf{u}, \mathbf{v}), f_3(\mathbf{u}, \mathbf{v}), f_4(\mathbf{u}, \mathbf{v}))$ is computed as defined in (1) and stored in a three-dimensional hash table. Note that since $d$ is fixed we do not use $f_1$ as part of the descriptor. Furthermore, in contrast to [11], we do *not* consider all pairs of oriented points, but only those which fulfill $\|\mathbf{p}_u - \mathbf{p}_v\| \in [d - \delta_d, d + \delta_d]$, for a given tolerance value $\delta_d$. This has several advantages. The space complexity is reduced from $O(m^2)$ to $O(m)$, where $m$ is the number of points in $\mathbf{M}$ (this is an empirical measurement further discussed in [17]). For large $d$, the pairs we consider are wide-pairs which allow a much more stable computation of the aligning rigid transform than narrow-pairs do [17]. A further advantage of wide-pairs is due to the fact that the larger the distance the less pairs we have. Thus, computing and storing descriptors of wide-pairs leads to less populated hash table cells

which means that we will have to test less transform hypotheses in the online recognition phase and will save computation time.

Note, however, that the pair width $d$ can not be arbitrary large due to occlusions in real world scenes. For a typical value for $d$, there are still a lot of pairs with similar descriptors, i.e., there are hash table cells with too many entries. To avoid this overpopulation, we remove as many of the most populated cells as needed to keep only a fraction $K$ of the pairs in the hash table (in our implementation $K = 0.1$). This strategy leads to some information loss about the object shape. We take this into account in the online phase of our algorithm.

The final representation of all models $\mathbf{M}_1, \ldots, \mathbf{M}_q$ is computed by processing each $\mathbf{M}_i$ in the way described above using *the same* hash table. In order not to confuse the correspondence between pairs and models, each cell contains a list for each model which has pairs stored in the cell. In this way, new models can be added to the hash table without recomputing it.

### 3.2 Online Recognition Phase

The online recognition phase can be outlined as follows:

1. Initialization
   (a) Compute an octree for the scene $\mathbf{S}$ to produce a modified scene $\mathbf{S}^*$.
   (b) $\mathcal{T} \leftarrow \emptyset$ (an empty solution list).
2. Compute a number of iterations $N$ needed to achieve a probability for successful recognition higher than a predefined value $P_S$.

**[repeat $N$ times]**

3. Sampling
   (a) Sample a point $\mathbf{p}_u$ uniformly from $\mathbf{S}^*$.
   (b) Sample $\mathbf{p}_v \in \mathbf{S}^*$ uniformly from all points at a distance $d \pm \delta_d$ from $\mathbf{p}_u$.
4. Estimate normals $\mathbf{n}_u$ and $\mathbf{n}_v$ at $\mathbf{p}_u$ and $\mathbf{p}_v$, respectively, to get an oriented scene point pair $(\mathbf{u}, \mathbf{v}) = ((\mathbf{p}_u, \mathbf{n}_u), (\mathbf{p}_v, \mathbf{n}_v))$.
5. Compute the descriptor $f_{\mathbf{uv}} = (f_2(\mathbf{u}, \mathbf{v}), f_3(\mathbf{u}, \mathbf{v}), f_4(\mathbf{u}, \mathbf{v}))$ (see (1)).
6. Use $f_{\mathbf{uv}}$ as a key to the model hash table to retrieve the oriented model point pairs $(\mathbf{u}_j, \mathbf{v}_j)$ similar to $(\mathbf{u}, \mathbf{v})$.
   **[repeat for each $(\mathbf{u}_j, \mathbf{v}_j)$]**
   (a) Get the model $\mathbf{M}$ of $(\mathbf{u}_j, \mathbf{v}_j)$.
   (b) Compute the rigid transform $T$ that best aligns $(\mathbf{u}_j, \mathbf{v}_j)$ to $(\mathbf{u}, \mathbf{v})$.
   (c) Set $\mathcal{T} \leftarrow \mathcal{T} \cup (\mathbf{M}, T)$ if $(\mathbf{M}, T)$ is accepted by an acceptance function $\mu$.
   **[end repeat]**

**[end repeat]**

7. Filter conflicting hypotheses from $\mathcal{T}$.

For our algorithm to be fast, we need to search efficiently for closest points (in step 4) and for points lying on a sphere around a given point (in step 3b). These operations are greatly facilitated if a neighborhood structure is available

for the point set. Note that the 2D range image grid defines such a structure which, however, is not well suited for the above mentioned geometric operations. This is due to the fact that points which are neighbors on the gird are not necessarily close to each other in $\mathbb{R}^3$ because of perspective effects and scene depth discontinuities. A very efficient way to establish spatial proximity between points in $\mathbb{R}^3$ is to use an octree.

**Step 1, Initialization** In step 1a of the algorithm, we construct an octree with a fixed leaf size $L$ (the edge length of a leaf). The full octree leaves (the ones which contain at least one point) can be seen as voxels ordered in a regular axis-aligned 3D grid. Thus, each full leaf has unique integer coordinates $(i, j, k)$. We say that two full leaves are neighbors if the absolute difference between their corresponding integer coordinates is $\leq 1$. Next, we down-sample $\mathbf{S}$ by setting the new scene points in $\mathbf{S}^*$ to be the centers of mass of the full leaves. The center of mass of a full leaf is defined to be the average of the points it contains. In this way, a one-to-one correspondence between the points in $\mathbf{S}^*$ and the full octree leaves is established. Two points in $\mathbf{S}^*$ are neighbors if the corresponding full leaves are neighbors.

**Step 2, Number of Iterations** This step is explained in Section 3.3.

**Step 3, Sampling** In the sampling stage, we make extensive use of the scene octree. As in the classic RANSAC, we sample minimal sets from the scene. In our case, a minimal set consists of two oriented points. However, in contrast to RANSAC, they are *not* sampled uniformly. Only the first point, $\mathbf{p}_u$, is drawn uniformly from $\mathbf{S}^*$. In order to draw the second point, $\mathbf{p}_v$, we first retrieve the set $\mathbf{L}$ of all full leaves which are intersected by the sphere with center $\mathbf{p}_u$ and radius $d$, where $d$ is the pair width used in the offline phase (see Section 3.1). This operation can be implemented very efficiently due to the hierarchical structure of the octree. Finally, a leaf is drawn uniformly from $\mathbf{L}$ and $\mathbf{p}_v$ is set to be its center of mass.

**Step 4, Normal Estimation** The normals $\mathbf{n}_u$ and $\mathbf{n}_v$ are estimated by performing a Principal Component Analysis. $\mathbf{n}_u$ and $\mathbf{n}_v$ are set to be the eigenvectors corresponding to the smallest eigenvalues of the covariance matrix of the points in the neighborhood of $\mathbf{p}_u$ and $\mathbf{p}_v$, respectively. The result is the oriented scene point pair $(\mathbf{u}, \mathbf{v}) = ((\mathbf{p}_u, \mathbf{n}_u), (\mathbf{p}_v, \mathbf{n}_v))$.

**Steps 5 and 6, Hypotheses Generation and Testing** Step 5 involves the computation of the descriptor $f_{\mathbf{uv}} = (f_2(\mathbf{u}, \mathbf{v}), f_3(\mathbf{u}, \mathbf{v}), f_4(\mathbf{u}, \mathbf{v}))$ (see (1)). In step 6, $f_{\mathbf{uv}}$ is used as a key to the model hash table to retrieve all model pairs $(\mathbf{u}_j, \mathbf{v}_j)$ which are similar to $(\mathbf{u}, \mathbf{v})$. For each $(\mathbf{u}_j, \mathbf{v}_j)$, the model $\mathbf{M}$ corresponding to $(\mathbf{u}_j, \mathbf{v}_j)$ is retrieved (step 6a) and the rigid transform $T$ which best aligns (in least squares sense) $(\mathbf{u}_j, \mathbf{v}_j)$ to $(\mathbf{u}, \mathbf{v})$ is computed (step 6b). The result of these two sub-steps is the hypothesis that the model $\mathbf{M}$ is in the scene at the location defined by $T$. In order to save the hypothesis in the solution list, it has to be accepted by the acceptance function $\mu$.

**Step 6c, The Acceptance Function** $\mu$ measures the quality of a hypothesis $(\mathbf{M}, T)$ and consists of a support term and a penalty term. As in RANSAC, the support term, $\mu_S$, is proportional to the number $m_S$ of transformed model

points (i.e., points from $T(\mathbf{M})$) which fall within a certain $\epsilon$-band of the scene. More precisely, $\mu_S(\mathbf{M}, T) = m_S/m$, where $m$ is the number of model points. To compute $m_S$, we back project $T(\mathbf{M})$ in the scene range image and count the number of points which have a z-coordinate in the interval $[z - \epsilon, z + \epsilon]$, where $z$ is the z-coordinate of the corresponding range image pixel.

In contrast to RANSAC, our algorithm contains a penalty term, $\mu_P$, which is proportional to the size of the transformed model parts which occlude the scene. It is clear that in a scene viewed by a camera a correctly recognized (non-transparent) object can not occlude scene points reconstructed from the same viewpoint. We penalize hypotheses which violate this condition. We compute the penalty term by counting the number $m_P$ of transformed model points which are between the projection center of the range image and a valid range image pixel and thus are "occluding" reconstructed scene points. We set $\mu_P(\mathbf{M}, T) = m_P/m$, where $m$ is the number of model points.

For $(\mathbf{M}, T)$ to be accepted as a valid hypothesis it has to have a support higher than a predefined $S \in [0, 1]$ and a penalty lower than a predefined $P \in [0, 1]$.

**Step 7, Filtering Conflicting Hypotheses** We say that an accepted hypothesis $(\mathbf{M}, T)$ explains a set $\mathbf{P} \subset \mathbf{S}^*$ of scene points if for each $\mathbf{p} \in \mathbf{P}$ there is a point from $T(\mathbf{M})$ which lies within the octree leaf corresponding to $\mathbf{p}$. Note that the points from $\mathbf{P}$ explained by $(\mathbf{M}, T)$ are *not* removed from $\mathbf{S}^*$ because there could be a better hypothesis, i.e., one which explains a superset of $\mathbf{P}$. Two hypotheses are conflicting if the intersection of the point sets they explain is non-empty. At the end of step 6, many conflicting hypotheses are saved in the list $\mathcal{T}$. To filter the weak ones, we construct a so called conflict graph. Its nodes are the hypotheses in $\mathcal{T}$ and an edge is connecting two nodes if the hypotheses they represent are conflicting ones. To produce the final output, the solution list is filtered by performing a non-maximum suppression on the conflict graph: a node is removed if it has a better neighboring node.

### 3.3 Time Complexity

The complexity of the proposed algorithm is dominated by three major factors: (i) the number of iterations (the loop after step 2), (ii) the number of pairs per hash table cell (the loop in step 6) and (iii) the cost of evaluating the acceptance function for each object hypothesis (step 6c). In the following, we discuss each one in detail.

**(i)** Consider the scene $\mathbf{S}^*$ consisting of $|\mathbf{S}^*| = n$ points and a model instance $\mathbf{M}$ therein consisting of $|\mathbf{M}| = m$ points. We already saw in the discussion on RANSAC at the end of Section 2 that we need

$$N = \frac{\ln(1 - P_S)}{\ln(1 - P_M)} \tag{4}$$

iterations to recognize $\mathbf{M}$ with a predefined success probability $P_S$, where $P_M$ is the probability of recognizing $\mathbf{M}$ in a single iteration. Recall from Section 2 that in the classic RANSAC applied to 3D object recognition we have $P_M \approx 1/n^3$.

Our sampling strategy and the use of the model hash table lead to a significant increase of $P_M$ and thus to a reduction of the complexity. In the following, we estimate $P_M$.

Let $P(\mathbf{p}_u \in \mathbf{M}, \mathbf{p}_v \in \mathbf{M})$ denote the probability that both points are sampled from $\mathbf{M}$ (see step 3 in Section 3.2). Thus, the probability of recognizing $\mathbf{M}$ in a single iteration is

$$P_M = KP(\mathbf{p}_u \in \mathbf{M}, \mathbf{p}_v \in \mathbf{M}), \tag{5}$$

where $K$ is the fraction of oriented point pairs for which the descriptors are saved in the model hash table (see Section 3.1). Using conditional probability and the fact that $P(\mathbf{p}_u \in \mathbf{M}) = m/n$ we can rewrite (5) to get

$$P_M = (m/n)KP(\mathbf{p}_v \in \mathbf{M} | \mathbf{p}_u \in \mathbf{M}). \tag{6}$$

$P(\mathbf{p}_v \in \mathbf{M} | \mathbf{p}_u \in \mathbf{M})$ is the probability to sample $\mathbf{p}_v$ from $\mathbf{M}$ given that $\mathbf{p}_u \in \mathbf{M}$. Recall from Section 3.2 that $\mathbf{p}_v$ is not independent of $\mathbf{p}_u$ because it is sampled uniformly from the set $\mathbf{L}$ consisting of the scene points which lie on the sphere with center $\mathbf{p}_u$ and radius $d$, where $d$ is the pair width used in the offline phase. Under the assumptions that the visible object part has an extent larger than $2d$ and that the reconstruction is not too sparse, $\mathbf{L}$ contains points from $\mathbf{M}$. Thus, $P(\mathbf{p}_v \in \mathbf{M} | \mathbf{p}_u \in \mathbf{M}) = |\mathbf{L} \cap \mathbf{M}|/|\mathbf{L}|$ is well-defined and greater than zero. $|\mathbf{L} \cap \mathbf{M}|/|\mathbf{L}|$ depends on the scene, i.e., it depends on the extent and the shape of the visible object part. Estimating $C = |\mathbf{L} \cap \mathbf{M}|/|\mathbf{L}|$ by, e.g., $1/4$ (this is what we use in our implementation) accounts for up to 75% outliers and scene clutter. Thus, we get for $P_M$ as a function of $n$ (the number of scene points)

$$P_M(n) = (m/n)KC. \tag{7}$$

Again, approximating the denominator in (4) by its Taylor series $\ln(1-P_M(n)) = -P_M(n) + O(P_M(n)^2)$ we get for the number of iterations

$$N(n) \approx \frac{-\ln(1-P_S)}{P_M(n)} = \frac{-n\ln(1-P_S)}{mKC} = O(n). \tag{8}$$

This proves that the number of iterations depends linearly on the number of scene points. Furthermore, it is guaranteed that the model instances will be recognized with the desired probability $P_S$.

**(ii)** The number of pairs per hash table cell depends on the number of models as well on the number of points of each model. An algorithm is considered to scale well with the number of models if its runtime is less than the sum of the runtime needed for the recognition of each model separately [10, 18]. In other words, an algorithm should need less time than it is needed for a sequential matching of each model to the scene. The use of the model hash table ensures this in the case of our method. For almost all real world objects it holds that a hash table cell does not store pairs from all models. Furthermore, not all pairs originating from a model end up in the same hash table cell.

**(iii)** The acceptance function $\mu$ runs in $O(l)$ time, where $l$ is the number of model points. Note that $\mu$ does not depend on the number of scene points since back projecting a model point in the range image is performed in constant time.

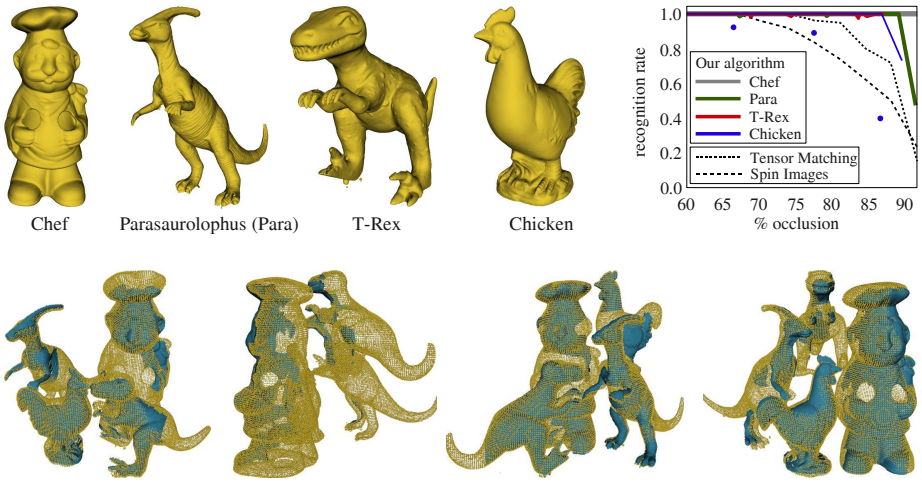



**Fig. 2.** (Upper left) The models used in the comparison test case. (Upper right) The continuous lines indicate the recognition rate of our algorithm for each object as a function of its occlusion. The dashed lines give the recognition rate of the spin images and the tensor matching approaches on the same scenes as reported in [3]. Note that our method outperforms both algorithms. The chef is recognized in all trials, even in the case of occlusion over 91%. The blue dots represent the recognition rate in the three chicken test scenes in which our method performs worse than the other algorithms. This is due to the fact that in these scenes only the chicken's back part is visible which contains strongly varying normals which makes it difficult to compute a stable aligning transform. (Lower row) Four (out of 50) test scenes and the corresponding recognition results. The recognized models are rendered as yellow point clouds and superimposed over the scenes which are rendered as blue meshes. These are challenging examples since only small parts of the objects are visible.

## 4 Experimental Results

**Comparison with spin images [4] and tensor matching [3]** In the first test scenario, we compare the recognition rate of our algorithm with the spin images [4] and the tensor matching [3] approaches on occluded real scenes. We test our method on the same 50 data sets which are used in [3]. This allows for a precise comparison without the need of re-implementing neither of the two algorithms. The models of the four toys to be recognized are shown in the upper row of Fig. 2. Each test scene contains the toys (not necessary all four of them) in different positions and orientations. Each scene is digitized with a laser range finder from a single viewpoint which means that the back parts of the objects are not visible. Furthermore, in most scenes the toys are placed such that some of them occlude others which makes the visible object parts even smaller. The lower row of Fig. 2 shows exemplary four (out of 50) test scenes with the corresponding recognition results obtained with our algorithm. Since our algorithm is a probabilistic one we run 100 recognition trials on each scene

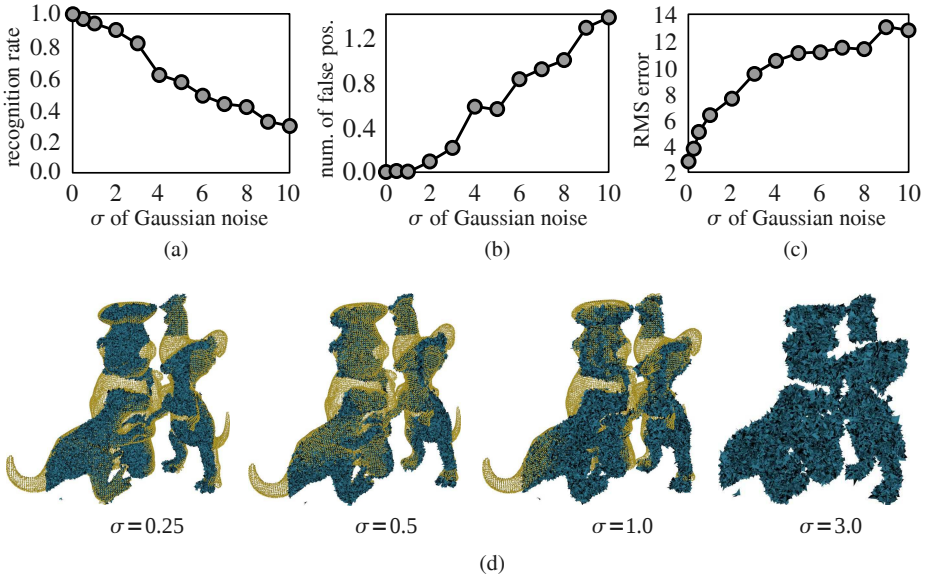



**Fig. 3.** (a) - (c) Recognition rate, mean number of false positives and mean RMS error as functions of the $\sigma$ of Gaussian noise. One $\sigma$ unit equals 1% of the bounding box diagonal length of the scene. The RMS units are in millimeters. (d) Typical recognition results for noise degraded data sets.

and compute the recognition rate for each object represented in the scene in the following way. We visually inspect the result of each of the 100 trials. If object **A** was recognized $n$ times ($0 \leq n \leq 100$) then the recognition rate for **A** is $n/100$. Since the occlusion of every object in each scene is known we report the recognition rate for each object as a function of its occlusion. According to [4], the occlusion for an object model is given by $1 - \frac{\text{area of visible model surface}}{\text{total area of model surface}}$. The results of the tests and the comparison with the spin images [4] and the tensor matching [3] approaches are summarized in the upper right part of Fig. 2.

**Noisy and Sparse Scenes** In the second scenario, we run tests under varying noisy conditions. The models to be recognized are the same as in the last test case and the scene is the third one in the lower row of Fig. 2. Next, several versions of the scene are computed by degrading it by zero-mean Gaussian noise with different variance values $\sigma$. Again, we perform 100 recognition trials for each noisy scene and compute the recognition rate, the mean number of false positives and the mean RMS error as functions of $\sigma$. For a point set **P**, a (rigidly) transformed copy **Q** and a (rigid) transform $T$ the RMS error measures how close each point $\mathbf{p}_i \in \mathbf{P}$ comes to its corresponding point $\mathbf{q}_i \in \mathbf{Q}$ after transforming **Q** by $T$. Thus RMS measures the quality of $T$. It is given by

$$\text{RMS}(T) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|\mathbf{p}_i - T(\mathbf{q}_i)\|^2}, \tag{9}$$

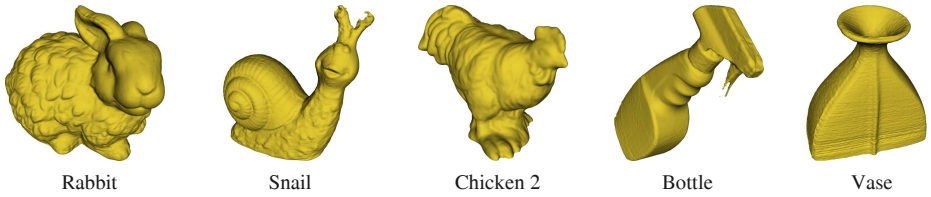



**Fig. 4.** The models used for object recognition in scenes reconstructed with a low-cost light intersection based device.

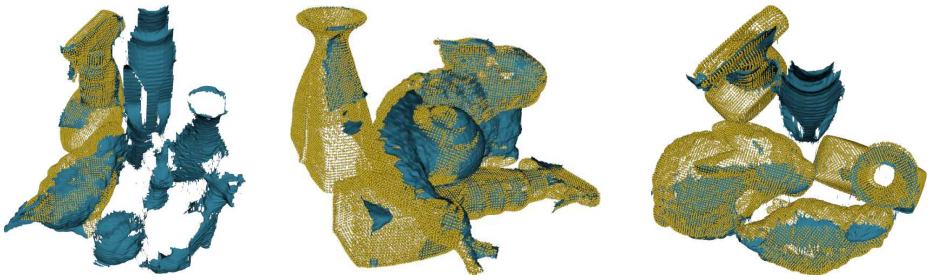


**Fig. 5.** Typical recognition results obtained with our method for three test scenes. The scenes are shown as blue meshes and the recognized model instances are rendered as yellow point clouds and superimposed over the meshes. Some of the scenes contain unknown objects (the left and the right one). Note that the scene reconstruction contains only small portions of the objects.

where $N$ is the number of points in $\mathbf{P}$. Since we know the ground truth location of each model in the test scene the RMS error of the rigid transform computed by our method can be easily calculated.[2] The results of all noise tests are summarized in Fig. 3(a) – (c). Typical recognition results and four of the noisy scenes are shown in Fig. 3(d).

Next, we demonstrate the ability of our method to deal with data sets corrupted by noise which is not artificially generated but originates in scan device imprecision. Note that the scenes used in [4] and [3] are dense and have a relatively good quality. We use a low-cost light section based scanner which gives sparser and noisier data sets. The models used in this test scenario are shown in Fig. 4. Typical recognition results of our method are shown in Fig. 1 and Fig. 5.

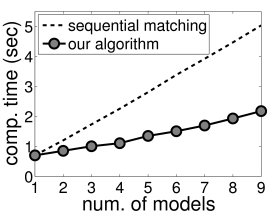
**Runtime** In the last test scenario, we experimentally verify the two main claims regarding the time complexity of our algorithm, namely that it needs less time than it is required for a sequential matching of each model to the scene and that it has a linear complexity in the number of scene points.

First, we measure the runtime dependency on the number of models. The models used in this test case are the ones shown in Fig. 2 and Fig. 4 and the

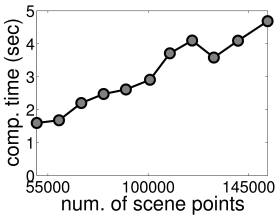
[2] The ground truth rigid transform for the models for each scene is available on the webpage of the authors of [3].

| model | comp. time (sec) |
|---|---|
| Chef | 0.568 |
| Para | 0.533 |
| T-Rex | 0.5 |
| Chicken | 0.522 |
| Rabbit | 0.536 |
| Snail | 0.546 |
| Chicken 2 | 0.551 |
| Bottle | 0.577 |
| Vase | 0.566 |

(a)




(b)




(c)

**Fig. 6.** (a) Recognition time for each model. (b) Computation time for a simultaneous recognition of multiple objects (solid line) compared to a sequential matching of each model to the scene (dashed line). The runtime in the case of the sequential matching is the sum of the times reported in (a) for each model. (c) Linear time complexity in the number of scene points for the simultaneous recognition of 9 models.

scene is the leftmost one in Fig. 5. The recognition time for each object (when it is the only one loaded in the hash table) is reported in Fig. 6(a). In Fig. 6(b), the computation time of our algorithm as a function of the number of models loaded in the hash table is compared with the time needed for a sequential matching of each model to the scene. The difference in the performance is obvious.

Second, we measure how the runtime depends on the number of scene points. There are eleven different data sets involved in this test case — a subset from the scenes used in the comparison test case. It is important to note that we do *not* take a single data set and down/up-sample it to get the desired number of points. Instead we choose eleven *different* scenes with varying scene extent, number of points and number of objects. This suggests that the results will hold for arbitrary scenes. We report the results of this test in Fig. 6(c).

The algorithm presented in this paper is implemented in C++ and all tests were performed on a laptop with an Intel Core 2 Duo 3GHz CPU and 4GB RAM.

## 5 Conclusions

In this paper, we introduced a new algorithm for multiple 3D object recognition in noisy, sparsely reconstructed and unsegmented range data. The method combines a robust descriptor, a hashing technique and an efficient RANSAC-like sampling strategy. We provided a complexity analysis of the algorithm and derived the number of iterations required to recognize the model instances with a given probability. In the experimental part of the paper, it was verified that the proposed algorithm scales well with the number of models and that it has a linear time complexity in the number of scene points. Furthermore, we showed that our method performs well on noisy, sparse and unsegmented scenes in which only small parts of the objects are visible. A comparison showed that our method outperforms the spin images [4] and the tensor matching [3] approaches in terms of recognition rate.

## References


1. Lamdan, Y., Wolfson, H.: Geometric Hashing: A General And Efficient Model-based Recognition Scheme. In: ICCV. (1988) 238–249
2. Ballard, D.H.: Generalizing the Hough Transform to Detect Arbitrary Shapes. Pattern Recognition **13** (1981) 111–122
3. Mian, A.S., Bennamoun, M., Owens, R.A.: Three-Dimensional Model-Based Object Recognition and Segmentation in Cluttered Scenes. IEEE TPAMI **28** (2006) 1584–1601
4. Johnson, A., Hebert, M.: Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes. IEEE TPAMI **21** (1999) 433–449
5. Hetzel, G., Leibe, B., Levi, P., Schiele, B.: 3D Object Recognition from Range Images Using Local Feature Histograms. In: CVPR. (2001) 394–399
6. Frome, A., Huber, D., Kolluri, R., Bülow, T., Malik, J.: Recognizing Objects in Range Data Using Regional Point Descriptors. In: ECCV. (2004) 224–237
7. Gelfand, N., Mitra, N., Guibas, L., Pottmann, H.: Robust Global Registration. In: Eurographics Symposium on Geometry Processing. (2005) 197–206
8. Zaharescu, A., Boyer, E., Varanasi, K., Horaud, R.: Surface Feature Detection and Description with Applications to Mesh Matching. In: CVPR. (2009) 373–380
9. Sun, J., Ovsjanikov, M., Guibas, L.J.: A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion. Comput. Graph. Forum **28** (2009) 1383–1392
10. Matei, B., Shan, Y., Sawhney, H.S., Tan, Y., Kumar, R., Huber, D.F., Hebert, M.: Rapid Object Indexing Using Locality Sensitive Hashing and Joint 3D-Signature Space Estimation. IEEE TPAMI **28** (2006) 1111–1126
11. Winkelbach, S., Molkenstruck, S., Wahl, F.M.: Low-Cost Laser Range Scanner and Fast Surface Registration Approach. In: Pattern Recognition, 28th DAGM Symposium, Proceedings. (2006) 718–728
12. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24** (1981) 381–395
13. Wang, H., Suter, D.: Robust Adaptive-Scale Parametric Model Estimation for Computer Vision. IEEE TPAMI **26** (2004) 1459–1474
14. Wang, H., Mirota, D., Hager, G.D.: A Generalized Kernel Consensus-Based Robust Estimator. IEEE TPAMI **32** (2010) 178–184
15. Chen, C.S., Hung, Y.P., Cheng, J.B.: RANSAC-Based DARCES: A New Approach to Fast Automatic Registration of Partially Overlapping Range Images. IEEE TPAMI **21** (1999) 1229–1234
16. Schnabel, R., Wahl, R., Klein, R.: Efficient RANSAC for Point-Cloud Shape Detection. Comput. Graph. Forum **26** (2007) 214–226
17. Aiger, D., Mitra, N.J., Cohen-Or, D.: 4-points Congruent Sets for Robust Pairwise Surface Registration. ACM Trans. Graph. **27** (2008)
18. Shan, Y., Matei, B., Sawhney, H.S., Kumar, R., Huber, D.F., Hebert, M.: Linear Model Hashing and Batch RANSAC for Rapid and Accurate Object Recognition. In: CVPR. (2004) 121–128

# Towards On-Line Intensity-Based Surface Recovery from Monocular Images

Oliver Ruepp[1]
ruepp@in.tum.de
Darius Burschka[1]
burschka@cs.tum.edu
Robert Bauernschmitt[2]
bauernschmitt@dhm.mhn.de

[1] Institut für Informatik VI
Technische Universität München
Boltzmannstraße 3
85748 Garching, Germany

[2] Deutsches Herzzentrum München
Lazarettstr. 36
80636 München, Germany

**Abstract**

We present a novel method for vision-based recovery of three-dimensional structures through simultaneous model reconstruction and camera position tracking from monocular images. Our approach does not rely on robust feature detecting schemes (such as SIFT, Good Features to Track etc.), but works directly on intensity values in the captured images. Thus, it is well-suited for reconstruction of surfaces that exhibit only little texture due to partial homogenity of the surfaces.

## 1 Introduction

Tracking and reconstruction of surfaces from video data is a problem that has been subject of extensive research work, and a number of methods exist for this problem. Many of the established methods, however, rely on presence of salient image features, such as SIFT [11] features, Good Features to Track [22], FAST corner detection [20] and so on. In some settings, however, the objects one is dealing with do not exhibit much structure, which makes it very hard to find robust, dense feature sets using traditional methods. In such situations, it pays off to use intensity-based methods, which is what we have investigated.

Originally, our idea was to generalize an approach developed by Ramey et al. [19] for efficient tracking of the disparity map in stereo video streams. Their method is quite general in that it can be used in conjunction with arbitrary parametric models of disparity maps, and it is especially efficient if the model is linear in parameters. In their test setups, they have used a B-Spline surface to represent the disparity map. We wanted to generalize their approach in the sense that the cameras do not need to be mounted on a stereo rig, but instead they are allowed to move independently from each other.

As an intermediate step towards this goal, we developed the method presented in this paper, which allows simultaneous model reconstruction and camera localization from monocular images in static scenes. In comparison to the two-camera scenario described above, this is equivalent to a situation where two cameras are present, but only one of them is moving, and the observed scene is static.

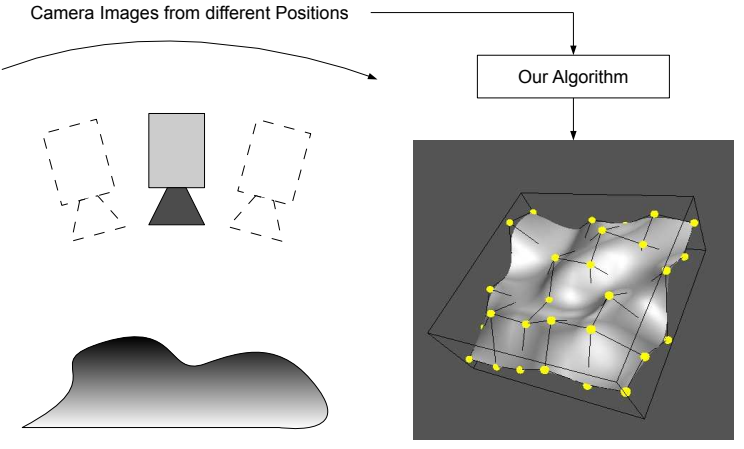



Figure 1: Schematic overview of the main idea of our algorithm.

Some examples of real-time dense reconstruction methods are described in papers by Palaanen et al. [16], Pan et al. [17], and in the recent work by Newcombe et al. [14]. All of these methods have in common, however, that they rely on some kind of feature detecting scheme, which is what we want to avoid here.

A number of offline methods for model-based bundle-adjustment have been described with applications to face modeling [5, 21]. Our method is different in that it tries to build the model during run-time, starting out with a very crude initial model (a plane) and refining the model in each step.

A part of the problem of surface reconstruction from image intensities is the surface modeling and reconstruction methodology itself. A thorough treatment of that problem has been done recently [1, 3], and results have been established using feature-based methods.

Ramey's tracking method [19] that inspired our development basically employs the Gauss-Newton minimization algorithm for tracking. The generalization that we have performed leads to an optimization problem that corresponds to intensity-based bundle-adjustment that is restricted to two frames. An in-depth survey of the original bundle-adjustment method is given in the book by Hartley and Zisserman [9]. The paper by Triggs et al. [23] provides a good overview of bundle adjustment variants and related methods. There is also a more recent paper evaluating the status of real-time bundle adjustment methods [4].

We are interested in recovering the surface of a 3D object on-line from a stream of monocular camera images. The object we are interested in must be static. Furthermore, since we are also tracking the object of interest, it is required that during the video sequence, sight of the object is not lost. Occlusions or self-occlusions are, until now, not accounted for. However, such problems have already been examined by other researchers, e.g. [6, 18], and we expect it to be possible to incorporate similar techniques into our solution.

The basic concept of the algorithm is visualized in Figure 1. It can be summarized as follows: In traditional bundle adjustment, coordinates of 3D points that are associated with feature points are recovered from a set of 2D feature position measurements. This approach will obviously work only if a feature detecting scheme can be used at all. It has the advantage that the images can be taken from very different camera positions. In our case, we do not assume that robust feature extraction is possible, and thus we do not work with 2D positions,

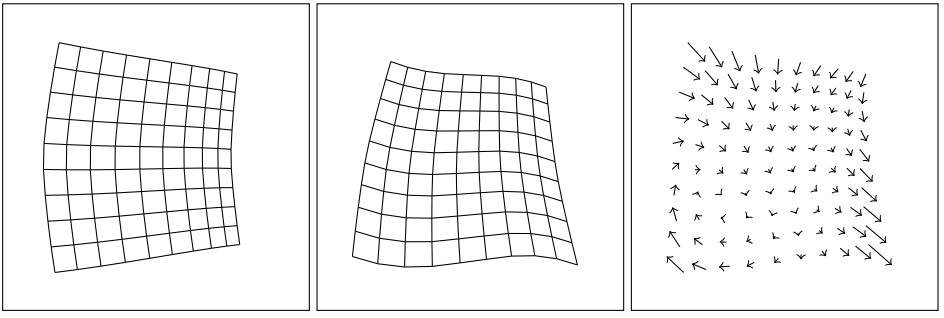


Figure 2: Left, middle: Surface under two different camera positions. Right: Warping of surface coordinates from left to right image.

but with image intensities. This is only feasible if the camera positions of subsequent images are not too far away from each other.

In the next section, we give a detailed explanation of the method we have developed. Results have been obtained from real world data sets as well as synthetic data sets, and are presented in Section 3.

# 2 Method

## 2.1 Overview

There are many possibilities for representing a model of a scene, with the most straightforward one being a point cloud. This is a very general representation that is actually used in the traditional bundle adjustment algorithm, where it works well under the assumption that points can be reliably identified. Unfortunately, this assumption can not be used for intensity-based methods, since identifying a point based on its intensity is obviously bound to fail. This disqualifies the point cloud model for our purposes.

The usual approach taken to address this problem is the introduction of additional constraints in form of a parametric surface model of type $S : \mathbb{R}^k \times \mathbb{R}^2 \to \mathbb{R}^3$, on which the points lie. Mathematically speaking, $S$ maps a set of $k$ parameters together with surface coordinates $u, v$ to three-dimensional spatial coordinates. Such a model is especially suitable for representation of scenarios that can be described with a small parameter set. Compared to the point cloud representation, it constitutes a loss of generality, but this is a compromise that seems to be necessary to make.

Inspired by the method of Ramey et al. [19], we do not directly model the scene as a 3D surface. Instead, we choose the model to be a depth map of some object of interest for some reference image of the video stream. A 3D surface model can easily be retrieved from that representation, as will be shown later.

Observing a static, three-dimensional smooth surface $S$ under two different camera positions will essentially yield two images that are related to each other via a "warping" function. If, for two snapshots of a scene, we exactly know the corresponding extrinsic camera parameters and we have a perfect mathematical description of the surface that we are observing, we can, for each surface pixel in one image, determine the position of that pixel in the other image. In other words, we can formulate a function of type $\mathbb{R}^2 \to \mathbb{R}^2$ that transforms pixel

coordinates from one image to another, and we would expect the corresponding image values to be equal. Figure 2 shows an example for the warping function.

Because we require the surface to be completely visible at all times, it would not make sense to try to establish a depth map for the whole image, points could very easily be lost immediately after initialization as a result of minor camera movements. Instead, if we focus on only a certain region of interest within the image, it is easier for the user to assure that that region is always visible. Thus, before starting the actual reconstruction process, we have the user choose such an area within a reference image.

## 2.2 Mathematical Model

We do not take into account all pixels in the region of interest because the optimization process is quite costly. Instead, we only focus on a number of reference pixels that are selected according to a weak criterium that will be described later. These pixels are picked from a user-defined region of interest in a reference image and tracked through the entire image sequence.

As we have mentioned earlier, we are modeling the depth map of the region of interest that has been chosen by the user. That depth map is then a function $S_{\mathbf{d}}(u,v)$ mapping a $k$-dimensional parameter vector $\mathbf{d}$ together with image coordinates $(u,v) \in \mathbb{R}^2$ to a depth value $\lambda \in \mathbb{R}$ at the specified coordinate. Given intrinsic camera parameters, this depth map can actually be interpreted as a 3D surface. Before we start to derive the image warping function, we want to give an overview of definitions and notations. In the following, images are numbered consecutively, and the numbering starts with $n = 0$. Then, let

- $\mathbf{d_n}$ denote the $k$-dimensional vector of parameters of the model describing the depth map.
- $S_{\mathbf{d}}(u,v)$ denote a function of type $\mathbb{R}^k \times \mathbb{R}^2 \to \mathbb{R}$ that maps model parameters together with image pixel coordinates to 1D pixel depth values.
- $\mathbf{p}_n = (\mathbf{t}_n, \mathbf{q}_n)$ denote the extrinsic camera parameters corresponding to image $n$, consisting of translation vector $\mathbf{t}_n \in \mathbb{R}^3$ and rotation quaternion $\mathbf{q}_n \in \mathbb{R}^4$.
- $T(\mathbf{t},\mathbf{q},\mathbf{p}) : \mathbb{R}^3 \times \mathbb{R}^4 \times \mathbb{R}^3 \to \mathbb{R}^3$ is a transformation mapping 3D spatial coordinates $\mathbf{p}$ to 3D coordinates in the camera frame described by a translation vector $\mathbf{t}$ and a rotation quaternion $\mathbf{q}$.
- $\pi(\mathbf{p})$ be the projection of a 3D point $\mathbf{p}$ to 2D image coordinates, according to the internal camera calibration parameters of the camera used.
- $I_n(x,y)$ be the image function of image $n$, containing all pixel values. $I_0$ is hence the reference image function.
- $(u_1,v_1),\ldots,(u_m,v_m)$ denote the pixel coordinates of the $m$ reference pixels, chosen from the ROI in the reference image.

For the monocular camera, we assume a pinhole model with projection function

$$\pi(\mathbf{p}) = \left( \frac{\mathbf{p}_1 f_x}{\mathbf{p}_3} + c_x, \frac{\mathbf{p}_2 f_y}{\mathbf{p}_3} + c_y \right)^T \tag{1}$$

where $f_x, f_y$ are focal lengths in terms of pixel dimensions, $c_x, c_y$ describe the location of the camera center, and $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)^T$ is a vector of Cartesian point coordinates. In case of significant radial distortions, the images can be rectified before usage.

If we associate the camera frame in image 0 with the reference frame, each pixel of the region of interest corresponds to a ray originating from the camera position (which coincides with the origin) that intersects the object surface at a certain depth. The pixel color then corresponds (ignoring possible specularities) to the color of the surface texture at that position. The ray corresponding to pixel coordinates $(u,v)$ can then be parameterized by depth $\lambda$, yielding a function $r_{u,v}(\lambda)$:

$$r_{u,v}(\lambda) = \lambda \cdot \left( \frac{u - c_x}{f_x}, \frac{v - c_y}{f_y}, 1 \right)^T. \tag{2}$$

It is obvious then that the composite function $r_{u,v}(S_{\mathbf{d}}(u,v))$ is a description of the three-dimensional model shape. If that model is observed from a different camera position $\mathbf{p}_n$, yielding a different image with index $n$, we need to rotate and translate the 3D coordinates produced by above function. This can be achieved by using the formula $T(\mathbf{p}_n, r_{u,v}(S_{\mathbf{d}}(u,v))$.

If we knew the perfect model parameters $\mathbf{d}$ and exact camera parameters $\mathbf{p}_n$ for image $n$, we would expect the relationship $I_n(\pi(T(\mathbf{p}_n, r_{u,v}(S_{\mathbf{d}}(u,v)))) = I_0(u,v)$ to hold for all model surface coordinates $(u,v)$.

Thus, we assume that the correct camera position and the correct model parameters together minimize some difference measure $c$ (*e.g.* least squares) on intensity values, which can be formulated as

$$c(I_n(\pi(T(\mathbf{p}_n, r_{u,v}(S_{\mathbf{d}}(u,v))))) - I_0(u,v)). \tag{3}$$

As has been mentioned before, the optimization process necessary for determining camera and model parameters is quite computationally intensive. Thus, we will not include all possible pixel $(u,v)$ coordinates in the optimization process, but only the coordinates of $m$ chosen reference points. The corresponding objective function $o(\mathbf{d}, \mathbf{p}_n)$ can then be defined as

$$o(\mathbf{d}, \mathbf{p}_n) = \sum_{i=1}^{m} (c(I_n(\pi(T(\mathbf{p}_t, r_{u_i,v_i}(S_{\mathbf{d}}(u_i,v_i))) - I_0(u_i,v_i)))^2 \tag{4}$$

Our problem of finding a warping function from the template image $I_0$ to the current image $I_n$ could then be stated as the problem of minimizing the error function with respect to camera and depth map parameters.

There are two minor issues that we should also address: Because quaternions are used to represent the rotation of the camera frame, we need to constrain the corresponding parameters $\mathbf{q}_n$ to represent a unit quaternion, and thus, a unit vector. This can trivially be formulated as a constraint $h_1(\mathbf{q}_n) = 0$ with $h_1(\mathbf{q}_n) = |\mathbf{q}_n|^2 - 1$. Furthermore, it is well-known that reconstruction from monocular images can only be done up to scale. However, it is desireable then at least to enforce a constant scale during the reconstruction process. This can be achieved with the formulation of a constraint $h_2(\mathbf{d}_n) = 0$ with $h_2(\mathbf{d}_n) = S_{\mathbf{d}_n}(u_1,v_1) - l$ for some constant $l$.

Since through optimizing above function, we implicitly try to track point positions through intensity values, our approach will have difficulties tracking points in areas with completely homogeneous intensity. Thus, whereever possible, the reference points are chosen from the ROI in such a way that they lie at pixel positions where the image derivative is non-zero.

Furthermore, reference points should be distributed in the region of interest such that the parameters determining the depth map are well constrained. For a B-Spline depth map model, one will, e.g., need at least a number of reference points that is equal to the number of control points used.

## 2.3 Optimization Method

It is clear that, to actually recover the model parameters from the scene, we need some method to minimize the cost function described above. Since we are dealing with a constrained problem, an adequate method for optimization is Sequential Quadratic Programming (SQP). For a more detailed description of the method, the reader is referred to [15].

The basic idea is as follows: Let $f : \mathbb{R}^k \to \mathbb{R}$ be the scalar function to be minimized, and let $h : \mathbb{R}^k \to \mathbb{R}^l$ be a function that describes a constraint of the form $h(x) = 0$ on solutions. It is well-known that for such problems, the so-called Karush-Kuhn-Tucker (KKT) conditions must hold for any value $x^*$ that is a minimum. These conditions can be formulated in equation form as:

$$\begin{pmatrix} \nabla L(x,\lambda) \\ h(x) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \text{with} \quad \mathscr{L}(x,\lambda) = f(x) + \lambda^T h(x). \tag{5}$$

The term $\lambda \in \mathbb{R}^l$ is the Lagrange multiplier associated with the minimum. This is, in general, a nonlinear system of equations. The Lagrange-Newton-Method can be applied to these equations, and we can compute an update $\Delta x$ to $x$ and a new Lagrange multiplier $\lambda^+$ by solving the equation system

$$\begin{pmatrix} \nabla^2_{xx}\mathscr{L}(x,\lambda) & \nabla_x h(x) \\ \nabla_x h(x)^T & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \lambda^+ \end{pmatrix} = - \begin{pmatrix} \nabla_x f(x) \\ h(x) \end{pmatrix}. \tag{6}$$

Ultimately, we need to compute the Hessians $\nabla^2_{xx}\mathscr{L}$ as well as the transposed Jacobian $\nabla_x h$ of $h$. Since $f$ is, in our case, a quite complex composition of multi-dimensional functions, it is not straightforward to compute the full precision Hessian. Instead, it is common practice to use the Gauss-Newton approximation of the Hessian, as detailed below.

In our case, the objective function $f$ is the composition $c \circ g$ of a scalar cost function $c$ with some multi-dimensional comparison function $g$. The cost function could, *e.g.*, be the least-squares cost $g(x) = x^T x$, but since we also want our optimization to be robust against outliers, we will use something more robust, like the Blake-Zisserman [2] cost function. In any case, the Hessian approximation that we are going to use is:

$$\nabla^2_{xx}(c \circ g)(x) \approx (\nabla_x g)(x) \cdot (\nabla^2_{xx} c)(f(x)) \cdot (\nabla_x g)^T(x)$$

A technique introduced for the popular method of Levenberg-Marquardt optimization [10, 13] is addition of a damping term $\lambda I$ to the Hessian. This allows the method to interpolate between Gauss-Newton and gradient descent steps, and greatly enhances the robustness of the method. This idea has been applied with success to the SQP method, e.g., in the work by Gong et al. [7].

The Jacobians of $f$ and $h$ are computed using mainly Automatic Differentiation [8]. The only exception for this is the image function, which is interpolated and derivatives are computed by hand. All of the involved matrices exhibit a high degree of sparsity. After computation of the Jacobian is finished, the approximate Hessian can be evaluated and the QP system is solved repeatedly using a sparse $LDL^T$ Cholesky transformation on the whole

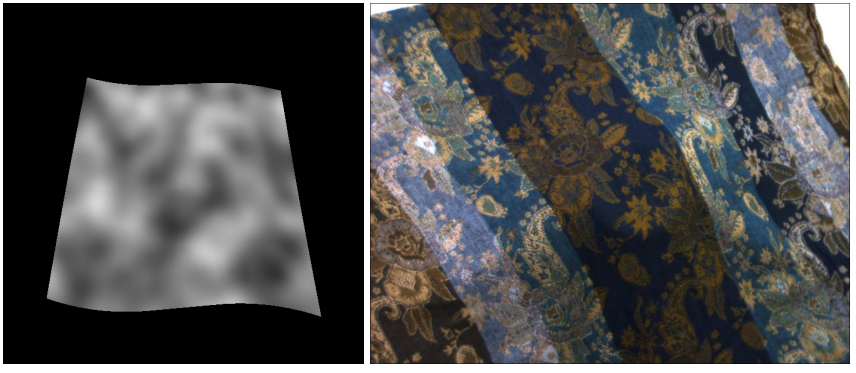


Figure 3: Left: Sample from artificial sequence, Right: Sample from real-world sequence.

system. Our SQP algorithm has been implemented using the efficient Eigen [1] linear algebra library. As has been indicated above, we have used the Blake-Zisserman cost function in all of our experiments.

## 2.4 Dealing with Large Displacements

After we had implemented the optimization process as described above, it was evaluated on some image sequences. We found out that it works well on image sequences where camera movement is sufficiently smooth and no large pixel displacements occur between subsequent frames. However, problems occured when that was not the case. This was to be expected, since the algorithm operates on intensity values and will have trouble aligning with the correct values again if they are too far away.

The typical way to deal with this would be a pyramidal approach: One could start with the optimization on a coarse scale, and then move up to finer scales. This idea could probably be incorporated into our optimization approach. However, the idea has also been used by Lucas and Kanade [12] for their optical flow algorithm, which is well-established and implementations of which are readily available.

Thus, instead of incorporating the pyramidal approach directly into our method, we for now chose to implement a two-step technique: The first step when optimizing the model and aligning it to a new image would be to compute the optical flow between the previous image and the current image and perform optimization based solely on the 2D pixel coordinates of the reference points. The point position estimates derived from the optical flow algorithm shall in the following be denoted by $(u'_i, v'_i)$. The objective function that we use for that optimization is just a simplified version of the cost function for the intensity based optimization, namely

$$\sum_{i=1}^{m} c'((u'_i, v'_i) - \pi(T(\mathbf{p}_n, r_{u_i,v_i}(S_{\mathbf{d}}(u_i, v_i))))) \tag{7}$$

This is basically the original objective function, with the mapping from 2D coordinates to image intensities removed and with a different cost function $c'$ instead of $c$. The cost function we used was the robust Pseudo-Huber cost function [9, p. 619]. The optimization is

[1] http://eigen.tuxfamily.org

performed only with respect to camera parameters, since outliers in optical flow are quite common and tend to significantly disturb surface parameters in a full optimization step.

In the next step, we apply the original intensity based optimization process to realign the points to the reference intensity values and further optimize the surface parameters. This essentially prevents drifting away from the original point intensity values, which could easily occur over time if only optical-flow based optimization was used.

# 3 Results

We have tested our algorithm on a set of artificial rendered image sequences, as well as on sequences of real scenes. The artificial data set was useful for generating images with known ground truth, while the sequences of real images have been used to show that the approach also works in the "real world." As depth map model, we have used B-Spline surfaces of varying order and complexity.

Our first tests were on artificial images generated by a renderer. Here, we show results for one of the used sequences. Figure 3 shows an example image from the sequence, showing a surface with a very difficult to track texture. Because we wanted to get a rough idea of how well traditional approaches would work on that sequence, we ran a SIFT feature detector on some of the images. The feature detection process resulted in about 20 features, depending on the actual image. Even when assuming that all features can be reliably identified through the whole sequence, and that no false feature matchings occur, this is by far not enough to fully describe the complexity of the actual surface. The surface is a quadratic spline surface determined by 25 control points (5 in each direction).

Figure 4 shows a plot visualizing the reconstruction quality achieved by our algorithm as compared to the ground truth of the artificial sequence. The left plot indicates the difference (measured by normalized cross correlation, since the reconstruction is only up to scale) between the surface parameters determined by our algorithm and the ground truth used by the renderer. The reconstruction can be seen to be pretty accurate, even though it is not 100% stable and temporarily diverges from a previously found accurate model. This can be attributed to problems in determining the optical flow. However, as can also be seen from the plot, the algorithm is able to recover after a small number of steps.

The right plot in Figure 4 shows a comparison of camera parameters to ground truth. Camera rotation is compared based on the dot product between rotation axes. Note that the dot product between rotation axes is equal to the cosine of the angle between the axes, thus 1 is the best value one can achieve here. We have also compared rotation angle magnitude and camera translation direction, and results were almost equivalent, thus further plots are omitted.

The artifical sequences have been used because it is really difficult in a real-world scenario to determine the ground truth. Still, it is important to show that our approach also works on actual data generated from a camera. Hence, we have tested our method an scene that was showing a piece of cloth draped over a cup. You can see one image of the recorded sequence in Figure 3. Figure 5 shows two views of the resulting 3D model.

Due to the piece of cloth being quite wrinkled, we were actually expecting more difficulties in reconstructing the real-world scene. However, we have seen that a spline surface with only $12 \times 12$ control points was already enough to model the scene.

As for running times: Our algorithm has been tried on a system with a 1.86 GHz dual core CPU. Using only one of the two CPU cores, framerates of about 10 frames per second

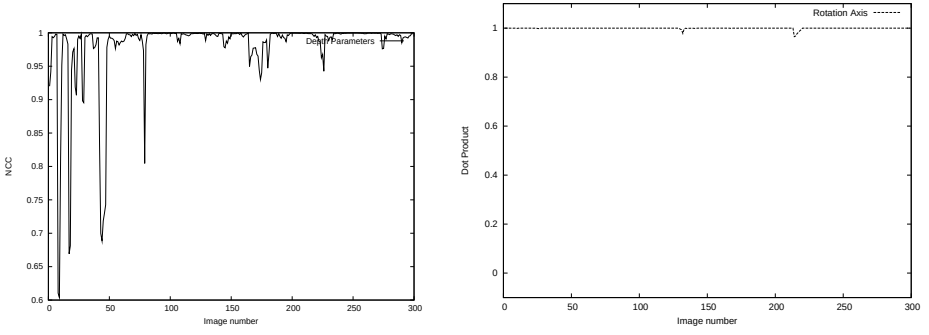



Figure 4: Left: Plot showing comparison between ground truth depth map parameters and recovered depth map parameters. Right: Analogous comparison of camera parameters.

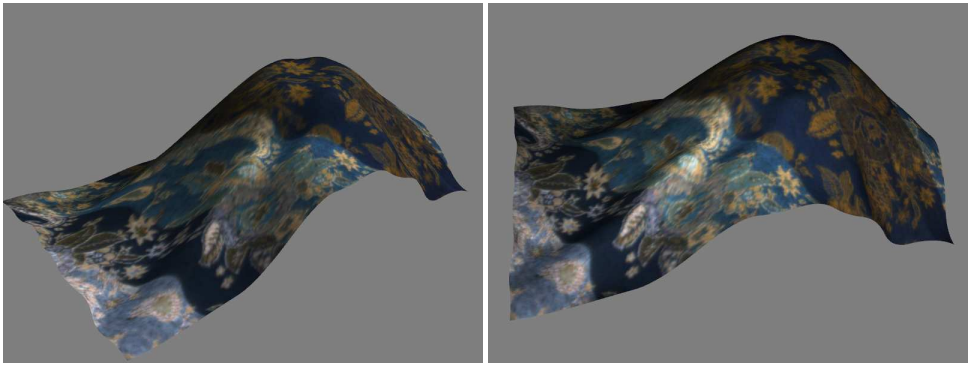


Figure 5: Reconstruction result from real-world scenario.

were achieved. The performance is promising, and we expect it to be possible to further improve performance, e.g., by utilizing GPU hardware.

# 4 Conclusion

The basis for further research has been established with our monocular model recovery and validation algorithm. There are many possible extensions and improvements to this technique.

First of all, while the reference-point based reconstruction works surprisingly well, it would probably constitute a major improvement if we were able to capture, in addition to point intensity values, some characteristics of the surface texture surrounding a reference point, thus introducing a patch-based correlation function. We would expect this to improve the stability and convergence speed of the optimization method considerably.

Furthermore, we did not address the issue of changing illumination conditions. We would like to be able to deal with changes in brightness, but also with specularities, which would, in the current approach, both cause severe problems. However, some techniques for dealing with problems of that kind have already been developed, e.g., normalized cross-correlation matching for brightness-invariant matching. It should be possible to integrate them into our method.

We would also like to extend the approach such that deformable surfaces can be reconstructed and tracked. For tackling this problem, we intend to use a setup of two independently moving cameras. Based on such an idea, we would like to introduce a method for determining deformation parameters, allowing us also to predict and simulate deformations. We see applications for such a technique mainly in medical imaging.

# References

[1] Adrien Bartoli, Mathieu Perriollat, and Sylvie Chambon. Generalized thin-plate spline warps. *Int. J. Comput. Vision*, 88(1):85–110, 2010. ISSN 0920-5691. doi: http://dx.doi.org/10.1007/s11263-009-0303-4.

[2] Andrew Blake and Andrew Zisserman. *Visual Reconstruction*. MIT Press, 1987. ISBN 0-262-02271-0.

[3] F. Brunet, A. Bartoli, N. Navab, and R. Malgouyres. Nurbs warps. In *British Machine Vision Conference (BMVC)*, London, September 2009.

[4] C. Engels, H. Stewénius, and D. Nistér. Bundle adjustment rules. In *Photogrammetric Computer Vision (PCV)*. ISPRS, September 2006.

[5] P. Fua. Using model-driven bundle-adjustment to model heads from raw video sequences. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 46–53 vol.1, 1999. doi: 10.1109/ICCV.1999.791196.

[6] Vincent Gay-Bellile, Adrien Bartoli, and Patrick Sayd. Direct estimation of nonrigid registrations with image-based self-occlusion reasoning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(1):87–104, 2010.

[7] Rubin Gong and Gang Xu. Quadratic surface reconstruction from multiple views using sqp. pages 197–217, 2004.

[8] Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Number 105 in Other Titles in Applied Mathematics. SIAM, Philadelphia, PA, 2nd edition, 2008. ISBN 978–0–898716–59–7.

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

[10] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly Journal of Applied Mathmatics*, II(2):164–168, 1944.

[11] David G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, pages 1150–1157, 1999.

[12] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision (darpa). In *Proceedings of the 1981 DARPA Image Understanding Workshop*, pages 121–130, April 1981.

[13] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963. doi: 10.1137/0111030. URL http://link.aip.org/link/?SMM/11/431/1.

[14] Richard Newcombe and Andrew Davison. Live dense reconstruction with a single moving camera. In *2010 IEEE Conference on Computer Vision and Pattern Recognition CVPR'10*, 2010.

[15] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, August 2000. ISBN 0387987932. URL http://www.worldcat.org/isbn/0387987932.

[16] Pekka Paalanen, Ville Kyrki, and Joni-Kristian Kamarainen. Towards Monocular On-Line 3D Reconstruction. In *Workshop on Vision in Action: Efficient strategies for cognitive agents in complex environments*, Marseille France, 2008. Markus Vincze and Danica Kragic and Darius Burschka and Antonis Argyros. URL http://hal.inria.fr/inria-00325795/en/.

[17] Q. Pan, G. Reitmayr, and T. Drummond. ProFORMA: Probabilistic Feature-based On-line Rapid Model Acquisition. In *Proc. 20th British Machine Vision Conference (BMVC)*, London, September 2009.

[18] D. Pizarro and A. Bartoli. Feature-based non-rigid surface detection with self-occlusion reasoning. In *Fifth International Symposium on 3D Data Processing, Visualization and Transmission*, May 2010.

[19] Nicholas A. Ramey, Jason J. Corso, William W. Lau, Darius Burschka, and Gregory D. Hager. Real Time 3D Surface Tracking and Its Applications. In *Proceedings of Workshop on Real-time 3D Sensors and Their Use (at CVPR 2004)*, 2004.

[20] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, May 2006. doi: 10.1007/11744023_34. URL http://mi.eng.cam.ac.uk/~er258/work/rosten_2006_machine.pdf.

[21] Ying Shan, Zicheng Liu, and Zhengyou Zhang. Model-based bundle adjustment with application to face modeling. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 644–651 vol.2, 2001. doi: 10.1109/ICCV.2001.937687.

[22] Jianbo Shi and Carlo Tomasi. Good features to track. In *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593 – 600, 1994.

[23] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *ICCV '99: Proceedings of the International Workshop on Vision Algorithms*, pages 298–372, London, UK, 2000. Springer-Verlag. ISBN 3-540-67973-1.