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:	D14
Deliverable Title :	Integrated perception/context model extended to the perception of hand-environment interaction for an initial set of objects
Type (Internal, Restricted, Public):	PU
Authors	M. Vincze, M. Richtsfeld, L. Lefakis, M. Pascual; D. Burschka, C. Papazov; N. Bergström, J. Bohg, D. Kragic;
Contributing Partners	TUW, TUM, KTH

Contractual Date of Delivery to the EC:	28-02-2009
Actual Date of Delivery to the EC:	28-02-2009

Contents

Chapter 1

Executive Summary

This report presents the work of year one in WP4. WP4 is concerned with perceiving the object and hand involved in the grasp and all contextual information relevant. With grasp context we refer to the information relevant to the grasp, which at its core includes the grasp points on the objects but also the relationship to the total object, the hand, the task, and the attention on the target object. The overall objective is to perceive grasping points on unknown objects by the end of the project.

Work in year two concerned

- **[Task 4.1] - Acquiring (perceiving, formalising) knowledge through hand-environment interaction** The objective of this task is to obtain many cues for observing the hand to object relationship for grasping. The idea is to use these cues not only to obtain information for the observation of a human handling objects but also for the robot executing the grasping.
- **[Task 4.2] - Perceiving task relations and affordances** The objective is to exploit the set of features extracted in Task 4.1 to obtain a vocabulary of features relevant to the grasping of objects and to learn the feature relations to the potential grasping behaviours and types.

The work in this deliverable relates to the following second year Milestones:

- **[Milestone 4]** Analysis of action-specific visuo-spatial processing, vocabulary of human actions/interactions for perception of task relations and affordances.

The advance in year two is again structured in relation to contextual knowledge of objects from known over familiar to unknown objects.

- Known objects are detected using a new algorithm for multiple 3D object recognition in noisy, outlier corrupted and cluttered scenes. The method is based on a sampling strategy which runs in constant time in the number of input scene points. To the best of our knowledge, there is no other object recognition algorithm in which the main procedure has a constant time complexity. We use the new method to detect known objects in 3D point sets obtained by a stereo reconstruction. Appendix [A] presents this work.
- Familiar objects can be grasped by finding a suitable object representation that allows to transfer grasping experience from similar objects. Previous work on using a 2D shape descriptor for the detection and learning of grasping points on familiar objects has been extended with sparse stereo information and is presented in Appendix [B]. Through this integration, we can additionally to the grasping point also find a suitable approach vector grounded in the 3D structure of the object. Relating to Task 4.2, planar surfaces of any direction in the sparse 3D model serve as an affordance cue for elementary grasping actions.
- To approach "Unknown objects" Appendix [C] presents the 3D estimation of cylindrical objects and top surfaces for grasping from stereo. This enables to detect these two classes of objects. It can use laser depth data as well as stereo depth data. Actually, a combined processing delivers more surfaces and hence more features and a more complete object description. The approach extends

to a large set of objects that have a visible top surface. Hence, this approach is the first to clearly tackle the task of moving towards estimating grasping points for objects not seen before. Moving towards Task 4.2 the cylindrical as well as planar patches are the first two higher level elements to build a grasping affordance description.

- Given objects of more complex shapes an improved convex hull-based segmentation algorithm is used to deliver potential grasping poses. First the algorithm segments the potential core part and all sub-parts of the object. The contribution is a method that segments a point cloud as well as mesh data and in comparison to other segmentation methods the proposed algorithm based on spherical mirroring shows best time performance. This segmentation algorithm can be applied to a reasonable set of objects with different applications presented in Appendix [D].

- Results in the first year [Task 4.1] showed that local image information can be very well used to obtain shape information about objects. Based on this, a new method for learning grasp points in images of previously unseen objects is presented in Appendix [E]. The method resorts to semi-local grasping point shape and a new devised descriptor, to learn a discriminative vocabulary of grasp point models in 2D. To learn the grasping point representation, an annotated database is provided. Newly detected grasp points can be used to bootstrap the learned models after the appropriated validation - e.g. through human interaction or simulation - of the grasp points in order to enrich the previous vocabulary towards an incremental-learning approach. Extensive evaluations show that the method outperforms previous work, though future work is still necessary to use sparse stereo (see also Appendix [B]) to obtain 3D features.

  This work is complemented by an investigation to learn the importance between image features and their geometrical distribution within an object class regarding generic object recognition, which is presented in Appendix [F]. The method, which can be used with different image features - e.g. appearance or shape -, exploits the feature's distribution to obtain a robust object class model. This will be used to move towards a comprehensive description of shapes combining the visual features, that will be further used as context to improve the previously presented grasping point detection mechanism.

- A series of works is concerned with the hand-object interaction.

  Objects reconstructed from stereo cameras mounted on the head of the robot result in 2.5D representation of the scene where multiple surfaces facing away from the robot are not visible to the system. These faces cannot be validated by the robot in the set-up and assumptions need to be made to complete the shape representation. Only a complete 3D shape of an object allows a successful grasp planning on the object. The research presented in [Appendix G] allows a completion of the object from a camera system moved in the scene. This can be applied to both a robot moving around the scene or a camera-in-hand moved by the robot to complete the view. The system performs not only a robust 3D reconstruction but also is capable of estimation of extrinsic and intrinsic camera parameters during the exploration. A novel model validation approach allows hereby a reconstruction of surfaces with partially homogeneous areas. The hypotheses about the surface properties are encoded in a mechanical model of the surfaces where the corresponding smoothness and stiffness parameters can be encoded for different parts of the surface independently. This allows also a meaningful completion of the reconstructed data in areas, where missing texture does not provide any additional information.

- In preparation for applications of the robotic system in hushed scenarios with very sparse texture on the surfaces, structured light approaches were added to the processing chain of the visual system. Scenes with low texture are boosted with additional texture projected onto them that allows a robust reconstruction in areas with no or little texture (see Fig. 1.1).

- A novel active stereo system was develop at TUM with a specific aim on camera-in-hand application, where a second camera was replaced by an active DLP (projector) that projects a calibrated pattern onto the scene that is sensed by the camera mounted in a pre-calibrated location relative to the projector (Fig. 1.2). This allows also a projection of additional information by the robot during the interaction to simplify the human robot interaction in manipulation experiments. The projection is light-weight and can be supplied directly from the USB port of the computer processing the stereo information (to be submitted to CVPR International Workshop on Projector-Camera Systems).

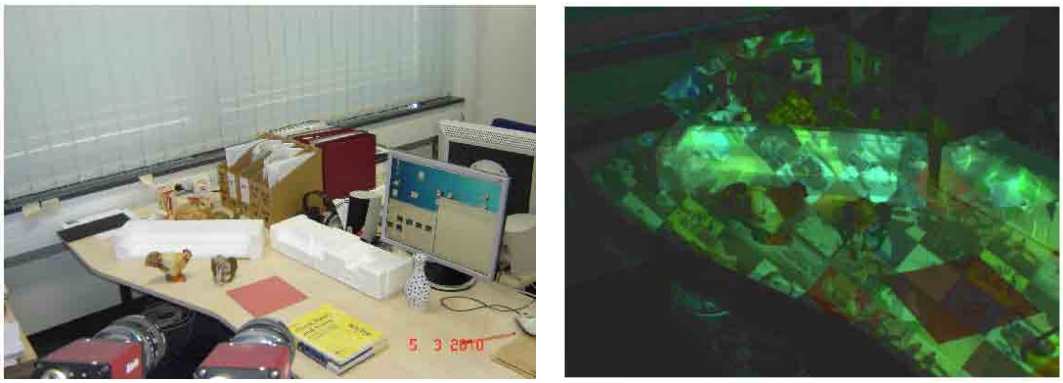

Figure 1.1: Additional texture projected in one frame onto the scene improves the reconstruction of the surface details in homogeneous areas.

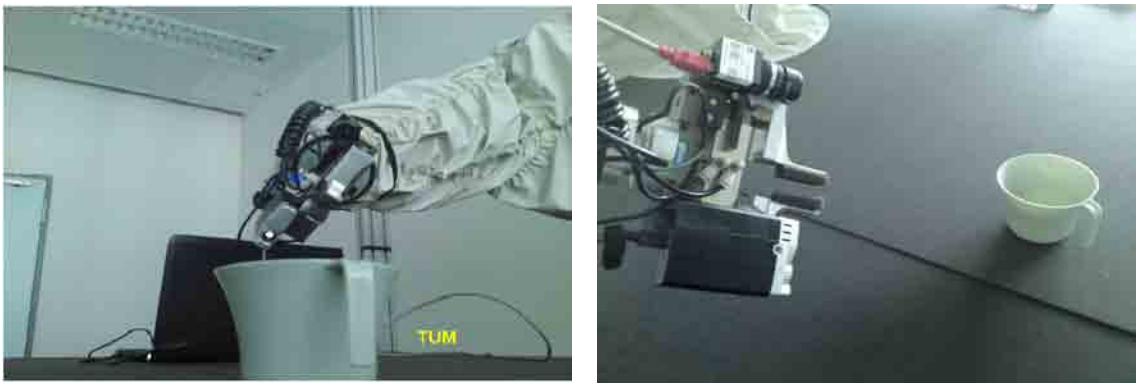

Figure 1.2: One of the cameras of a stereo setup is replaced by an active DLP (digital light processor) to project calibrated texture onto the scene.

- Finally, in collaboration with WP5, the pose change during the human interaction with the robot is tracked by the action analysis system develop within WP5 [1]. This system is capable of monitoring of motion trajectories during handling of objects. This capability can also be used during manipulation attempts by the robot to verify the stability of the grip applied to the object. The system is capable of tracking a relative position to any given reference. The reference can be the background scene which results in a trajectory in 3D through the space or it can be a motion relative to the gripper in which case the stability of the grasp is verified.

[1] See paper in the Deliverable of WP5: Petsch and Burschka: Estimation of Spatio-Temporal Object Properties for Manipulation Tasks from Observation of Humans; ICRA 2010, accepted.

Appendix A

Appendix A: Attached Papers

A Chavdar Papazov, Darius Burschka: "Sampling in Constant Time for 3D Object Detection in Noisy and Cluttered Scenes"; draft version, final version to be submitted to ECCV 2010.

B Niklas Bergstrm, Jeannette Bohg, and Danica Kragic: Integration of Visual Cues for Robotic Grasping; ICVS - Int. Conf. on Computer Vision Systems, 2009.

C Mario Richtsfeld, Markus Vincze: Robotic Grasping from a Single View; Proceedings of RAAD - Robotics in AlpeAdriaDanube Region, 2009.

D Mario Richtsfeld, Markus Vincze: Point Cloud Segmentation Based on Radial Reflection; International Conference on Computer Analysis of Images and Patterns CAIP, 2009.

E L. Lefakis, M. Pascual, H. Wildenauer: Boosted Edge Orientation Histograms for Grasping Point Detection; ICPR 2010, submitted.

F M. Pascual, H. Wildenauer: Combining Geometry and Local Appearance for Object Detection; ICPR 2010, submitted.

G Oliver Ruepp, Darius Burschka: Towards On-Line Intensity-Based Surface Recovery from Monocular Images; RSS 2010, submitted.

Sampling in Constant Time for 3D Object Detection in Noisy and Cluttered Scenes

Chavdar Papazov and Darius Burschka

Machine Vision and Perception Group (MVP)
Department of Computer Science
Technische Universität München, Germany
email: {papazov, burschka}@in.tum.de

Abstract. In this paper we propose a sampling strategy that runs in constant time and allows for efficient 3D object detection in noisy, outlier corrupted and cluttered scenes. We assume that each object is represented by a model consisting of a set of points with corresponding surface normals. The scene should be given in form of a range image. Our method detects 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 constant time in the number of scene points. Moreover the approach is conceptually simple and easy to implement. Tests on a variety of real data — obtained by a stereo reconstruction — show that the proposed method performs well on noisy, outlier corrupted and cluttered scenes in which only small parts of the objects are visible.

1 Introduction

Object detection is one of the most fundamental problems of computer vision. Most algorithms fall into two general classes. One class consists of methods operating on two-dimensional images. These methods are sensitive to changes in viewpoint and illumination. In recent years, advances in 3D geometry acquisition technology have led to a growing interest in object detection techniques which work with three-dimensional data. Moreover, if a three-dimensional representation of objects and scene is available the detection procedure does not have to deal with viewpoint and illumination issues.

Referring to [1] the object detection problem can be stated as follows. Given a set $\mathcal{M} = \{\mathbf{M}_1, \ldots, \mathbf{M}_m\}$ 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 detection algorithm is a set $\{(\mathbf{M}_{k_1}, T_1), \ldots, (\mathbf{M}_{k_n}, T_n)\}$ where $\mathbf{M}_{k_j} \in \mathcal{M}$ is a detected model instance and T_j is a transform which aligns a subset of $\mathbf{M}_{k_j}$ to a subset of the scene. In this paper, we discuss a special instance of this problem which is given by the following assumption.

Assumption 1. (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, i.e., the points are ordered in a rectangular two-dimensional grid such that each point $\mathbf{p}_j$ has (besides its coordinates in $\mathbb{R}^3$) unique two-dimensional integer coordinates. Note that each point set can be converted into a range image using z-buffering. This procedure is, of course, not bijective.
(iv) The transform T_j which aligns a subset of the model $\mathbf{M}_{k_j}$ to a subset of the scene is a rigid motion.

Even under these assumptions the problem remains hard because of several reasons: it is a priori not known which of the models are represented in the scene and how they are oriented, the scene points are typically corrupted by noise and outliers, the objects are only partially visible due to scene clutter, occlusion and scan device limitations.

Contributions and Overview In this paper, we introduce an efficient algorithm for solving the object detection problem under the conditions defined in Assumption 1. We make the following contributions:

(i) The way of representing the models using a hash table of pairs of oriented points — first presented in [3] in the context of surface registration — is significantly modified such that it can be used for object detection.
(ii) A new constant time random sampling strategy for fast generation of object hypotheses is introduced.
(iii) We provide an analysis of our sampling strategy to derive the number of iterations needed to detect 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 this paper.

2 Related Work

Object detection should not be confused with object recognition/classification. 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 [4], [5]. 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.

Two major classes of object detection methods are built by the voting approaches and the correspondence based approaches. Well-known voting methods are the generalized Hough transform [6] and geometric hashing [1]. In the generalized Hough transform approach, the space of rigid transforms is discretized

and votes for transforms which map a model to the scene are cast into accumulator bins. The bin with the most votes indicates the desired rigid motion. This procedure has an unfavorable 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. A further disadvantage is the fact that in the case of multiple models one has to match sequentially each one of them against the scene.

The geometric hashing approach [1] is similar to the generalized Hough transform. The main difference is that one does not vote for transform parameters but for pairs consisting of a model and a basis. In this way a simultaneous detection of all models is possible without the need of sequential matching. However, geometric hashing 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.

The second class of object detection methods are designed to solve the correspondence problem between (a subset of) the model points and (a subset of) the scene points. This is usually done using local geometric descriptors. Before detection, the descriptors for the points of all models are computed and stored. At recognition time, a scene point is selected and the descriptor for its local neighborhood is computed. If there is a good match between the scene descriptor and a model descriptor a correspondence between the underlying points is established. This procedure is repeated until a sufficient number of correspondences is computed. The aligning rigid transform is then calculated based on the established correspondences.

There is a vast variety of descriptors which can be used in a correspondence based object detection framework. Johnson and Hebert introduce in their work [7] spin images and use them for object detection. The presented results are impressive, but no tests with noisy or outlier corrupted data are performed. Gelfand *et al.* [8] develop a local descriptor which performs well under artificial noisy conditions (Gaussian noise), but still, defining robust local descriptors in the presence of significant noise and a great amount of outliers remains a difficult task. Other descriptors are curvedness [9], local feature histograms [10] and shape contexts [11], just to name a few. All correspondence based algorithms rely heavily on the assumption that the models to be detected have few distinctive feature points, i.e., points with rare descriptors. In many cases, however, this assumption does not hold. A cylinder, for example, has too many points with similar descriptors. This results in many ambiguous correspondences between a model and the scene and the detection method degenerates to a brute force search.

In our detection technique, we use a robust descriptor in combination with a sampling procedure that runs in constant time. Before we describe the algorithm in detail, we briefly review the surface registration technique presented in [3] because it is of special relevance to our work.

2.1 Fast Surface Registration

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

$$f(\mathbf{u}, \mathbf{v}) = \begin{pmatrix} d_{uv} \\ \alpha_{uv} \\ \beta_{uv} \\ \gamma_{uv} \end{pmatrix} = \begin{pmatrix} \|\mathbf{p}_u - \mathbf{p}_v\| \\ \arccos[\mathbf{n}_u \cdot \mathbf{n}_v] \\ \arccos[\mathbf{n}_u \cdot (\mathbf{p}_v - \mathbf{p}_u)] \\ \arccos[\mathbf{n}_v \cdot (\mathbf{p}_u - \mathbf{p}_v)] \end{pmatrix}. \tag{1}$$

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 the colliding pairs (in least square sense) gives a transform hypothesis for the surfaces.

According to [3], this process is repeated until a hypothesis is good enough, a predefined time limit expires 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.

3 Method Description

Like all object detection methods cited in this paper, our method 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 detected. The second phase is the online detection which is executed on the scene using the model representation computed in the offline phase. In the rest of this section, we describe both stages in detail and discuss the computational complexity of our algorithm.

3.1 Model Preprocessing Phase

In the offline phase, a representation for each model is computed such that efficient detection in cluttered and occluded scenes becomes possible. For a given object model $\mathbf{M}$ we sample all pairs of oriented points $(\mathbf{u}, \mathbf{v}) \in \mathbf{M} \times \mathbf{M}$ for which $\mathbf{p}_u$ and $\mathbf{p}_v$ are approximately at a distance d from each other. For each pair, the descriptor $f(\mathbf{u}, \mathbf{v})$ is computed as defined in (1) and stored in a four-dimensional hash table. Note that in contrast to the technique presented in [3] 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 δ_d. This has several advantages. The space complexity is reduced from $O(n^2)$ to $O(n)$, where n is

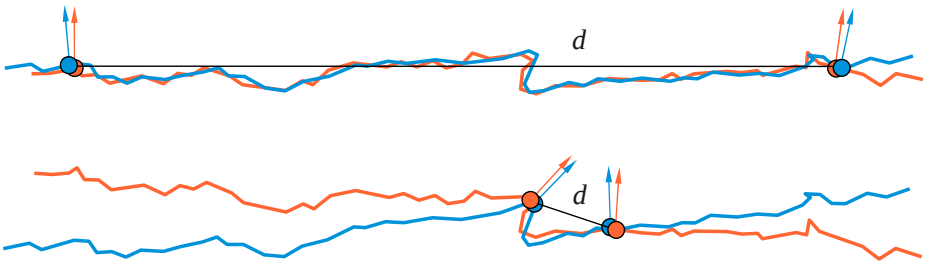

Fig. 1. Stability of wide-pairs. An alignment based on a wide-pair (top) is more stable than an alignment based on a narrow-pair (bottom) (see [12]). In cluttered and occluded scenes the width d of the pair is limited by the extent of the visible portion of the objects.

the number of oriented 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 (see Fig. 1). Another advantage of wide-pairs is due to the fact that (for roughly uniformly sampled surfaces) the larger the distance between the points of a pair the less pairs we have. Thus computing and storing the 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 detection phase and will save computation time.

Note, however, that the pair width d can not be arbitrary large. For a typical value for d (which allows object detection in cluttered and occluded scenes), there are still a lot of pairs with similar descriptors, i.e., there are hash table cells with too many entries. This problem is best illustrated by simple shapes like, e.g., cubes. If the pair width is set to be less than the cube's side length, all pairs with points sampled from one side of the cube will have similar descriptors and will fall within the same hash table cell. To avoid this overpopulation of cells, we remove as many of the most populated cells as needed to keep only $K\%$ of the pairs in the hash table ($K < 100$). This strategy, of course, 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 $\mathbf{M}_1, \ldots, \mathbf{M}_m$ is computed by processing each $\mathbf{M}_j, j = 1, \ldots, m$ 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. Thus new models can be added to the hash table without recomputing it. In the next section, we will see how this model representation allows for simultaneously object detection without trying to match sequentially all models against the scene data.

3.2 Online Detection Phase

As already mentioned in the introduction, the scene $\mathbf{S} = \{\mathbf{p}_1, \ldots, \mathbf{p}_s\} \subset \mathbb{R}^3$ has to be in form of a range image. The output of the algorithm is a list $\mathcal{T} =$

$\{(\mathbf{M}_{k_1}, T_1), \ldots, (\mathbf{M}_{k_n}, T_n)\}$, where $\mathbf{M}_{k_j}$ is a detected model instance and T_j is the associated rigid transform which aligns a subset of $\mathbf{M}_{k_j}$ with a subset of the scene. The overall procedure can be outlined as follows:

1. Initialization
   (a) Preprocess the scene $\mathbf{S}$ to produce a modified scene point set $\mathbf{S}^*$.
   (b) $\mathcal{T} \leftarrow \emptyset$ (an empty solution list).
2. Compute a number of iterations N needed to achieve a probability for successful detection 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 $f_{\mathbf{uv}} = f(\mathbf{u}, \mathbf{v})$ according to (1).
6. Use $f_{\mathbf{uv}}$ as a key to the model hash table to retrieve the oriented model point pairs $(\mathbf{u}_m, \mathbf{v}_m)$ similar to $(\mathbf{u}, \mathbf{v})$.
   [repeat for each $(\mathbf{u}_m, \mathbf{v}_m)$]
   (a) Get the model $\mathbf{M}$ of $(\mathbf{u}_m, \mathbf{v}_m)$.
   (b) Compute the rigid transform T that best aligns $(\mathbf{u}_m, \mathbf{v}_m)$ 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 μ.
   [end repeat]

[end repeat]

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

Step 1, Initialization For our algorithm to be fast, we need to search efficiently for closest points (in steps 4 and 6c) 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. Although the order of the scene points given by the 2d range image grid defines such a structure, it 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 [13]. The full leaves of an octree — these are the leaves 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).

Let $\mathcal{O}$ be an octree and $\mathcal{O}(i, j, k)$ be a full leaf with coordinates $(i, j, k) \in \mathbb{Z}^3$. A neighborhood $\mathbf{N}(i, j, k)$ for $\mathcal{O}(i, j, k)$ is given as

$$\mathbf{N}(i,j,k) = \{\mathcal{O}(x,y,z) \; : |x-i| \leq 1, \, |y-j| \leq 1, \, |z-k| \leq 1, \\ \mathcal{O}(x,y,z) \text{ is a full leaf and } x,y,z \in \mathbb{Z}\}. \tag{2}$$

Based on (2), we define a K-ring neighborhood $\mathbf{N}_K(i,j,k)$ for $K \geq 1$ as

$$\mathbf{N}_K(i,j,k) = \bigcup_{x,y,z \in \{-K+1,\ldots,K-1\} \subset \mathbb{Z}} \mathbf{N}(i+x, j+y, k+z) \tag{3}$$

Points which are lying in the same or in neighboring leaves are close to each other in the sense of the Euclidean metric in $\mathbb{R}^3$. In step 1a of the algorithm, we down-sample $\mathbf{S}$ by constructing an octree for a given leaf size L and 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 lying in this leaf. 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 according to (3).

Step 2, Number of Iterations This step involves the computation of the number of iterations and will be explained in detail in Section 3.3.

Step 3, Sampling In the sampling stage, we make extensive use of the scene octree. 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 [13]. 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 for the points in the K-ring neighborhood of $\mathbf{p}_u$ and $\mathbf{p}_v$, respectively. $\mathbf{n}_u$ and $\mathbf{n}_v$ are set to be the eigenvectors corresponding to the smallest eigenvalues of the covariance matrix of the points in the K-ring 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 $f_{\mathbf{uv}} = f(\mathbf{u}, \mathbf{v})$, as defined in (1). In step 6, $f_{\mathbf{uv}}$ is used as a key to the model hash table (computed in the offline phase, see Section 3.1) to retrieve all model pairs $(\mathbf{u}_m, \mathbf{v}_m)$ which are similar to $(\mathbf{u}, \mathbf{v})$. For each $(\mathbf{u}_m, \mathbf{v}_m)$, the model $\mathbf{M}$ corresponding to $(\mathbf{u}_m, \mathbf{v}_m)$ is retrieved (step 6a) and the rigid transform T which best aligns $(\mathbf{u}_m, \mathbf{v}_m)$ 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 μ.

The Acceptance Function μ measures the quality of a hypothesis $(\mathbf{M}, T)$ and consists of a support term and a penalty term.

The support term, μ_S, is proportional to the number m_s of transformed model points (i.e., points from $T(\mathbf{M})$) which fall within a certain ϵ-band of the

scene. More precisely, $\mu_S(\mathbf{M}, T) = m_s/m$, where m is the number of model points. If n is the number of scene points, a naïve implementation of μ_S would require $O(mn)$ number of distance computations each one consisting of expensive power raising and square rooting. We use a fast approximation of the naïve method which counts the number of transformed model points which fall within a full leaf of the scene octree. This procedure runs in $O(km)$ time, where k is the depth of the octree. Note that k is significantly smaller than n. Furthermore, instead of power raising and square rooting only simple number comparisons are performed.

The penalty term, μ_P, is proportional to the size of the transformed model parts which occlude the scene. This is the only stage of the algorithm where we make use of Assumption 1, namely that 1(ii) the models are representing non-transparent objects and 1(iii) the scene is in form of a range image. It is clear that in a scene viewed by a camera a correctly detected non-transparent object can not occlude scene points reconstructed from the same viewpoint. We penalize hypotheses which violate this condition. The penalty term is approximated very efficiently by counting the number m_p of transformed model points which are between the projection center of the range image and a full octree leaf 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 in 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 their corresponding hypotheses 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 n points and a model instance $\mathbf{M}$ therein consisting of m points. We call $S_M = m/n$ the relative size of $\mathbf{M}$. Let P_M denote

the probability of detecting $\mathbf{M}$ in a single iteration. The probability of at least one detection after N iterations is given by $1-(1-P_M)^N$. In order to achieve a predefined success probability P_S we need

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

iterations. Let us now 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

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

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 $P(\mathbf{p}_u \in \mathbf{M}) = m/n = S_M$ we can rewrite (5) to get

$$P_M = KP(\mathbf{p}_u \in \mathbf{M})P(\mathbf{p}_v \in \mathbf{M}|\mathbf{p}_u \in \mathbf{M}) \tag{6}$$

$$= KS_M P(\mathbf{p}_v \in \mathbf{M}|\mathbf{p}_u \in \mathbf{M}). \tag{7}$$

Note that the relative model size S_M does not depend on the number of input points: more scene points means more points which belong to the model instance so the ratio m/n remains the same.

$P(\mathbf{p}_v \in \mathbf{M}|\mathbf{p}_u \in \mathbf{M})$ is the probability that we 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 intersection set $\mathbf{L}$ of the full octree leaves and 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 at least one full octree leaf which belongs to $\mathbf{M}$. Thus $P(\mathbf{p}_v \in \mathbf{M}|\mathbf{p}_u \in \mathbf{M}) \geq 1/|\mathbf{L}|$, where $|\mathbf{L}|$ is the cardinality of $\mathbf{L}$. $|\mathbf{L}|$ is bounded above by the number $N_V < \infty$ of voxels intersected by a sphere with radius d. Since the pair width d and the octree leaf size L are fixed N_V does not depend on the number of scene points.

Setting $C = 1/N_V$ yields $P(\mathbf{p}_v \in \mathbf{M}|\mathbf{p}_u \in \mathbf{M}) \geq C$ and using (7) gives us an underestimate of P_M:

$$P_M \geq KS_M C = const. \tag{8}$$

Substituting (8) for P_M in (4) gives us a conservative estimate of N which is independent of the number of input scene points. This not only proves that this stage of the algorithm has a constant time complexity in the scene points but also guarantees that the model instances will be detected with a probability higher than P_S.

(ii) The number of pairs per hash table cell (see Section 3.1) does not depend on the input scene and thus is a constant factor in the time complexity.

(iii) The acceptance function μ runs in $O(km)$ time, where m is the number of model points and k is the depth of the scene octree. Since the octree leaf size L is fixed k depends only on the extent of the scene point set and not on the number of points. Thus the acceptance function is evaluated in constant time.

Fig. 2. (Left) Cluttered and occluded scenes. Only one image of each stereo pair is shown. (Right) Disparity maps for the scenes on the left calculated by a simple template matching based stereo algorithm.

4 Experimental Results

In this Section, we test our algorithm on two scenarios with different amount of occlusion and scene clutter (see Figure 2). The objects we are looking for are the Amicelli box, the white rectangular box and the yellow cylinder. The reconstructed point clouds are shown in Figures 3 and 4, whereas the detection results are shown in Figures 5 and 6.

5 Conclusion

In this paper we introduced a new algorithm for multiple 3D object detection in noisy, outlier corrupted and cluttered scenes. Our algorithm is based on a sampling strategy which runs in constant time in the number of input scene points. To the best of our knowledge, there is no other object detection method in which the main procedure has a constant time complexity. In the experimental part of the paper, we showed that our algorithm is able to detect objects reliably even when they are only partially visible in the scene.

Fig. 3. (Upper row) Three different views of the scene points reconstructed from the disparity map shown on the top right of Figure 2. Note the noise and the outliers in the background. (Lower row) zoom on the objects in the scene: Amicelli box, rectangular box and cylinder (from left to right). Note that the reconstruction is sparse, noisy and represents only small parts of the objects.

References

1. Lamdan, Y., Wolfson, H.: Geometric Hashing: A General And Efficient Model-based Recognition Scheme. In: Second International Conference on Computer Vision (ICCV), Proceedings. (1988) 238–249
2. Johnson, A., Hebert, M.: Recognizing Objects by Matching Oriented Points. In: Conference on Computer Vision and Pattern Recognition (CVPR), Proceedings. (1997)
3. 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
4. Novotni, M., Klein, R.: 3D Zernike Descriptors for Content Based Shape Retrieval. In: Symposium on Solid Modeling and Applications, Proceedings. (2003) 216–225
5. Wahl, E., Hillenbrand, U., Hirzinger, G.: Surflet-Pair-Relation Histograms: A Statistical 3D-Shape Representation for Rapid Classification. In: 4th International Conference on 3D Digital Imaging and Modeling (3DIM), Proceedings. (2003) 474–482
6. Ballard, D.H.: Generalizing the Hough Transform to Detect Arbitrary Shapes. Pattern Recognition **13** (1981) 111–122
7. Johnson, A., Hebert, M.: Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes. IEEE Trans. PAMI **21** (1999) 433–449
8. Gelfand, N., Mitra, N., Guibas, L., Pottmann, H.: Robust Global Registration. In: Eurographics Symposium on Geometry Processing. (2005) 197–206
9. Koenderink, J.J., van Doorn, A.J.: Surface Shape and Curvature Scales. Image Vision Comput. **10** (1992) 557–564

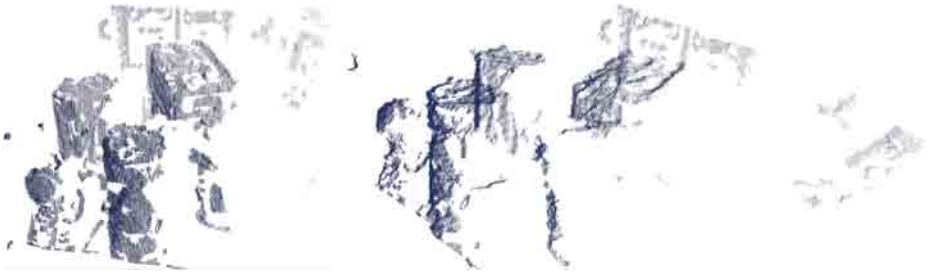

Fig. 4. Two different views of the scene points reconstructed from the disparity map shown on the bottom right of Figure 2. The scene contains a lot of noise and clutter and the objects are only partially visible.

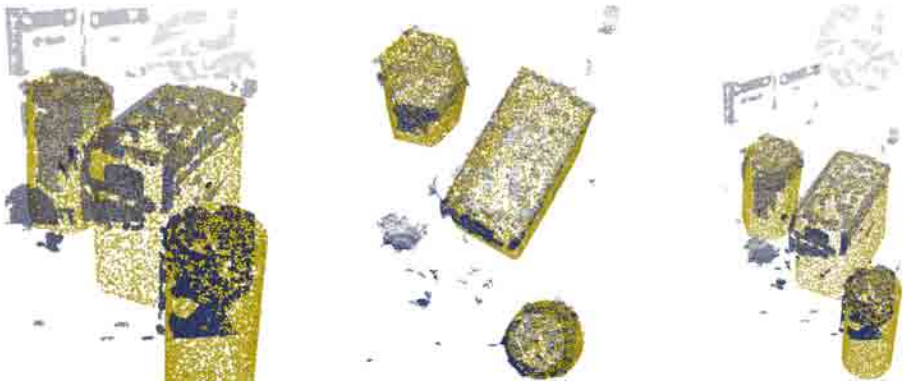

Fig. 5. Object detection results shown from three different viewpoints for the scene depicted in Figure 3. The computation time is about 3 seconds.

10. Hetzel, G., Leibe, B., Levi, P., Schiele, B.: 3D Object Recognition from Range Images using Local Feature Histograms. In: Conference on Computer Vision and Pattern Recognition (CVPR 2001), Proceedings. (2001) 394–399
11. Belongie, S., Malik, J., Puzicha, J.: Shape Matching and Object Recognition Using Shape Contexts. IEEE Trans. PAMI **24** (2002) 509–522
12. Aiger, D., Mitra, N.J., Cohen-Or, D.: 4-points Congruent Sets for Robust Pairwise Surface Registration. ACM Trans. Graph. **27** (2008)
13. de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.: Computational Geometry: Algorithms and Applications. 2 edn. Springer-Verlag (2000)

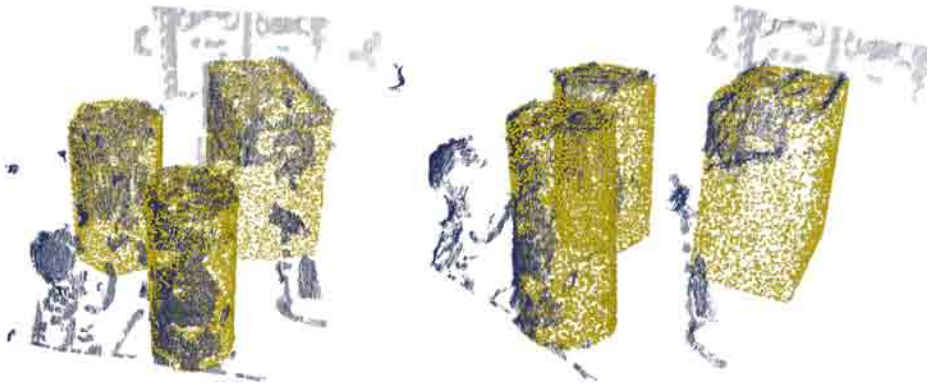

Fig. 6. Detection results shown from two different viewpoints for the scene depicted in Figure 4. The computation time is about 3 seconds.

Integration of Visual Cues for Robotic Grasping

Niklas Bergström, Jeannette Bohg, and Danica Kragic

Computer Vision and Active Vision Laboratory,
Centre for Autonomous System,
Royal Institute of Technology, Stockholm, Sweden
{nbergst, bohg, danik}@csc.kth.se

Abstract. In this paper, we propose a method that generates grasping actions for novel objects based on visual input from a stereo camera. We are integrating two methods that are advantageous either in predicting how to grasp an object or where to apply a grasp. The first one reconstructs a wire frame object model through curve matching. Elementary grasping actions can be associated to parts of this model. The second method predicts grasping points in a 2D contour image of an object. By integrating the information from the two approaches, we can generate a sparse set of full grasp configurations that are of a good quality. We demonstrate our approach integrated in a vision system for complex shaped objects as well as in cluttered scenes.

1 Introduction

Robotic grasping remains a challenging problem in the robotics community. Given an object, the embodiment of the robot and a specific task, the amount of potential grasps that can be applied to that object is huge. There exist numerous *analytical* methods based on the theory of contact-level grasping [1]. Even though these approaches work very well in simulation, they cannot simply be applied to object models reconstructed from typically sparse, incomplete and noisy sensor measurements.How to choose a feasible grasp from incomplete information about the object's geometry poses an additional challenge. This paper introduces a vision based grasping system that infers *where* and *how* to grasp an object under these circumstances. This involves a decision about where the hand is applied on the object and how it is orientated and configured.

Current state of the art methods usually approach this problem by concentrating on one of the two questions. The first group of systems, e.g. [2, 3] typically infers grasps based on 3D features resulting in many hypotheses where to apply the grasp. For each hypothesis, a hand orientation is determined. Heuristics are then applied to prune the number of grasp hypotheses. A drawback of these approaches is the high dependency on the quality of the reconstructed data. The second group of approaches, e.g. [4, 5] relies on 2D data and thus avoids the difficulty of 3D reconstruction. Grasp positions are inferred from a monocular image of an object. The difficulty here is the inference of a full grasp configuration from 2D data only. Additional 3D cues are required to infer the final grasp.

In this paper, we propose a method that aims at integrating 2D and 3D based methods to determine both *where* and *how* to grasp a novel, previously unseen object. The first part of the system matches contour segments in a stereo image to reconstruct a 3D wire frame representation of the object. An edge image containing only successfully matched contour segments serves as the input to the second part of the system. Hypotheses about where a grasp can be applied on the 2D contours are generated. By augmenting the 3D model with this 2D based information, we can direct the search for planar object regions. Plane hypotheses that are supported by contour points with a high grasping point probability will carry a high weight. The normal of these planes then define the approach vectors of the associated grasps. In that way both methods complement one another to achieve a robust 3D object representation targeted at full grasp inference.

This paper is structured as follows. In the next chapter we review different grasp inference systems that are applied in real world scenarios. In Sec. 3 we give an overview of the whole system. Section 4 describes the contour matching approach and Sec. 5 the grasp point inference system. This is followed by Sec. 6 where the integration of these two models is described. An experimental evaluation is given in Sec. 7 and the paper is concluded in Sec. 8.

2 Related Work

The work by [2] is related to our system in several aspects. A stereo camera is used to extract a sparse 3D model consisting of local contour descriptors. *Elementary grasping actions* (EGAs) are associated to specific constellations of small groups of features. With the help of heuristics the huge number of resulting grasp hypotheses is reduced. In our system however, the number of hypotheses is kept small from the beginning by globally searching for planar regions of the object model. [3] decompose a point cloud derived from a stereo camera into a constellation of boxes. The simple geometry of a box and reachability constraints due to occlusions reduce the number of potential grasps. A prediction of the grasp quality of a specific grasp can be made with a neural network applied to every reachable box face. In contrast to that, we drive the search for a suitable grasp through information about 2D grasping cues. These have been shown to work remarkably for grasping point detection in [4, 5].

In [4] an object is represented by a composition of prehensile parts. Grasping point hypotheses for a new object are inferred by matching local features of it against a codebook of learnt *affordance cues* that are stored along with relative object position and scale. How to orientate the robotic hand to grasp these parts is not solved. In [5] a system is proposed that infers a point at which to grasp an object directly as a function of its image. The authors apply machine learning techniques to train a grasping point model from labelled synthetic images of a number of different objects. Since no information about the approach vector can be inferred, the possible grasps are restricted to downward or outward grasps. In this paper, we solve the problem of inferring a full grasp configuration from 2D data by relating the 2D grasping cues to a 3D representation generated on-line.

There exist several other approaches that try to solve the problem of inferring a full grasp configuration for novel objects by cue integration. In [6], a stereo camera and a laser range scanner are applied in conjunction to obtain a dense point cloud of a scene with several non-textured and lightly textured objects. The authors extend their previous work to infer initial grasping point hypotheses by analysing the shape of the point cloud within a sphere centred around an hypothesis. This allows for the inference of approach vector and finger spread. In our approach however, we apply a stereo camera only and are not dependent on dense stereo matching. Due to the application of contour matching, we can obtain sparse 3D models of non-textured and lightly textured objects. [7] showed that their earlier 2D based approach is applicable when considering arbitrarily shaped 3D objects. For this purpose, several views of the object are analysed in terms of potential grasps. While the approach vector is fixed to be either from the top or from the side, the fingertip positions are dependent on the object shape and the kinematics of the manipulator. The best ranked grasp hypothesis is then executed. In our approach, we are not restricted to specific approach vectors whereas our grasp type is assumed to be one of the EGAs defined in [2]. Additionally determining the fingertip positions with the method proposed by [7] is regarded as future work. Finally, in [8] a framework is introduced in which grasp hypotheses coming from different sources e.g. from [2] are collected and modelled as *grasp hypothesis densities*. The grasp hypotheses are strongly dependent on the quality of the 3D object model. The density will therefore contain numerous potential grasps that may not be applicable at all. The authors propose to build a *grasp empirical density* by sampling from the hypotheses that are then grasped with the robot hand. In our case, we are also inferring potential grasps that may not be applicable in practice. However, we are not enumerating hypotheses from different sources but are integrating the information to infer fewer and better hypotheses that are ranked according to their support of 2D grasping cues.

3 System Overview

In our approach the process of grasp inference involves several steps: i) identification, ii) feature extraction, iii) cue integration and iv) grasping. A flow chart of the system is given in Fig. 1 and also shows the utilised hardware.

The first step involves figure-ground segmentation by means of fixation on salient points in the visible scene [9]. A combination of peripheral and foveal cameras is used that are mounted on a kinematic head. Figure 1 (b) and (c) show the left peripheral and foveal views of the head and (d) shows the segmented object.

In this paper, we focus on the feature extraction and cue integration. Full 3D reconstruction of objects with little or no texture from stereo vision is a difficult problem. However, it is debatable if a complete object model is always needed for grasping [7]. We propose a representation that is extractable from real world sensors and rich enough to infer how and where to grasp the considered

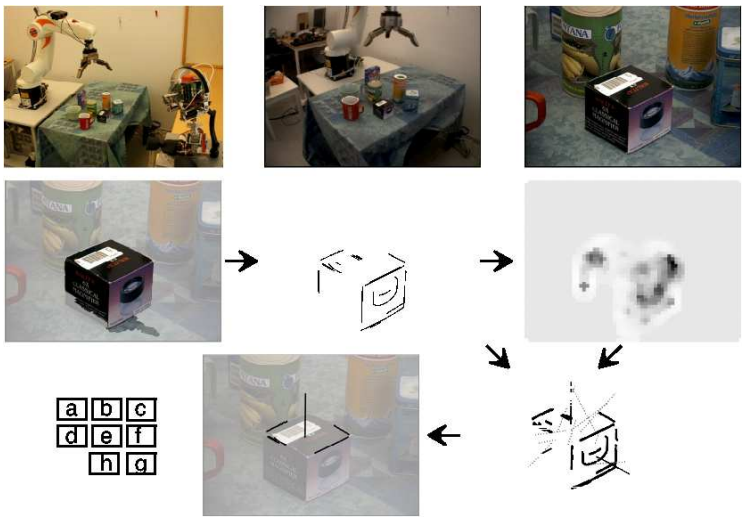

Fig. 1. (a): System setup with 6 DoF KUKA arm, a 7 DoF SCHUNK hand and the ARMAR 3 stereo head. (b,c): Left peripheral and foveal views. d-h: The steps of the grasping system.

object. A general observation that has driven our choice of representation is that many objects in a household scenario, including cups, plates, trays and boxes have planar regions. According to [2] these regions along with their coplanar relationships afford different EGAs. These grasps represent the simplest possible two fingered grasps humans commonly use.

The several steps to build such an object model composed of surfaces are shown in Fig. 1 (d-h). In the segmented foveal view (d) edges are detected and matched across the stereo images to form a 3D wire frame model (e). The projection of this wireframe in one of the images is used to predict where to grasp the object (f). The 3D model is then augmented with this information to detect planar regions that are supported by contour points with a high probability of being graspable (g). The four hypotheses with largest support are indicated with black lines, the others with dashed grey lines. The resulting surfaces provide hypotheses for how to grasp the object. The best hypothesis with respect to plane support and kinematic restrictions of the arm-hand configuration is finally shown in (h).

4 Partial 3D Reconstruction of Objects

Dynamic Time Warping (DTW) is a dynamic programming method for aligning two sequences. The method is described in detail in [10]. Below we give a brief overview of the key points of the algorithm, which is an extension to [11]. The different steps of the method are given in Fig. 2. The leftmost image shows the left foveal view of the object. Canny is used to produce an edge image from

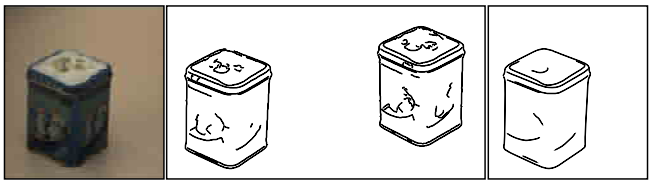

Fig. 2. Left: Left foveal view of object. **Middle**: Contours from left and right foveal views. **Right**: Successfully matched contours.

which connected edge segments (contours) are extracted. Spurious contours are filtered out by restricting their curvature energy and minimum length. The middle image pair shows the contour images from the left and right foveal views. Matching is performed between these two views. DTW is used both for solving the correspondence problem, i.e. which contour that belongs to which, and the matching problem, i.e. which point in the left contour corresponds to which point in the right contour. The latter is performed by calculating dissimilarities between the two contours based on the epipolar geometry, and finding the alignment that minimises the total dissimilarity. The former is performed by integrating the dissimilarity measure with gradient and curvature cues. This is one extension to [11], who could solve the correspondence problem more easily. Another difference is the extension of DTW to handle open and partial contours.

Many contours on the object surface correspond to texture. For 3D reconstruction, as well as 2D grasping point detection as described in Sec. 5, we are only interested in contours belonging to actual edges on the object. As seen in the middle image in Fig. 2, many contours stemming from texture do not have a corresponding contour in the other image and thus will be filtered in the DTW algorithm. Furthermore, shorter contours with higher curvature are less likely to be matched due to a too high total dissimilarity. The resulting matching is used to generate a sparse 3D model of the object.

5 Detecting Grasping Points in Monocular Images

Given the wireframe model reconstructed with the method introduced in the previous section, we search for planar regions that afford EGAs. As it will be shown later, fitting of planes to this raw model will result in many hypotheses stemming from noise and mismatches. In this section, we introduce a method that forms heuristics for searching and weighting of hypotheses according to their *graspability*. We introduce knowledge that comprises how graspable object parts appear in 2D and how these cues are embedded in the global shape of common household objects. Here, we are following a machine learning approach and classify image regions as graspable or not. We briefly describe how our feature vector is constructed and how the training of the model is done. A more detailed description can be found in [12].

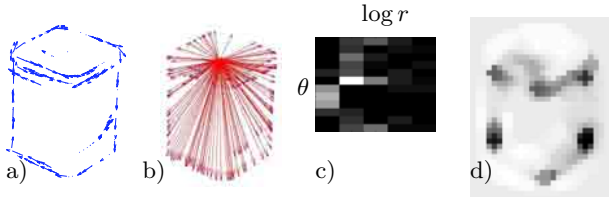

Fig. 3. Example of deriving the shape context descriptor for the matched contours shown in Fig. 2. (a) Sampled points of the contour with tangent direction. (b) All vectors from one point to all the other sample points. (c) Histogram with 12 angle bins and 5 log-radius bins. (d) Classification of the descriptors in each image patch.

Shape context (SC) [13] is a widely applied descriptor that encodes the property of *relative shape*, i.e. the relation of the global object shape to a local point on it. The descriptor is invariant to 2D rotation, scale and translation. Figure 3 shows an overview on the computation of SC. N samples are taken with a uniform distribution from the contour. For each point we consider the vectors that lead to the remaining $N-1$ sample points. We create a log polar histogram with K angle and radius bins to comprise this information. For the feature vector, we subdivide the image into 10×10 pixel patches. A patch descriptor is composed by accumulating the histograms of all those sample points that lie in the patch. We calculate the accumulated histograms at three different spatial scales centred at the current patch and concatenate them to form the final feature descriptor.

This feature vector is then classified by a grasping point model as either graspable or not. This model is an SVM that we trained off-line on the labeled database developed in [5]. An example of the classification results with an SVM trained on a pencil, a martini glass, a whiteboard eraser and two cups is shown in Fig. 3 d). Patches with a high graspability are characterised by rounded and parallel edges which indicate similarity to handles, rims or thin elongated structures. However, the approach direction is not easily inferred.

6 Cue Integration

To generate grasping hypotheses, we are interested in finding planar surfaces, i.e. finding contours that lie in the same plane. The set of plane hypotheses is defined as $\mathit{\Pi} = \{\pi_i\},\ \pi_i = (\overline{n}_i, \mu_i)$, where $\overline{n}_i$ is the normal and μ_i the centre point on the plane. When searching for hypotheses, we start be selecting a point p_1 on one of the contours and a point p_2 nearby. We assume that these points are likely to lie in the same planar region(s) on the object. Then, there will be a third point p_3 on the remaining contours that defines such a region. By searching over the set of potential p_3, we try to find all these planes. Given p_1, p_2 and p_3, a plane hypothesis $\tilde{\pi}_i$ can be defined. Since the depth is quantised, the three selected points may produce a non optimal plane. Therefore we use RANSAC [14] over small contour regions defined by these points to optimise the plane. The hypothesis is accepted or rejected depending on the amount of contour points neighbouring p_1, p_2 and p_3 that are close enough to $\tilde{\pi}_i$. If accepted a more exact π_i is computed by performing regression on the full set of contour points

not exceeding a certain distance to $\tilde{\pi}_i$. After the planes related to p_1 have been found, a new p_1 is selected and the procedure is repeated.

In order to restrict the search, whenever a contour point has been assigned to a plane it will be unavailable when choosing p_1. This will, apart from reducing the computational time, drastically reduce the number of hypotheses and remove most duplicates. This puts requirements on how the selection of p_1 is made. If chosen badly, it is possible to miss good hypotheses if for instance p_1 is not chosen from a contour corresponding to an actual edge. To solve this problem we use the information from the 2D grasping point detection. We start by extracting local maxima from the classification result. Because contour points in these regions are likely to be graspable, we choose p_1 from among these. As we will show in Sec. 7, this will result in a faster and more reliable search than randomly choosing p_1. The search for hypotheses continues until all points from regions with local maxima have been considered. We enforce that the normals are in the direction pointing away from the mean of all contour points.

As a final step planes are ranked according to graspability. For each plane

$$support(\pi_i) = \sum_{j \in \{all\ \ points\}} w(p_j) * P(p_j)/(\lambda_1 + \lambda_2) \tag{1}$$

where $w(p_j) = 1 - 2\frac{1}{1+e^{-d(p_j,\pi_i)}}$, $d(p_j, \pi_i)$ is the distance of p_j to the plane π_i, $P(p_j)$ is the probability that p_j is a grasping point, and $\lambda_{1,2}$ are the two largest eigenvalues from PCA over the inliers. This gives a support value that favours planes with dense contours whose points have a high graspability. Estimated planes may have a normal that does not correspond perfectly to the normal of the real plane. This plane will still get support from points that are close and are likely to stem from the real plane. Normalising with the sum of the eigenvalues ensures that planes without gaps are favoured over planes formed only from e.g. two sides. It also reduces the support for planes with points from falsely matched contours that will lie far from the actual object. Moreover, by calculating the eigenvalues we are able to filter out degenerate planes that have a small extension in one direction.

The normals of the final plane hypotheses are then defining the approach direction of the grasp and the smallest eigenvector of the related set of contour points the wrist orientation.

7 Experiments

The goal of the proposed method is to generate good grasping hypotheses for unknown objects in a robust and stable manner. Furthermore, as few false positives as possible should be generated. In this section, we will show that this is achieved for objects and scenes of varying geometrical and contextual complexity.

Figure 4 shows different objects used for the experiments. The corresponding matched contours are shown on the row below. The upper right of the figure contains the output of the grasping point detection. Finally, the last row shows

the five planes with best support for each object. These four objects are selected to pose different challenges to our system: The hole puncher has a *complex geometric structure*, but with easily detectable edges. Due to many close parallel contours on the tape roll, we get some *false matches*. The tea canister object is *highly textured*, and its lid has many *parallel edges* which causes problems when finding the top plane. The magnifier box resides in a more complex scene in which Canny produces more *broken edges* that complicate the matching problem.

In all cases the two best hypotheses (red and green) shown in the bottom row are graspable, and correspond to how a human probably would have picked up the objects under the same conditions. For the puncher, the hypotheses give the choice of picking up from the object's front or top. This is an example of one of the benefits of our method: we do not need to constrain the approach direction. In the tape roll case there are several severe mismatches (marked in the figure). These correspond to a depth error of up to 50 cm, and are actually part of three plane hypotheses. Here the normalisation makes sure they get low support. Because of the parallel edges on the tea canister's lid, several hypotheses with good support are found on the top. The red hypothesis gets more support though, as it has more contour points close to the plane. In the case of the magnifier box, matching is harder, and we get much fewer and shorter edges. The longest contour is actually the one corresponding to the image of the magnifier. This affects the results from the support computations since the contours from the sides are not complete. The hypothesis from the right side clearly gets largest support. When finally choosing a grasp configuration kinematic constraints or other preferences will guide which of them to choose.

As mentioned in the previous section, the choice of the starting point is crucial to the performance of plane detection. We compared the method described in Sec. 6 to other approaches like random choice or a systematic search from the longest to the shortest contour. The assumption behind the latter method is that longer contours are more likely to originate from an actual edge of the object rather than from texture. We have performed an extensive evaluation of each method on the data in Fig. 4 to estimate their robustness, and will show how the proposed method outperforms the random and sequential method. Given the same input, all three methods will result in different plane hypotheses for each run due to the application of RANSAC in the plane estimation phase. The quality of a detected plane is measured by Eq. 1.

Figure 5 shows three representative examples for each of the three methods applied to the magnifier box. The two plane hypotheses that have the highest support are red and green. The best results for each method are shown in the leftmost column. Our method produced results similar to the top left example in Fig. 5 most times. The best result for the random selection only contains two hypotheses corresponding to real planes. The other two examples contain cases of missed planes (e.g. the top plane in the middle figure) and wrong planes being preferred over hypotheses corresponding to real planes. As with our method, the sequential selection produces more stable results. However, the problem of missed planes and ranking wrong planes higher than real ones persists.

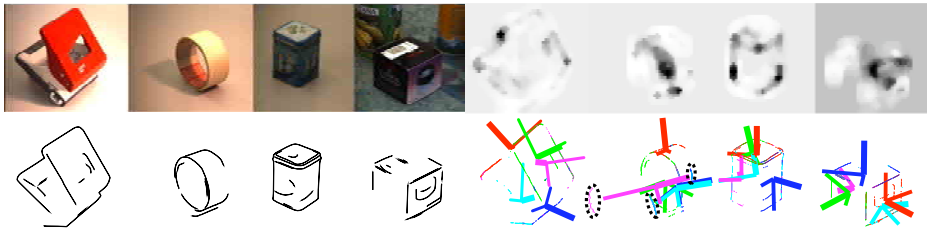

Fig. 4. Four objects, their matched contours, grasping point probabilities and finally the five best hypotheses for each object. The hypotheses are coloured, from best to worst, red, green, blue, cyan, magenta. False matches are circled in black. (Best viewed in colour)

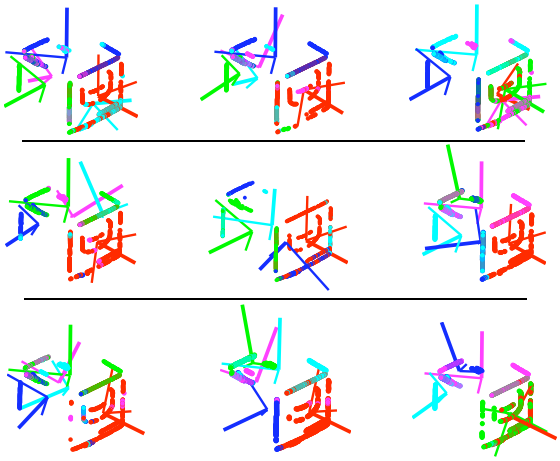

Fig. 5. Top row: Proposed method. Middle row: Random selection. Bottom row: Sequential selection. Colours in the same order as in Fig. 4 (Best viewed in colour)

In cases of simple hardly textured objects in non-cluttered scenes, all three methods have a comparable performance. However, in real world applications we need to deal with objects of arbitrary geometry in complex scenes in which segmentation is hard due to sensory noise, clutter and overlaps.

8 Conclusion

We have presented a method for generating grasping actions for novel objects based on visual input from a stereo camera. Two methods have been integrated. One generates a wire frame object model through curve matching, and associates EGAs to it. The other predicts grasping points in a 2D contour image of the object. The first accurately predicts how to apply a grasp and the other where to apply it. The integration generates a sparse set of good grasp hypotheses. We have demonstrated the approach for complex objects and cluttered scenes.

Our future work will exploit the use of the method in an integrated learning framework. Hypotheses will be generated as proposed and used for picking up objects. The system will then be able to view the object from different directions in order to generate a more detailed model.

Acknowledgments This project has been supported by the EU IST-FP7-IP GRASP (2008-2012) and the Swedish Foundation for Strategic Research through project CORS.

References

1. Nguyen, V.D.: Constructing stable grasps. Int. J. on Robotics Research **8**(1) (1989) 26–37
2. Kraft, D., Pugeault, N., Baseski, E., Popovic, M., Kragic, D., Kalkan, S., Wörgötter, F., Krueger, N.: Birth of the Object: Detection of Objectness and Extraction of Object Shape through Object Action Complexes. Int. J. of Humanoid Robotics (2009)
3. Hübner, K., Kragic, D.: Selection of Robot Pre-Grasps using Box-Based Shape Approximation. In: IEEE Int. Conf. on Intelligent Robots and Systems. (2008) 1765–1770
4. Stark, M., Lies, P., Zillich, M., Wyatt, J., Schiele, B.: Functional Object Class Detection Based on Learned Affordance Cues. In: 6th Int. Conf. on Computer Vision Systems. Volume 5008 of LNAI., Springer-Verlag (2008) 435–444
5. Saxena, A., Driemeyer, J., Kearns, J., Ng, A.Y.: Robotic Grasping of Novel Objects. Neural Information Processing Systems **19** (2006) 1209–1216
6. Saxena, A., Wong, L., Ng, A.Y.: Learning Grasp Strategies with Partial Shape Information. In: AAAI Conf. on Artificial Intelligence. (2008) 1491–1494
7. Speth, J., Morales, A., Sanz, P.J.: Vision-Based Grasp Planning of 3D Objects by Extending 2D Contour Based Algorithms. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. (2008)
8. Detry, R., Başeski, E., Krüger, N., Popović, M., Touati, Y., Kroemer, O., Peters, J., Piater, J.: Learning object-specific grasp affordance densities. In: Int. Conf. on Development and Learning. (2009)
9. Björkman, M., Eklundh, J.O.: Attending, Foveating and Recognizing Objects in Real World Scenes. In: British Machine Vision Conference. (2004)
10. Bergström, N., Kragic, D.: Partial 3D Reconstruction of Objects for Early Reactive Grasping. Technical report, CAS, KTH Stockholm (2009) `www.csc.kth.se/~nbergst/files/techreport09.pdf`.
11. Romero, J., Kragic, D., Kyrki, V., Argyros, A.: Dynamic Time Warping for Binocular Hand Tracking and Reconstruction. In: IEEE Int. Conf. on Robotics and Automation. (May 2008) 2289–2294
12. Bohg, J., Kragic, D.: Grasping Familiar Objects Using Shape Context. In: Int. Conf. on Advanced Robotics. (June 2009)
13. Belongie, S., Malik, J., Puzicha, J.: Shape Matching and Object Recognition Using Shape Contexts. IEEE Trans. on Pattern Analysis and Machine Intelligence **24**(4) (2002) 509–522
14. 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**(6) (1981) 381–395

Proceedings of the RAAD 2009
18th International Workshop on Robotics in Alpe–Adria–Danube Region
May 25-27, 2009, Brasov, Romania

Robotic Grasping from a Single View

Mario Richtsfeld and Markus Vincze

Institute of Automation and Control
Vienna University of Technology
Gusshausstr. 27-29, Vienna, Austria
[rm, vm]@acin.tuwien.ac.at

Abstract. "People have always been fascinated by the exquisite precision and flexibility of the human hand. When hand meets object, we confront the overlapping worlds of sensorimotor and cognitive functions" Castiello (2005). In the last few decades the grasping task has been studied from a psychological, biological and engineering focus but is still unresolved. There exist different solutions for certain cases, however there is still no general valid solution. This paper presents a method for segmentation of a 2.5D point cloud into parts, assembly of parts into objects and calculation of grasping points and poses, which works for rotation symmetrical objects as well as arbitrary objects. The algorithm checks potential collisions between the gripper, the object to be grasped, all surrounding objects and the table top. Thus the algorithm finds the objects, which are graspable without collision. The experimental results show that the presented grasping system is able to detect practical grasping points and poses to grasp a wide range of objects.

Keywords. grasping, laser range scanning, 2.5D point clouds.

1. Introduction

This paper describes the development of a vision based grasping system for unknown objects based on 2.5D point clouds, where the complete scene was scanned from only one single view[1]. We present an algorithm that automatically segments 2.5D point clouds, reassembles rotation symmetrical objects from parts and calculates practical grasping points. The algorithm was developed for simple objects and rotation symmetrical objects, but we achieved also good results on more complex object shapes.

The outline of the paper is as follows: Section 2 introduces our robotic system and its components. Section 3 describes the segmentation of 2.5D point clouds into parts, the assembly of parts into objects and details the merging of clipped rotation symmetrical objects. Section 4 details the calculation of grasping points for rotation symmetrical objects and optimal hand poses for arbitrary objects to grasp and manipulate an object without collision. Section 5 shows the achieved results and Section 6 finally concludes the paper.

1.1. Problem Statement and Contribution

The goal of this work is to show a robust way to calculate possible grasping points for rotation symmetrical objects and grasping poses for unknown objects despite noise, outliers and shadows (two shadows appear, from a single view one from the camera and another one from the laser), which can be caused by specular or reflective surfaces. We calculate collision free hand poses with a 3D model of the used gripper to grasp the objects, as illustrated in Fig. 1[2]. We have decided to point out the general feasibility to realize stable grasps from only one single view. That means that occluded objects can not be analyzed or grasped and we assume that all objects or parts of objects on the table are visible.

The problem of automatic 2.5D reconstruction to get practical grasping points and poses consists of several challenging parts. Objects can be broken into discon-

[1]This work was supported by the EU Project "GRASP" with the grant agreement number 215821.

[2]All images are best viewed in color.

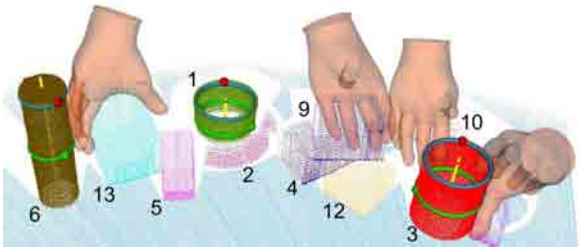

Fig. 1. Detection of grasping points and hand poses. The green points display the computed grasping points for rotation symmetrical objects. The red points show an alternative grasp along the top rim. The illustrated hand poses show a possible grasp for the remaining graspable objects.

nected parts, due to missing sensor data from shadows or poor surface reflectance and we have only information from one single view. Our grasping algorithm was developed for arbitrary objects with a special focus on rotation symmetrical objects, because these objects are some times splitted into two parts and these objects allow a cylindrical grasp and along the top rim a tip grasp, Schulz et al. (2005). To calculate correct grasping points and poses, we need to identify complete objects and therefore reassemble parts belonging to the same object. Thereby we calculate grasping points for rotation symmetrical objects (parts) and grasp poses for arbitrary objects. To realize an unbiased evaluation of our multi step solution procedure, we defined 18 different objects, which are shown in Fig. 2.

1.2. Related Work

In the last few decades the problem of grasping novel objects in a fully automatic way has gained increasing importance in machine vision and there are existing many approaches for grasping quasi planar objects, Sanz et al. (1999). Recatalà et al. (2008) created a framework for the development of robotic applications on the synthesis and execution of grasps. Li et al. (2007) presented a data driven approach to realize a grasp synthesis. Their algorithm uses a database of captured human grasps to find the best grasp by matching hand shape to object shape. Our presented algorithm includes a simple grasping method, where the 3D model of the hand is also used to find a collision free grasp. Ekvall and Kragic (2007) analyzed the problem of automatic grasp generation and planning for robotic hands, where shape primitives are used in synergy to provide a basis for a grasp evaluation process when the exact pose of the object is not available. Their algorithm calculates the approach vector based on the sensory input and in addition tactile information that finally results in a stable grasp. Miller et al. (2004) developed the interactive grasp simulator "GraspIt!" for different hands, hand configurations and objects. The method evaluates the grasps formed by these hands. Goldfeder et al. (2007) presented a grasp planner which considers the full range of parameters of a real hand and an arbitrary object, including physical and material properties as well as environmental obstacles and forces. Our grasping system includes also a collision detection, between our gripper, an arbitrary object and potential environmental obstacles on the table, based on the laser range scanner information. A 3D model based work is presented by El-Khoury et al. (2007). They consider the complete 3D model of one object, which will be segmented into single parts. After the segmentation step each single part is fitted with a simple geometric model. A learning step is finally needed in order to find the object component that humans choose to grasp it. Stansfield (2002) presented a system for grasping 3D objects with unknown geometry using a Salisbury robotic hand, where every object was placed on a motorized and rotated table under a laser scanner to generate a set of 3D points. These were combined to form a 3D model. In our case we do not operate on a motorized and rotated table, which is unrealistic for real world use. The goal is to grasp objects, which are seen from only one view.

2. Experimental Setup

Our approach is based on scanning the objects on the table by a rotating laser range scanner with a pan/tilt unit and execution of subsequent path planning and grasping motion. The robot arm is equipped with a hand prosthesis from the company Otto Bock[3], which we are using as gripper, see Fig. 3. There is a defined pose between the AMTEC[4] robot arm with seven degrees of freedom and the scanning unit. The hand prosthesis has integrated tactile force sensors, which are used to detect a potential sliding of objects, which initializes a readjustment of the grip force applied by the pressure of the fingers. It has three active fingers the thumb, the index finger and the middle finger, the last two fingers are for just cosmetic reasons. The middle between the fingertip of the thumb, the index and the last finger is defined as tool center point (TCP). To calculate a collision free path, we use a commercial path planning tool from AMROSE[5]. The grasping algorithm consists of six main steps, see Fig. 4:

- Raw Data Pre Processing: The raw data points are preprocessed with a smoothing filter to reduce noise and outliers.

[3] http://www.ottobock.de/
[4] http://www.amtec-robotics.com/
[5] http://www.amrose.dk/

Fig. 2. 18 different objects were selected to evaluate our grasp point and grasp pose detection algorithm, from left: 1. Coffee Cup (small), 2. Saucer, 3. Coffee Cup (big), 4. Cuboid, 5. Geometric Primitive, 6. Spray on Glue, 7. Salt Shaker (cuboid), 8. Salt Shaker (cylinder), 9. Dextrose, 10. Melba Toast, 11. Amicelli, 12. Mozart, 13. Latella, 14. Aerosol Can, 15. Fabric Softener, 16. C 3PO, 17. Cat, 18. LINUX Penguin.

- Range Image Segmentation: This step identifies different parts of an object based on a 3D DeLaunay triangulation.
- Merging of Rotation Symmetrical Parts: Finding high curvature points, which indicate the top rim of an object part and fit a circle to these points. Merging of rotation symmetrical parts by matching the calculated circles. Thereby open objects can be identified.
- Approximation of 2.5D Objects to 3D Objects: This step is only important to detect potential collisions by the path planning tool. Thereby we differentiate between:

- Rotation Symmetrical Objects: Add additional points by using the main axis information.
- Arbitrary Objects: The non visible range will be closed with planes, normal to the table plane.

- Grasp Point and Pose Detection:

- Rotation Symmetrical Objects: Calculate potential grasping points with the help of the gained features (open or closed, radius along the top rim, main axis).
- Arbitrary Objects: Calculate potential grasping poses with the principal axis of the top surface.

- Collision Detection: Considering all surrounding objects and the table surface as obstacles, to evaluate the calculated hand pose.

Fig. 3. Overview of the system components and their interrelations.

3. Range Image Segmentation

The range image segmentation starts by detecting the surface of the table with a RANSAC (Fischler et al. (1981)) based plane fit, Stiene et al. (2002). We define an object (part) as a set of points, with distances between neighbors. For that we build a kd tree (Bentley (1975)) to find neighbors and calculate the minimum d_{min}, maximum d_{max} and average distance d_a between all neighboring points, Arya et al. (1998). The segmentation of the point cloud will be achieved with the help of a 3D mesh generation, based on the triangles calculated by a 3D DeLaunay triangulation, as published by O'Rourke (1998). The necessary settings for the mesh generation will be achieved with d_{min}, d_{max} and d_a between all neighboring points. After mesh generation, all segments of the mesh are extracted from the mesh

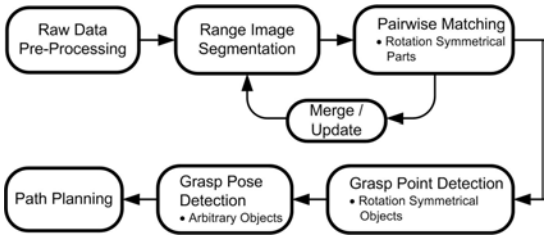

Fig. 4. Overview of our grasp point and gripper pose detection algorithm.

by a connectivity filter, Belmaonte et al. (2004). This step segments the mesh into different components (objects or parts). An additional cut refinement was not arranged. Thereby it can come to an over or an under segmentation, depending on the overlap of the objects, as illustrated in Fig. 5.

Fig. 5. Results after the first segmentation step. Eleven objects are detected, where in reality only ten are. Object no. 1 and 3 are clipped into two parts and object no. 4 and 9 are overlapped. The wrongly segmented objects are red encircled.

As top surface we define the surface of an object from the top view, whereby this surface can also be opened or curved. After the object segmentation step the algorithm finds the top surfaces of all objects using a RANSAC based plane fit and generates a 2D DeLaunay triangulation, with this 2D surface information the top rim points and top feature edges of every object can be detected. For the top surface detection the algorithm uses a pre processing step to find out all points of the object (all points of the generated mesh) with a normal vector in x direction bigger than in y or z direction, $n[x] > n[y] \wedge n[x] > n[z]$, whereby the x direction is normal to the table plane. The normal vectors of all points are calculated with the faces of the generated mesh.

3.1. Pairwise Matching

We developed a matching method, which is limited to rotation symmetrical objects. This method finds the top rim circle of rotation symmetrical objects. A RANSAC based circle fit with a range tolerance of $2mm$ is used. Several tests have shown that this threshold provides good results for our currently used laser range scanner. For an explicit description, the data points are defined as $(p_{x_i}, p_{y_i}, p_{z_i})$ and (c_x, c_y, c_z) is the circle's center with a radius r. The error must be smaller than a defined threshold:

$$|\|\vec{p} - \vec{c}\| - r| \leq 2 \tag{1}$$

This operation will be repeated for every point of the top rim. The run with the maximum number n of included points wins.

$$n = |\{p|\|\vec{p} - \vec{c}\| - r| \leq 2\}| \tag{2}$$

If more than 80% of the rim points of both parts (rotation symmetrical parts) lie on the circle, the points of both parts are examined more closely with the fit. For that we calculate the distances of all points of both parts to the rotation axis, see Equ. 3, the yellow lines represent the rotation axis, see Fig. 1. If more than 80% of all points of both parts agree, both parts are merged to one object, see Fig. 1, object no. 1.

$$d = (\vec{p} - \vec{c}) \times \vec{n} \tag{3}$$

3.2. Approximation of 3D Objects

This step is important to detect potential collisions by the path planning tool from AMROSE. In order to avoid wrong paths and collisions with other objects, due to missing model information, because in 2.5D point clouds every object is seen from only one view, but the path planning tool needs full information to calculate a collision free path. During the matching step the algorithm detected potential rotation symmetrical objects and merged clipped parts. With this information, the algorithm rotates only points along the axis by 360° degrees in 5° steps, which fulfill the necessary rotation constraint. This means that only points will be rotated, which have a corresponding point on the opposite side of the rotation axis (Fig. 5, object no. 1) or build a circle with the neighboring points along the rotation axis, as illustrated in Fig. 5, object no. 6 and Fig. 6a, object no. 1 and 6. By this relatively simple constraint object parts such as handles or objects close to the rotation symmetrical object will not be rotated. For all other arbitrary objects, every point will be projected to the table plane and with a 2D DeLaunay triangulation the rim points can be detected. These points correspond with the rim points of the visible surfaces. So the non visible surfaces can be closed, these surfaces will be filled with points between the corresponding rim points, as illustrated in Fig. 6a. Filling the non visible range with vertical planes may lead to incorrect results, especially when the back side of the objects is far from vertical, but this step is only to detect potential collisions by the path planning tool.

4. Grasp Point and Pose Detection

The algorithm for grasp point detection is limited to rotation symmetrical objects and the grasp poses will be calculated for arbitrary objects. After the segmentation step we find out if the object is open or closed, for that we fit a sphere into the top surface. If there is no point of the object in this sphere we consider the object is opened. Now, the grasping points of all cylindrical objects can be calculated. For every rotation symmetrical object we calculate two grasping points along the rim in the middle of the object (green colored points, as illustrated in Fig. 6a). If the path planner is not able to find a possible grasp, the algorithm calculates alternative grasping points along the top rim of the object near the strongest curvature, as illustrated in Fig. 6a as red points. If it is an open object one grasping point is enough to realize a stable grasp near the top rim. The grasping points should be calculated in such a way that they are next to the robot arm, which is mounted on the opposite side of the laser range scanner. We find out the strongest curvature along the top rim with a gauss curvature filter Porteous (1994).

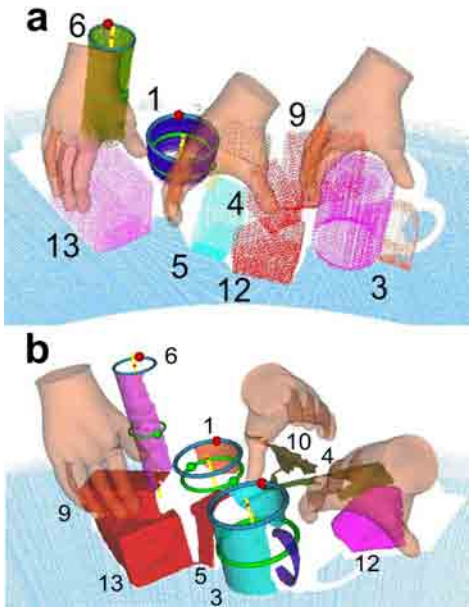

Fig. 6. **a** The green points illustrate the grasping points for rotation symmetrical objects. The red points illustrate alternative grasping points, thereby one grasping point is enough for an open object. For object no. 6 the scanner was not able to detect the top surface and so the algorithm find out that the object is open, which is in reality wrong. We calculate possible grasp poses for all other objects. **b** Calculated possible grasping points and poses to grasp the objects. The illustrated objects are very difficult to scan, due to shadows, reflections and absorptions.

To successfully grasp an object it is not always sufficient to locally find the best grasping pose. The algorithm should calculate an optimal grasping pose to realize a good grasp without collision as fast as possible. In general, conventional multidimensional "brut force" search methods are not practical to solve this problem. Li et al. (2007) show a practical shape matching algorithm, where a reduced number of 38 contact points are considered. Most shape matching algorithms need an optimization step through that the searched optimum can be efficiently computed.

At the beginning the internal center and the principal axis of the top surface are calculated with a transformation that fits a sphere inside, see Fig. 7 the blue top surfaces. After the transformation this sphere has an elliptical form in alignment of the top surface points, whereby also the principal axis is founded. The algorithm transforms the rotation axis of the gripper (defined by the fingertip of the thumb, the index finger and the last finger) along the principal axis of the top surface and the center (calculated with the fingertips) of the hand c_h will be translated to the center of the top surface c_{top}, whereby $c_h = c_{top}$ results. Thereby the hand will be rotated, so the normal vector of the hand aligns in reverse direction with the normal vector of the top surface. Afterwards the hand is shifted along the normal vectors up to a possible collision with the grasping object. Then the calculated grasp pose will be checked for a potential collision with the the remaining objects on the table. Thus we determined, if it is possible to grasp the object depending of the remaining objects, as illustrated in Fig. 6a.

5. Experiments and Results

In our work, we demonstrate that our grasp point detection algorithm for different objects shows promising results. We evaluated the detected grasping points and poses with the path planning tool from AMROSE. The object segmentation and grasp point detection for rotation symmetrical objects is performed by a PC with 3.2GHz dual core processor and takes about 20sec. and the calculation of possible grasp poses takes about 30sec., the calculation time depends on the number of the surrounding objects on the table. The algorithm is implemented in C++ using the Visualization Tool Kit (VTK)[6]. In testing of 5 different point clouds for every object in different combination with other objects from the 18 objects the algorithm shows positive results. A remaining problem is, that in some cases for shiny objects interesting parts of the objects are not visible for the laser range scanner and thus our algorithm is not able to calculate the correct grasping points or pose of the object.

[6]Open source software, http://public.kitware.com/vtk/

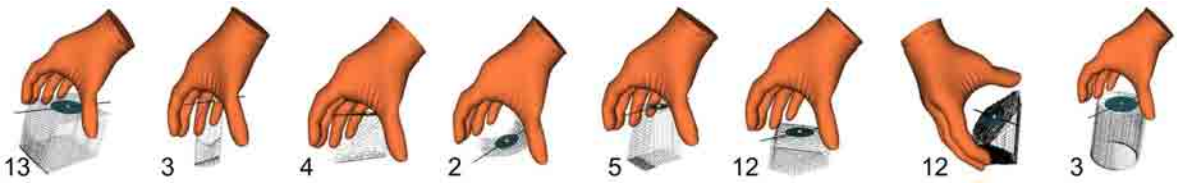

Fig. 7. Calculated possible hand poses to grasp the objects.

The quality of the point cloud is in some cases not good enough to guarantee a successful grasp, as illustrated in Fig. 6b. So the success of our grasping point algorithm depends on the ambient light, object surface properties, laser beam reflectance, absorption of the objects and vibrations. For object no. 2 the algorithm can not detect possible grasping points or a possible grasping pose, because of shadows of the laser range scanner with the coffee cup, as illustrated in Fig. 1. For all other objects we achieved an average grasp rate of more than 70%.

6. Conclusion and Future Work

The presented method for automatic grasping of unknown objects with a hand prosthesis, by incorporating a laser range scanner shows a high reliability. Thus the approach for object grasping is well suited for use in related applications under difficult conditions and can be applied to a reasonable set of objects. We presented a method for automatic reassembly of parts of 2.5D point clouds for rotation symmetrical objects using the top rim points. After the segmentation step we calculate grasping points of rotation symmetrical objects and grasping poses for arbitrary objects with the help of the top surfaces. In the near future we plan to use a deformable hand model to reduce the opening angle of the hand, so we can model the closing of a gripper in the collision detection step.

7. References

Arya S., Mount, D. M., Netanyahu, N. S., Silverman, R. 1998. An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions. *Journal of the ACM*, Vol. 45, No. 6, pp. 801-923.

Bentley, J. L. 1975. Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM*, Vol. 18, No. 19, pp. 509-517.

Belmonte, Ó, Remolar, I., Ribelles, J., Chover, M., Fernández, M. 2004. Efficiently using connectivity information between triangles in a mesh for real-time rendering. *Elsevier Science*, Vol. 20, No. 8, pp. 1263-1273.

Besl, P. J., McKay, H. D. 1992. A method for registration of 3-D shapes, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 14, No. 2, pp. 239-256.

Castiello, U. 2005. The neuroscience of grasping, *Nature Reviews Neuroscience*, Vol. 6, No. 9, pp. 726-736.

Ekvall, S., Kragic, D. 2007. Learning and Evaluation of the Approach Vector for Automatic Grasp Generation and Planning, *International Conference on Robotics and Automation*, pp. 4715-4720.

El-Khoury, S., Sahbani A., Perdereau, V. 2007. Learning the Natural Grasping Component of an Unknown Object, *International Conference on Intelligent Robots and Systems*, pp. 2957-2962.

Fischler, M. A., Bolles, R. C. 1981. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography, *Communications of the ACM*, Vol. 24, No. 6, pp. 381-395.

Goldfeder, C., Allen, P. K., Lackner, C., Pelossof, R. 2007. Grasp Planning via Decomposition Trees, *International Conference on Robotics and Automation*, pp. 4679-4684.

Li, Y., Fu, J. L., Pollard, N. S. 2007. Data-Driven Grasp Synthesis Using Shape Matching and Task-Based Pruning, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 13, No. 4, pp. 732-747.

Miller, A. T., Allen, P. K., 2004. GraspIt! A Versatile Simulator for Robotic Grasping, *IEEE Robotics & Automation Magazine*, Vol. 11, No. 4, pp. 110-112.

O'Rourke, J. 1998. Computational Geometry in C, *Univ. Press, Cambridge*, 2nd edition.

Porteous, I. R. 1994. Geometric Differentiation, *Univ. Press, Cambridge*.

Recatalà, G., Chinellato, E., Del Pobil, *Á*. P., Mezouar, Y., Martinet, P. 2008. Biologically-inspired 3D grasp synthesis based on visual exploration, *Autonomous Robots*, Vol. 25, No. 1-2, pp. 59-70.

Sanz, P. J., Iñesta, J. M., Del Pobil, *Á*. P. 1999. Planar Grasping Characterization Based on Curvature-Symmetry Fusion, *Applied Intelligence*, Vol. 10, No. 1, pp. 25-36.

Stansfield, S. A. 1991. Robotic grasping of unknown objects: a knowledge-based approach, *International Journal of Robotics Research*, Vol. 10, No. 4, pp. 314-326.

Schulz, S., Pylatiuk, C., Reischl, M., Martin, J., Mikut, R., Bretthauer, G. 2005. A hydraulically driven multifunctional prosthetic hand, *Robotica*, Cambridge University Press, Vol. 23, pp. 293-299.

Stiene, S., Lingemann, K., Nüchter, A., Hertzberg, J. 2006. Contour-based Object Detection in Range Images, *Third International Symposium on 3D Data Processing, Visualization, and Transmission*, pp. 168-175.

Point Cloud Segmentation Based on Radial Reflection

Mario Richtsfeld and Markus Vincze

Institute of Automation and Control
Vienna University of Technology
Gusshausstr. 27-29, Vienna, Austria
[rm, vm]@acin.tuwien.ac.at

Abstract. This paper introduces a novel 3D segmentation algorithm, which works directly on point clouds to address the problem of partitioning a 3D object into useful sub-parts. In the last few decades, many different algorithms have been proposed in this growing field, but most of them are only working on complete meshes. Experimental evaluations of a number of complex objects demonstrate the robustness and the efficiency of the proposed algorithm and the results prove that it compares well with a number of state-of-the-art 3D object segmentation algorithms.

Key words: point cloud segmentation, mesh segmentation, mesh decomposition, mesh generation, pose-invariant representation of point clouds.

1 Introduction

Cutting up an object into simpler sub parts has several benefits in modeling [11], robotics [14] or collision detection [18]. The presented work includes a new segmentation algorithm, based on radial reflection. Although the examples in this paper are related to applications in the area of computer graphics and robotics, the majority of the algorithms developed here can be applied with only trivial modifications to more complex shape matching problems.

1.1 Problem Statement and Contributions

Object segmentation and analysis, which can be interpreted as purely geometric sense are challenging problems in computer vision. An ideal shape descriptor should be able to find out the main features of an object and segment it into useful parts, which can be used for automatic processes such as matching, registration, feature extraction [13] or comparison of shapes. The object should be segmented into parts that correspond to relevant features and that are uniform with respect to some properties. This time different methods for mesh segmentation exist (e.g. Plumber [19], feature point and core extraction [15], Hierarchical Fitting Primitives (HFP) [3], spectral methods [25],...), but most of them are only able to work on a mesh and not a point cloud. This paper presents an

algorithm which works directly on point clouds and is invariant under rotation, translation and scaling.

1.2 Algorithm Overview

Fig. 1[1] gives an overview of our segmentation algorithm. The proposed segmentation algorithm is based on radial reflection. At the beginning the algorithm calculates the internal center and the radius of the bounding sphere by computing the smallest enclosing sphere of points [12], see Fig. 1d. Then, all points are radial reflected inside in the direction to the center. Thus all points which are inside on the original point cloud are farthest out after this step. The algorithm uses the reflected point cloud to calculate the convex hull [20], Fig. 1e (yellow hull), whereby all adhering parts on the core part will be automatically cut off. To realize a hole free segmentation of the core part all vertices of the convex hull are transformed in the direction to the center depending on the distances of the neighboring points [2], see Fig. 1e (red hull). Based on these vertices an inner convex hull is calculated. These inner convex hull surrounds the rest parts of the object. Then our algorithm automatically segments the 3D point cloud into a set of sub-parts by recursive flood-filling [9] based on the segmented core part, see Fig. 1f. To realize a pose invariant object segmentation our algorithm generates a 3D mesh based on the power crust algorithm [1], see Fig. 1b, and uses multi-dimensional scaling (MDS) to get a pose-invariant model representation, see Fig. 1c. Thereby every vertex on the pose-invariant model corresponds to a vertex of the mesh and every point of the original point cloud corresponds to a vertex of the mesh.

1.3 Related Work

Different methods to automatic 3D object segmentation into meaningful parts have been published in the last few years.

3D Model Segmentation: algorithms can be categorized into two main classes. The first class is developed for applications like reverse engineering of CAD models [5]. The second class tries to segment natural objects into meaningful parts. Most work on mesh segmentation is based on iterative clustering. [22] segmented models into meaningful pieces using k-means clustering. Based on this idea [16] developed a fuzzy clustering and minimal boundary cuts method to achieve smoother boundaries between clusters. Unsupervised clustering techniques like mean shift can also applied to mesh segmentation [21]. [10] published a method using skeletons to generate a hierarchical mesh decomposition. [15] published a mesh segmentation algorithm based on pose-invariant models and extraction of core part and feature points. The method is able to produce consistent results. An computation intensive method is used to find feature points, to limit the complexity and number of parts of models.

[1] All images are best viewed in color. The core part is in every case red colored.

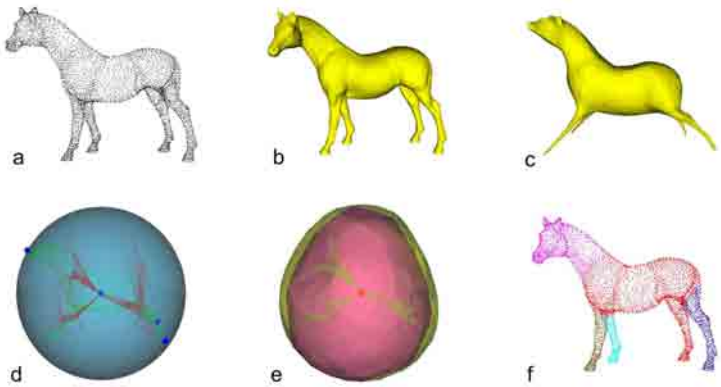

Fig. 1. Overview of our segmentation algorithm: **a** 3D point cloud (5360 points). **b** 3D mesh based on the power crust algorithm (58441 vertices). **c** Pose-invariant model representation based on multi-dimensional scaling (MDS) (58441 vertices). **d** Center and bounding sphere, the radial reflected point cloud (5360 points) is red colored, the original point cloud (5360 points) is green colored. The blue points (along the bounding sphere) correspond with the blue center of the radial reflected point cloud. **e** Outer convex hull (yellow), internal convex hull (red) to realize a hole free core part. **f** Segmented point cloud (2035 core points, 3275 rest points).

Mesh Generation: We decide to use the power crust algorithm for the surface reconstruction [1] of the 3D model, because this algorithm delivers very good results and is quite fast. It realizes a construction which takes a sample of points from the surface of a 3D object and produces a surface mesh and an approximate medial surface axis. The approach approximates the medial axis transform (MAT) of the object. Then it uses an inverse transform to produce the surface representation from the MAT.

Our Method: The basic idea is based on [15] work to extract the core part of the object with feature points and to use multi-dimensional scaling to realize a pose-invariant model representation. The difference to the existing core extraction algorithm is the radial reflection of the points in the direction to the center of the object and to calculate an internal convex hull to get a hole free core part, which is used to cut the 3D model. Additionally our algorithm works directly on point clouds, whereby no mesh generation is needed. The mesh generation with the power crust algorithm [1] is only needed to use multi-dimensional scaling (MDS) to get a pose-invariant model representation.

Pose-Invariant Mesh Representation: To realize a pose-invariant mesh representation multi-dimensional scaling (MDS) is used. MDS is a generic name for a family of algorithms that construct a configuration of points in a target metric space from information about inter-point distances (dissimilarities), measured in some other metric space [8]. In our experiments, dissimilarities are defined as geodesic distances δ_{ij} between all vertices v_i on the mesh $\mathcal{M}$ in a symmetrical dissimilarities matrix $\Delta = \mathcal{N} \times \mathcal{N}$ between N points on a Rieman-

nian manifold $\mathcal{S}$. Methods to calculate the dissimilarity matrix more effectively are based on the fast marching method on triangulated domains [17] or parametric fast marching [23]. We differentiate between metric and non-metric MDS (Shephard-Kruskal). Metric MDS preserves the intervals and the ratios between the dissimilarities and non-metric MDS only preserves the order of the dissimilarities. The goal is to minimize the embedding error, i.e. minimizing the sum of distances between the optimal scaled data $f(\delta_{ij})$ and the euclidean distances d_{ij}, where f is an optimal monotonic function (in order to obtain optimally scaled similarities). Thereby a stress function $\mathcal{F}_s$ will be used to measure the degree of correspondence of the distances between vertices. We use the scaled gradient-descent algorithm (SMACOF), as published by [8]. This algorithm is one of the most efficient at the moment and it allows real-time performance. Each vertex in MDS space corresponds to a vertex in euclidean space. The details of the SMACOF algorithm can be found in the above paper. In order to speed up the calculation time, the geodesic distances are calculated only on a reduced set of landmark points. Approximately the original points of the point cloud of the mesh vertices as landmark points has an optimal balance between accuracy of representation and time. Fig. 2 illustrates our segmentation results based on pose-invariant model representation.

Fig. 2. Pose-invariance: each model was segmented separately.

2 Point Cloud Segmentation

This section describes each stage of the proposed segmentation algorithm for point clouds.

2.1 Core Extraction

The presented method is based on the principle of radial reflection. At the beginning the internal center $\mathcal{C}$ is calculated by computing smallest enclosing sphere of points [12]. The bounding sphere is defined by the maximum distance $\mathcal{R}$ between the center $\mathcal{C}$ and all points p_i:

$$\mathcal{R} = max\|p_i - \mathcal{C}\| \tag{1}$$

Each point p_i of the point cloud with n points is radial reflected inwards in the direction to the calculated center $\mathcal{C}$, as illustrated in Fig. 1d and Fig. 1e.

$$p_m^{'} = \mathcal{C} + (\mathcal{R} - \parallel p_i - \mathcal{C} \parallel) \frac{(p_i - \mathcal{C})}{\parallel p_i - \mathcal{C} \parallel} \tag{2}$$

Thus all points which are farthest outside on the original point cloud are farthest in after this step, Fig. 1d. This way, the points of the core part reside on the outer convex hull $\mathcal{H}_{out}$ [20], whereby all adhering parts on the core part will be automatically cut off.

$$\mathcal{H}_{out} = ConvexHull \left(\bigcup_{i=0}^{n-1} p_{m_i}^{'} \right) \tag{3}$$

Every vertex v_m of the k vertices that reside on the outer convex hull $\mathcal{H}_{out}$ will be transformed in the direction to the center, depending on the distances of the neighboring points [2] with an offset o_{ff}. For that the algorithm calculates for each point of the original point cloud the distance to the nearest neighbor and then the minimum d_{min}, maximum d_{max} and average d_a of these distances. Then the algorithm finds out for every vertex v_m on the outer convex hull all neighboring points $p_m^{'}$ with the average distance d_a and calculates the offset o_{ff}, depending of the z point neighbors, see Equ. 5. This step is important to realize a hole free core part.

$$o_{ff} = \frac{\sum_{i=0}^{z-1} \mid p_{m_i}^{'} - v_m \mid}{z} \tag{4}$$

The offset o_{ff} was calculated with all z neighboring points of the transformed point cloud of the vertex v_m on the convex hull $\mathcal{H}_{out}$. With the calculated offset o_{ff} the algorithm need no more connectivity analysis to realize a hole free core part. All vertices on the outer convex hull $\mathcal{H}_{out}$ will be transformed with an offset for every vertex:

$$v_m^{'} = v_{m_i} - o_{ff} * \frac{(v_{m_i} - \mathcal{C})}{\parallel v_{m_i} - \mathcal{C} \parallel} \tag{5}$$

This k transformed vertices $v_m^{'}$ are used to calculate an inner convex hull $\mathcal{H}_{in}$, as illustrated in Fig. 1e (red convex hull):

$$\mathcal{H}_{in} = ConvexHull \left(\bigcup_{i=0}^{k-1} v_{m_i}^{'} \right) \tag{6}$$

The resulting inner convex hull $\mathcal{H}_{in}$ is used to cut the radial reflected point cloud into a core part and a rest part, as illustrated in Fig. 1f.

2.2 Cut Refinement

If the core part is found, all other segments of the point cloud are extracted by recursive flood-filling [9]. We define an object-part as a set of points, with distances between neighbors below a threshold d_{max}. We build a kd-tree [7] to find

neighbors and use the recursive flood-filling function [9] to identify connected point sets. d_{max} is the maximum distance between the neighboring points, calculated by nearest neighbor search [2]. This step segments the point cloud into different components. An additional cut refinement was not arranged, because the main goal is to find out the core part. It is possible to improve the segmentation results with the help of a substantially curvature-based filter [24], mean shift, gaussian curvature or a feature point based approach [15]. It is also possible to improve the segmentation results with the calculation of the normal vector for every point, by fitting planes in a defined area d_a. Thus the angle α between the regarded point i and the considered point w can be used as weighting factor wg, as illustrated in Fig. 3.

$$\cos\alpha = \frac{\boldsymbol{n_i} \bullet \boldsymbol{n_w}}{\|\boldsymbol{n_i}\|\|\boldsymbol{n_w}\|} \tag{7}$$

$$wg = 1 - |\cos(\alpha)| \tag{8}$$

To belong to a fracture of the object the distance d between a fracture element w and the considered point i must be smaller than the average distance with the weighting factor.

$$d = \sqrt{(x_i - x_w)^2 + (y_i - y_w)^2 + (z_i - z_w)^2} \tag{9}$$

$$d < d_a \cdot wg \tag{10}$$

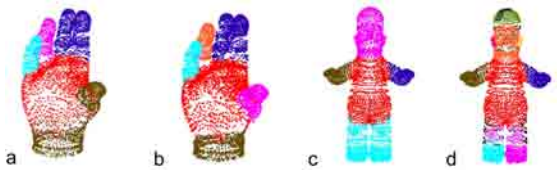

Fig. 3. Cut refinement: Improvement of the segmentation result by calculating an additional weighting factor. **a, c** Hand, Man: standard flood-filling. **b, d** Hand, Man: flood-filling with additional weighting function.

3 Results

We have created and collected at AIM@SHAPE repository[2] several challenging examples to test our segmentation algorithm, see Fig. 4. For similar segmentations of the same models in different poses, the segmentation based on pose-invariant models show almost best results. Our analysis shows that the position

[2] http://shapes.aim-at-shape.net/index.php

of the internal center of the models has a significant influence, as illustrated in Fig. 4g (dino) and h (elephant). It is important that the approximated center is inside the object. [15] presented another possible approximated method to calculate the internal center.

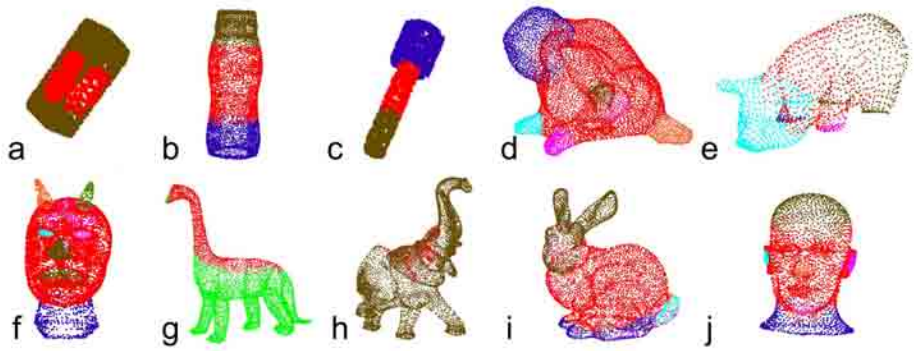

Fig. 4. Segmentation results: We analyzed different groups of models: **a** package, **b** coffee tin, **c** bolt, **d** frog, **e** pig, **f** oni, **g** dino, **h** elephant, **i** bunny, **j** mannequin.

Fig. 4 shows that the proposed algorithm is optimal to extract the core component and the surrounding parts.

Timing Results

On a 3.2GHz machine with 2GB RAM, we need on average 2-3min for generating a pose-invariant mesh with ∼3k points as landmarks, whereas the time expensive part is the calculation of the symmetrical dissimilarities matrix $\Delta = \mathcal{N} \times \mathcal{N}$ with all geodesic distances δ_{ij}. Core extraction needs less than 15sec., this includes also segmentation of the rest parts of the 3D model into sub-meshes based on recursive flood-filling. However the calculation time depends on the number of points of the 3D model. The algorithm is implemented in C++ using the Visualization Tool Kit (VTK)[3].

4 Conclusion

The proposed segmentation method represents a flexible and completely automatic way to segment a 3D object in a hierarchical manner, whereby the algorithm works directly on point clouds and shows high reliability. It is obvious from the results presented in this work that there exist no perfect segmentation algorithm. Each algorithm has his own benefits and drawbacks. Segmentation

[3] Open source software, http://public.kitware.com/vtk.

can neither be formalized nor measured mathematically, an empirical basis for research should be used. This can be realized by collecting hand-segmentations representing the ground-truth of various models, and comparing each algorithm results to it [5]. The pose-invariance is due to the use of MDS. We cut the object into sub-parts with an inner convex hull, which results from an outer convex calculated by radial reflection. This segmentation algorithm can be applied to a reasonable set of objects with different applications.

References

1. Amenta, N., Choi, S., Kolluri, R.: The power crust. Sixth ACM Symposium on Solid Modeling and Applications, pp. 249–260, (2001).
2. Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R., Wu, A.Y.: An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions. Journal of the ACM, vol. 45, no. 6, pp. 891-923, (1998).
3. Attene, M., Falcidieno, B., Spagnuolo, M.: Hierarchical Mesh Segmentation based on Fitting Primitives. The Visual Computer, vol. 22, no. 3, pp. 181–193, (2006).
4. Attene, M., Robbiano, F., Spagnuolo, F., Falcidieno, B.: Semantic Annotation of 3D Surface Meshes based on Feature Characterization. Lecture Notes in Computer Science (SAMT'07 Procs.), vol. 4816, pp. 126-139, (2007).
5. Attene, M., Katz, S., Mortara, M., Patane, G., Spagnuolo, M., Tal, A.: Mesh Segmentation - A Comparative Study. IEEE International Conference on Shape Modeling and Applications, SMI, pp. 7–18, (2006).
6. Biasotti, S.: Computational Topology methods for Shape Modelling Applications. PhD thesis, University of Genoa, Italy (2004).
7. Bentley, J.L.: Multidimensional Binary Search Trees Used for Associative Searching. Communications of the ACM, vol. 18, no. 19, pp. 509–517, (1975).
8. Bronstein, M.M., Bronstein, A.M., Kimmel, R., Yavneh, I.: Multigrid multidimensional scaling. Numerical Linear Algebra with Applications (NLAA), Special issue on multigrid methods, vol. 13, no. 2–3, pp. 149–171, (2006).
9. Burger, W., Burge, M.: Digital Image Processing - An Algorithmic Introduction Using Java. Springer, UK, London, 1st edition, (2007).
10. Cornea, N.D., Silver, D., Yuan, X., Balasubramanian, R.: Computing Hierarchical Curve-Skeletons of 3D Objects. The Visual Computer, vol. 21, no. 11, pp. 945-955, (2005).
11. Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S., Dobkin, D.: Modeling by example. ACM Transactions on Graphics, vol. 23, no. 3, pp. 652–663, (2004).
12. Gärtner, B.: Fast and Robust Smallest Enclosing Balls. Proceedings of 7th Annual European Symposium on Algorithms (ESA), Lecture Notes in Computer Science, Springer, pp. 325–338, (1999).
13. Gumhold, S., Wang, X., MacLeod, R.: Feature Extraction from Point Clouds. Proceedings of the 10th International Meshing Roundtable, pp. 293–305, (2001).
14. Huebner, K., Ruthotto, S., Kragic, D.: Minimum Volume Bounding Box Decomposition for Shape Approximation in Robot Grasping. IEEE International Conference on Robotics and Automation, ICRA, pp. 1628–1633, (2008).
15. Katz, S., Leifman, G., Tal, A.: Mesh segmentation using feature point and core extraction. The Visual Computer (Pacific Graphics), vol. 21, no. 8–10, pp. 649–658, (2005).
16. Katz, S., Tal, A.: Hierarchical mesh decomposition using fuzzy clustering and cuts. ACM Transactions on Graphics, vol. 22, no. 3, pp. 954–961, (2003).
17. Kimmel, R., Sethian, A.: Computing geodesic paths on manifolds. Proceedings of Natl. Acad. Sci., vol. 95, no. 15, pp. 8431–8435, (1998).
18. Li, X., Toon, T., Tan, T., Huang, Z.: Decomposing polygon meshes for interactive applications. Proceedings of the 2001 symposium on Interactive 3D graphics, pp. 35–42, (2001).
19. Mortara, M., Patanè, G., Spagnuolo, M., Falcidieno, B., Rossignac, J.: Plumber: a method for a multi-scale decomposition of 3D shapes into tubular primitives and bodies. Proceedings of the ninth ACM symposium on Solid modeling and applications, pp. 339–344, (2004).
20. O'Rourke, J.: Computational Geometry in C. Univ. Press, Cambridge, 2nd edition, 1998.
21. Shamir, A., Shapira, L., Cohen-Or, D., Goldenthal, R.: Geodesic mean shift. Proceedings of the 5th Korea-Israel Conference on Geometric Modeling and Computer Graphics, pp. 51–56, (2004).
22. Shlafman, S., Tal, A., Katz, S..: Metamorphosis of Polyhedral Surfaces using Decomposition. Computer Graphics Forum, vol.21, no. 3, pp. 219–229, (2002).
23. Spira, A., Kimmel, R.: An efficient solution to the eikonal equation on parametric manifolds. Interfaces and Free Boundaries, vol. 6, no. 3, pp. 315–327, (2004).
24. Trucco, E., Fisher, R.B.: Experiments in curvature-based segmentation of range data. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 17, no. 2, pp. 177–182, (2005).
25. Zhang, H., Kaick, O., Dyer, R.: Spectral Methods for Mesh Processing and Analysis. Proceedings of Eurographics 2007, pp. 1–22, (2007).

Boosted Edge Orientation Histograms for Grasping Point Detection

Abstract

In this paper, we describe a novel algorithm for the detection of grasping points in images of previously unseen objects. A basic building block of our approach is the use of a newly devised descriptor, representing semi-local grasping point shape by the use edge orientation histograms. Combined with boosting, our method learns discriminative grasp point models for new objects from a set of annotated real-world images. The method has been extensively evaluated on challenging images of real scenes, exhibiting largely varying characteristics concerning illumination conditions, scene complexity, and viewpoint. Our experiments show that the method, despite these variations, works in a stable manner and that its performance compares favorably to the state-of-the-art.

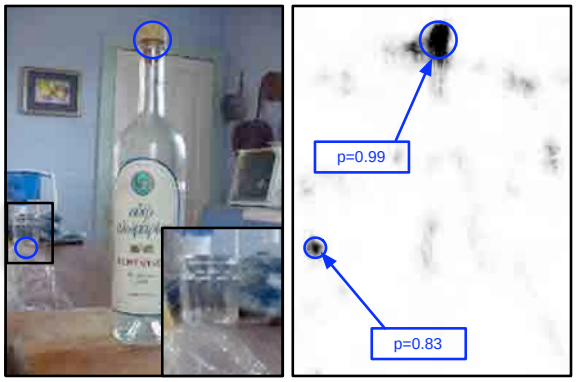

Figure 1. Detected grasp points (blue circles) and detector responses (right image). Note the zoom view of the bottle neck in the lower right of the left image.

1. Introduction

In this work, we focus on mining monocular vision input to detect potential points for robotic grasping of previously unseen objects. Grasping of novel objects using vision input is among the most challenging and difficult problem in robotic research. In the past, approaches either assumed a-priori knowledge about objects, or in case of previously unseen objects, relied on the extraction of sufficiently complete 3-d models eg. by using stereopsis. However, in realistic scenarios, where objects are occluded and only partially visible, or do not exhibit enough texture for stereo-based reconstruction, the latter are likely to fail.

Only recently, Saxena et al. [10, 11] presented a promising approach capable of grasping previously unseen objects (classes) purely based on vision. Their local, image-based, grasp point representation is learned from artificially created images of object examples and are separately searched for in pairs of stereo images. Then, only image locations with a high confidence of being a grasp point are triangulated to infer the 3D-position were the object can be grasped - thus avoiding the need of reconstructing the object's 3D shape. In [3] the authors find grasping points by describing the global object shape using shape context. However, as shape context is known to perform poorly in cluttered scenes [13] the work relies on high quality figure-ground segmentation, achieved with an active stereo setup, and knowledge about the workspace in which objects are placed.

Our approach is motivated by the existence of similar semi-local object parts in objects that themselves have rather dissimilar shapes. A typical example is the presence of handles in a large variety of objects ranging from scissors to jugs. In that sense, our method is similar in spirit to the one proposed in [10]. However, by encoding shape information of semi-local structures around grasp points, we arrive at discriminative representations which are able to ignore image clutter to a larger extent.

The contribution of our work is twofold: 1) We devised a novel image descriptor based on radially configured orientation-histograms. The descriptor is simple to implement, efficient, and can be easily extended to include a variety of cues such as color or texture.

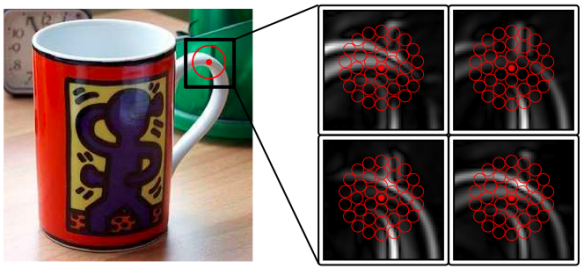

Figure 2. Illustration of the descriptor on image gradients. Probes (circles) are radially arranged arround the center (dot).

2) In contrast to preceding work utilizing artificially created data, we demonstrate that discriminative grasp point representations can be learned from images of real scenes.

2 Method

The presented approach consists of two stages: (1) Discriminative grasp points models are learned from annotated grasp points in real images. For this, a novel image descriptor is employed, which is able to efficiently encode the grasp points shape and its semi-local context. (2) In the detection phase, an input image is scanned densely over a range of scales using the learned model. On the resulting scale-space response maps, mean-shift mode seeking is employed to find the position and scale of potential grasp points. A typical result obtained with our method is depicted in Fig. 1.

2.1 Grasp Point Representation

Our representation of grasp points is an extension to Carmichael's [4] shape descriptor using a circular arrangement of edge probes. Each of these probes captures the density of the underlying edge image by weighted integration in a gaussian-shaped receptive field. Borrowing the idea from [14], we extend the descriptor to operate on channel images obtained from any orientation selective feature detector or filter. Specifically, having an input image $\mathbf{I}$, we compute a number C of *blurred orientation channels* $\mathbf{G}_o^{\sigma_p} = G_{\sigma_p} * \mathbf{C}_o, \;\; o = 1 \ldots C$, one for each discretisized orientation. Here, the channel image $\mathbf{C}_o$ is the component of the feature detector's output for direction o. G_{σ_p} denotes a Gaussian kernel with standard deviation σ_p and $*$ stands for convolution. Probe values at image location (x,y) for orientation o can be now efficiently obtained by simply accessing $\mathbf{G}_o^{\sigma_p}(x, y)$ which equals the pooled oriented response at that position. By stacking all channel-values for one probe location into a vector, a C-dimensional orientation histogram $\mathbf{p}$ is obtained.

Surrounding a probe at the query position, additional probes are located on K concentric circles with radii $r_k = k\sigma_p$, $k = 1..K$. Each circle is populated with an increasing number of $6k$ evenly spaced probes, see Fig. 2. For the choice of particular values for σ_p, K, and C we refer the reader to Sec.4.

2.2 Learning

Here, we utilize the GentleBoost algorithm to build a so-called strong classifier by iteratively combining the outputs of weak classifiers. The weak learners have the form of regression stumps [8] built from individual probe-based gradient histograms. At each boosting round, we run weighted Linear Discriminant Analysis (wLDA) [9] on the vectors formed by the bins of the orientation histograms for each probe position in the descriptor. The histogram-vectors are then projected onto the normal $\mathbf{w}$ of the discriminant and regression stumps are fitted to the resulting scalars.

After M rounds of boosting, the final classifier has the form of:

$$H = \sum_{m=1}^{M} a_m(\mathbf{w}_m^T \mathbf{p} > th_m) + b_m, \qquad (1)$$

where a_m, b_m, th_m are the parameters of the best weak classifier, and $\mathbf{w}_m$ is returned by wLDA - all at round m. $\mathbf{p}$ is the histogram described in Sec. 2.1.

At training time, positive examples are extracted by scaling the grasp regions in each image to the canonical scale and extracting the descriptor at the center of the annotated grasping region. To increase the number of positive samples, random variations of the grasp point examples, obtained by translation, re-scaling, and rotation in small ranges, are added [9]. To obtain negative examples, descriptors are extracted at random from the background of training images. For positions close to the grasping region the classifier is often not able to construct adequate discriminative models based on the randomly chosen negative examples. To counter this, we provide additionally negative examples near the grasping region [12]. In particular, we use positions located on circles centered at the grasp points, with a radius 1.5 times of that of the grasping region.

Once the initial detector is learned, one can bootstrap the gathering of further examples [5]. We scan the training set (see Sec. 2.3) for hard examples, i.e. misclassifications, and inject them into the training set for full retraining.

Figure 3. Examples from the dataset.

Figure 4. Grasping point (dots) and object annotation (bounding boxes).

2.3 Detection

Grasp points are found by a simple sliding window approach, as used in many object detection frameworks. We scan images in a range of predefined scales $\{s^k\}, k = 1 \dots K$. Specifically, for an image at scale s^k, one proceeds as follows: 1) Edges are computed and the components are distributed over C different channel images according to their orientation. The resulting maps are then smoothed by a Gaussian kernel to obtain blurred channel images $\mathbf{G}_o^{\sigma_p}$, see Sec. 2.1. 2) At each image position (x, y), the boosted classifier is evaluted on the descriptor values extracted by accessing the blurred orientation maps.

For each scale s^k and position (x, y) we obtain the classifiers confidence $H(x, y, s^k)$ which we convert to the posterior probabilities of a grasp point presence using the logistic transform proposed in [7]:

$$P(grasp\_point_{(x,y,s^k)}) = \frac{1}{1 + e^{-H(x,y,s^k)}} \qquad (2)$$

For a confidence map computed in such way, we refer the reader to Fig. 1. To find the set of grasp point detections, mean-shift mode estimation is adopted as described by Shotton et al. [12]. Location and scale of grasps point are given by detected modes; the detection confidence is obtained from the probability density estimate at the mode's location.

3. Experimental setup

We compiled a challenging dataset containing images of 3 object categories. The collection consists of 630 images, of which 210 show mugs, 210 bottles, and 210 Martini glasses. 30 of the mug images and 30 bottle images were taken from the database of Ferrari et al. [6], the remainder was found by Google image search. The images exhibit viewpoint changes, considerable background clutter and often more than one object instance and class are present, see Fig. 3. The number of annotated objects totaled 720.

Grasp points are represented by circular regions giving position and approximate scale of the relevant structure. Two grasp points were selected for each mug - one at the top of the handle and one in the middle. Martini glass grasp points are located at the upmost part of the shaft, bottles were annotated by the top of the neck. Overall, 956 grasp points have been annotated. In addition, each object instance is provided with a bounding box, designating the class of associated grasp points. Fig. 4 shows examples of annotated object instances and grasp points.

The dataset is split into two equally sized sets for training and testing. During training, images are rescaled such that each grasp point attains a canonical radius of 7 pixels before extracting the descriptor.

Test images were not rescaled and grasp points exhibit a scale range of roughly $3\times$ from smallest to largest. Given a minimum confidence threshold, detections are regarded as correct if the circular region of the inferred grasp point r_{inf} agrees sufficiently with the ground truth r_{gt}, checked by the symmetric overlap criterion $\frac{Area(r_{gt} \cap r_{inf})}{Area(r_{gt} \cup r_{inf})}) > 0.25$ similar to [1]. The overall performance of the detector is evaluated by means of precision-recall (PR) curves [2]

4. Results

In order to study the influence of histogram granularity and the particular choice of gradient computation, we compared Gaussian derivatives and the Sobel operator in two variants: Orientation estimation in the full 4-quadrant range, and ignoring the gradient direction by mapping its orientation in the range from 0 to π, i.e. bright to dark image transitions have the same orientation as dark to bright. Additionally, orientations were quantisized into $C = 4$ and $C = 8$ bin histograms (channel images). During all tests reported here, the remaining descriptor parameters (see Sec.2.1) were set to $\sigma_p = 5$ and $K = 5$, determined by cross-validation over the training set.

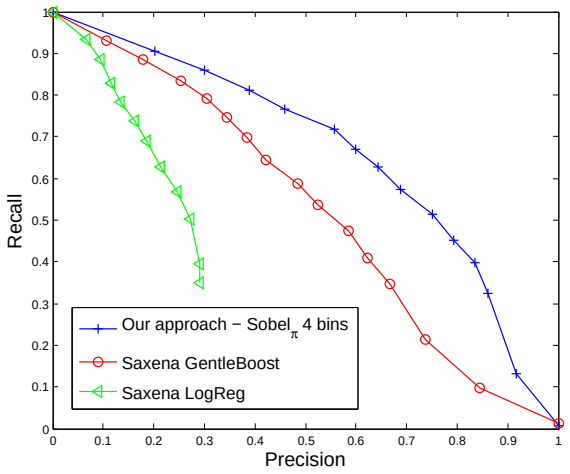

Figure 5. PR curves for our approach (blue crosses) versus Saxena's method (red circles and green triangles).

The results of these experiments are depicted in Fig. 5 (a), the corresponding area-under-curve values (PR-AUC) [12] are listed in Tab. **??**. Note that we omitted plots of $Sobel_\pi$ and $GaussD_\pi$ for 8 bins to reduce clutter. One can see that the Sobel filter consistantly outperforms Gaussian derivatives and that ignoring gradient polarity has the edge over its counterpart. This is in accordance with [5]. Overall, the best PR-AUC of 0.6656 was obtained by the polarity-ignoring Sobel operator $Sobel_\pi$ using orientation quantization into 4 channels. Fig. 6 shows some example detections taken from the test set.

In addition, we compared our method with the approach suggested in [10]. There, a descriptor based on Laws masks was used to encode texture over multiple scales. Since experiments revealed a poor performance (PR-AUC of 0.3460) of the proposed logistic regression algorithm, to have a fairer comparison we also present the improved results (PR-AUC of 0.5249) obtained using our GentleBoost-based learning framework. As can be seen from the precision-recall curves depicted in Fig. 5 (b), the proposed semi-local detector achieves significantly higher performance.

Finally, we tested our algorithm on images showing novel object classes not contained in the training set. The handles on the jar were detected as they resemble the mug handles. The same effect can be seen in the case of scissors. Furthermore, the detector is able to detect similarities which are not immediately apparent - the similarity of a flower stem to a martini glass shaft. These examples illustrate that the descriptor is capable of capturing the relevant shape similarity leading to meaningful detections of grasping regions.

Figure 6. Detection examples: Successful detections (red) and false positives (blue).

Figure 7. Meaningful detections (red) for classes not contained in the training set.

5. Conclusions

We presented a learning-based method for detecting grasp points in monocular images of newly seen objects. Extensive tests have shown that our approach based on boosted histograms outperforms the state-of-the-art. We were able to demonstrate that the approach is capable of capturing grasping relevant information, achieving promising results on familiarly shaped object from classes not contained in the training set.

Current work focuses on incorporating more monocular image cues as well as investigating extensions to automatically determine the blurring scale and aperture of the descriptors. Our next step will be the integration of our algorithm in a stereo-based setup similar to the one presented in [10].

References

[1] http://www.pascal-network.org/challenges/VOC.

[2] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *PAMI*, 26(11):1475–1490, 2004.

[3] J. Bohg and D. Kragic. Grasping familiar objects using shape context. In *14th International Conference on Advanced Robotics*, Munich, Germany, June 2009.

[4] O. Carmichael and M. Hebert. Shape-based recognition of wiry objects. *PAMI*, 26(12), 2004.

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages 886–893, 2005.

[6] V. Ferrari, F. Jurie, and C. Schmid. From images to shape models for object detection. *International Journal of Computer Vision*, 2009.

[7] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000, 1998.

[8] A. T. Kevin, K. P. Murphy, and W. T. Freeman. Sharing features: Efficient boosting procedures for multiclass object detection. In *CVPR*, pages 762–769, 2004.

[9] I. Laptev. Improving object detection with boosted histograms. *Image and Vision Computing*, (27):535–544, 2009.

[10] A. Saxena, J. Driemeyer, J. Kearns, and A. Y. Ng. Robotic grasping of novel objects. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1209–1216. 2007.

[11] A. Saxena, J. Driemeyer, and A. Y. Ng. Robotic grasping of novel objects using vision. *The Intl. Journal of Robotics Research*, 27(2):157–173, February 2008.

[12] J. Shotton, A. Blake, and R. Cipolla. Multiscale categorical object recognition using contour fragments. *T-PAMI*, 30(7):1270–1281, 2008.

[13] A. Thayananthan, B. Stenger, P. H. S. Torr, and R. Cipolla. Shape context and chamfer matching in cluttered scenes. In *CVPR*, pages 127–133, 2003.

[14] E. Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(1), 2009.

Combining Geometry and Local Appearance for Object Detection

Manuel Pascual García-Tubío
Automation and Control Institute
Vienna University of Technology
mp@acin.tuwien.ac.at

Horst Wildenauer
Pattern Recognition and Image Processing Group
Vienna University of Technology
wilde@prip.tuwien.ac.at

Abstract

In this paper we address the problem of object detection in cluttered scenes. Local image features and their spatial configuration act as representation of object classes which are learned in a discriminative fashion. Recent contributions in the area of object detection indicate the importance of using geometrical properties for representing object classes. Prompted by this, we devised an approach tailored to control the importance of the features and their spatial alignment. We quantitatively show that modeling the spatial distribution of local features and optimising the influence of both cues significantly boosts object detection performance.

1 Introduction

Object detection can be defined as finding instances of one or more object classes in images, by analysing characteristic image features and their spatial arrangement. Typically, an object class is represented by a set of spatially distributed features [3, 5, 11, 13] (e.g local image descriptors or edge maps associated with the object centroid) combined into a classifier that describes discriminative properties of each feature.

Recent approaches have shown remarkable results on a set of challenging databases. Their authors commonly emphasize the importance of encoding geometrical properties along with local features into an object model. For example, Opelt et al. [12] constrain the position of individual shape parts with relation to the object centroid. Shotton et al. [13] and Ferrari [6] also relate local shape based features to the object centroid and use this information to integrate the votes from individual parts. All these methods have in common that geometric information of individual parts plays an important role in the recognition process, though the use of particular features as well as the encoding and learning of geometric part-relations differs significantly.

Only recently, Stark and Schiele [14] attempted to analyze the gain in classification performance attained by the incorporation of information about geometric layout. The proposed localized Bag-of-Words approach enables for gradual addition of location information to object representations, thereby allowing for a quantification of the impact on classification accuracy. It was shown, that location information indeed boosts overall accuracy, most significantly when combined with less discriminative shape features.

Inspired by these findings, we propose a method that explicitly combines features with their geometrical relationships and allows to control the influence of both entities on the final result. In our case the similarity between the object model and the detected object instance is estimated from alignment as well as similarity of corresponding features. This solution produces a classifier that is sensitive to both and more importantly allows us to investigate the importance of features as well as geometry on the accuracy of object detection.

2 Method overview

The corpus of the method is divided in two main parts, namely the training of the object class model (Sec. 3) and the subsequent object detection procedure (Sec. 4). The process of the object model extraction is performed for each class separately and is divided into the following steps: 1) Image features along with their scale normalized positions are extracted from a set of training examples (Sec. 3.1). Examples of objects are obtained from rectangular regions in training images enclosed by the predefined bounding boxes. 2) During codebook generation (Sec. 3.2), features computed from training examples are quantized into clusters representing distinctive appearance and location inside the canonical bounding box. 3) A Gentle-Boost classifier is used to learn the object model, fed with the collection of positive and negative samples that result from matches between codebook and instances from the validation set

Preprint submitted to 20th International Conference on Pattern Recognition.
Received January 25, 2010.

(Sec. 3.3). The matching between features is tailored to optimize the balance between the feature description and the spatial information of those.

Once the object model is built, detection is performed with a sliding window approach. Classification votes are cast along the scale space and object hypothesis are aligned with the classification maxima. Multiple detections of the same object instance are eliminated by non-maxima suppression.

3 Object Representation

3.1 Feature extraction

We use SIFT [9] features extracted around symmetry based interest points [15]. We tend to use well understood features and focus on improving object detection by applying geometrical properties. In preliminary experiments we found that symmetry points are more repeatable than the typical DoG based interest points.

3.2 Codebook

Before we compute the appearance codebook for local image patches, feature sets from positive training instances are spatially normalized to the canonical bounding box. Then inspired by [6], each bounding box is subdivided into a regular 11×11 grid of rectangular cells from which all possible 3×3 blocks of adjacent cells are formed, see Fig. 1. For each of the resulting 121 blocks, features falling into it are clustered together using K-means. The complete codebook is obtained by concatenating all clusters along the grid. Due to the overlap of blocks, features can be shared between several clusters. For all our experiments K was experimentally set to 10, resulting in a total number of 1210 codebook entries.

3.3 Classifier Training

To learn the visual object class representation, we employ the discriminative machine learning technique of boosting. To be more specific, we use Gentle-Boost [7] with weak classifiers in the form of regression stumps as proposed by [16].

The Gentle-Boost classifier has a typical form of linear combination of M weak classifiers:

$$H(x,s) = \sum_{M} a_m \Big(\epsilon_m > \theta_m\Big) + b_m \qquad (1)$$

where a_m, b_m and θ_m are the parameters of the best weak classifier, and ϵ_m is the similarity between the m-th codebook entry and the corresponding feature in the analysed image or training example.

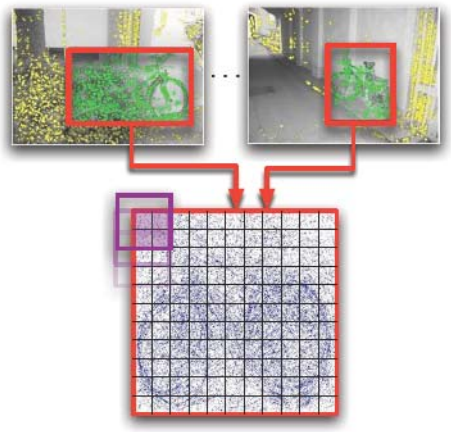

Figure 1: Illustration of spatial separation of training features for codebook generation. Features falling in the same 3x3 region (solid outline in violet) are clustered together. Notice the overlapping of the resulting regions.

In our approach, this similarity is defined by a combination of similarity of the feature vectors and the spatial alignment of their location in the image window. More formally:

$$\epsilon_m \equiv \epsilon_{ia} = \exp\left(-\frac{d_s(i,a)^2}{2\sigma_s^2}\right) \exp\left(-\frac{d_f(i,a)^2}{2\sigma_f^2}\right) \qquad (2)$$

where $d_s(i,a)$ is the spatial distance between the i-th codebook entry and the a-th object instance feature (normalized to the canonical bounding box extents). $d_f(i,a)$ is the euclidean distance between the feature vectors. Parameters $\sigma_{s,f}$ allow to regulate the amount of penalty that is associated with the feature misalignment and dissimilarity respectively:

- $\sigma_f \to 0 \ \wedge \ \sigma_s \to 0$ is a strict penalizer that only accepts perfectly aligned and identical features.
- $\sigma_f \gg 1 \ \wedge \ \sigma_s \gg 1$ produces almost a linear similarity function.
- $\sigma_f \gg 1 \wedge \sigma_s \to 0$ is forgiving for feature dissimilarity but requires a good spatial alignment which would agree with the notion of weak local features (both shape and appearance) and the importance of geometry in the object description.

The classifier training is performed on a set of positive (objects) and negative (background) examples extracted from the training data sets. Each example provides information about similarity and spatial alignment

Preprint submitted to 20th International Conference on Pattern Recognition.
Received January 25, 2010.

between codebook entries and the corresponding object features. Negative examples are randomly extracted from images not showing any instance of the specific class as well as from background of positive images. These additional negative examples are sampled from image regions with 30% overlap with the object bounding box (cf. [13]).

Due to the limited number of training images containing positive examples we artificially create additional examples by displacing, and stretching the bounding box window within small, predefined ranges to account for possible transformations not represented by the training data set [8]. Overall, we sampled positive and negative examples in a ratio of 1:10. Depending on the training data set the learner typically converged after selecting around 100 weak classifiers.

4 Object Detection

The presented method performs multi-scale object detection based on the sliding window approach [17, 4]. The classification scores $H(x, s)$ are computed across the scale-space and stored in a discrete 4D map that covers image locations and detection window sizes representing anisotropic scale. The number of scales to be searched is estimated from the training data set and represents frequently occurring combinations of window size and ratio.

Below we give a detailed description of the detection process computing classification score $H(x, s)$ for different locations and sizes of the detection window:

1. Establish codebook feature correspondences – each of the codebook entries is associated with the most similar feature inside the detection window using (2), that measures spatial alignment and feature similarity. Note that some of the image features inside the detection window may be left unassigned.

2. Compute classification score $H(x, s)$ (1) – object hypotheses are associated with the local classification maxima (in the scale-space domain) that are higher than the predefined confidence threshold Θ.

3. Non-maxima suppression – the corresponding overlap between remaining maxima is computed. If bounding boxes associated with two proximal maxima exhibit more than 50% symmetric area overlap (as in [1]), the one with smaller confidence is removed.

Database	Training	Test	AUC Our	AUC [13]
Horses	100	228	0.92	0.89
Faces	100	217	0.95	0.98
Aeroplanes	100	400	0.81	0.93
Motorbikes	100	400	0.97	1.0
Bikes (side)	72	72	0.72	0.69

Table 1: PR-AUC values for the performance of our method.

5 Evaluation

Our approach was tested on five databases widely used to assess the performance of object detectors, namely the Weizmann Horses [2], the Caltech Faces, Aeroplanes and Motorbikes [5], and the TU-Graz Bicycles Database [10]. For each dataset, we select a relatively small number of images ($< 25\%$) for codebook extraction and the remainder of the training set to serve for learning the boosted classifier. The full training set does not exceed 50% of the whole database in each case.

Figure 2 shows the performance of our algorithm utilising precision-recall curves (PR). Table 1 gives corresponding PR-AUC values in comparison to Shotton et al. [13]. Shotton operates on boundary-based features extracted from edges, which were tuned for detection of objects best characterized by shape. Despite using local appearance features, our method is able to compete and in some cases outperform [13], especially on the challenging databases containing horses and bicycles. Note that Shotton separated bicycles into side, frontal, and back views; we follow this and report only the side view results here.

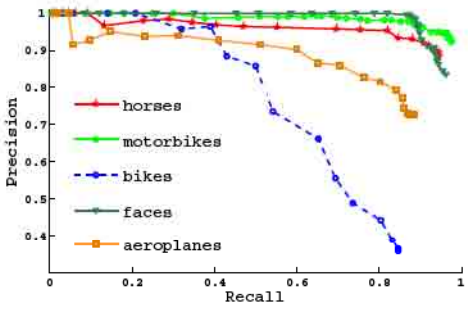

Figure 2: Precision-Recall curves for evaluated databases.

Figure 3 illustrates the dependency of PR-AUC for the bicycles, and faces databases on parameters σ_s and

Preprint submitted to 20th International Conference on Pattern Recognition.
Received January 25, 2010.

σ_f, which regulate the influence of spatial alignment and similarity between features (respectively) in the codebook and the detected object instance as discussed in Section 3.3. This graph clearly shows that once the influence of feature similarity is made dominant over spatial alignment ($\sigma_s \gg \sigma_f$) the detection performance deteriorates. The same is evident for the opposite case ($\sigma_f \gg \sigma_s$) although results are better, suggesting that geometrical properties play a more important role than the raw features similarity.

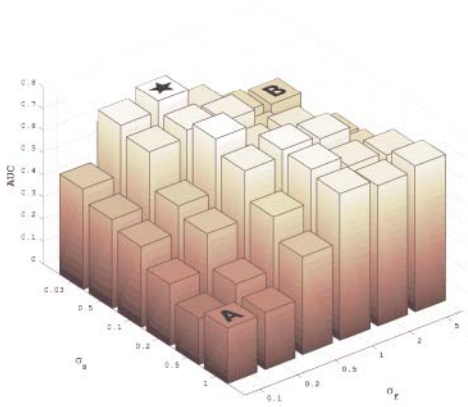

Figure 3: PR-AUC as a function of σ_f and σ_s for the bicycles database. Stars indicate a combination of σ_f and σ_s producing best scores. Letters **A** and **B** represent extreme cases of $\sigma_s \gg \sigma_f$ and $\sigma_f \gg \sigma_s$ respectively.

6 Conclusions

We have presented an object detection approach that combines the spatial configuration of local image features with the inherent information carried by these features. The method is tailored to carry out the evaluation that quantitatively shows the influence of geometry on the classification accuracy. We show that, despite the simplicity of the approach, excellent results are obtained by properly balancing geometry and feature similarity.

Our method compares favourably to the boundary-based approach of Shotton et al. which is another indication that not the type of features but their spatial distribution plays a crucial role in the ability to detect complex objects. The difference between the reported detection performance for the bicycle database and other databases points us onto the problem of intra-class variability which will be the primary topic of our future research.

References

[1] http://www.pascal-network.org/challenges/VOC.

[2] E. Borenstein, E. Sharon, and S. Ullman. Combining top-down and bottom-up segmentation. In *CVPRW '04: Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04) Volume 4*, page 46, Washington, DC, USA, 2004. IEEE Computer Society.

[3] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *In Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:886–893, 2005.

[5] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 264–271, June 2003.

[6] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *T-PAMI*, 30(1), 2008.

[7] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000, 1998.

[8] I. Laptev. Improving object detection with boosted histograms. *Image and Vision Computing*, 2008.

[9] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[10] A. Opelt, M. Fussenegger, A. Pinz, and P. Auer. Weak hypotheses and boosting for generic object detection and recognition. In T. Pajdla and J. Matas, editors, *ECCV (2)*, volume 3022 of *Lecture Notes in Computer Science*, pages 71–84. Springer, 2004.

[11] A. Opelt, A. Pinz, and A. Zisserman. A boundary-fragment-model for object detection. In *ECCV*, pages 575–588, 2006.

[12] A. Opelt, A. Pinz, and A. Zisserman. Incremental learning of object detectors using a visual shape alphabet. In *CVPR (1)*, pages 3–10. IEEE Computer Society, 2006.

[13] J. Shotton, A. Blake, and R. Cipolla. Multiscale categorical object recognition using contour fragments. *T-PAMI*, 30(7):1270–1281, 2008.

[14] M. Stark and B. Schiele. How good are local features for classes of geometric objects. In *ICCV*, pages 1–8, 2007.

[15] L. Szumilas, R. Donner, G. Langs, and A. Hanbury. Local structure detection with orientation-invariant radial configuration. In *CVPR*, 2007.

[16] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing features: Efficient boosting procedures for multiclass object detection. In *In CVPR*, pages 762–769, 2004.

[17] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 2004.

Preprint submitted to 20th International Conference on Pattern Recognition.
Received January 25, 2010.

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

Oliver Ruepp and Darius Burschka
Institut für Informatik
Technische Universität München
Boltzmannstrasse 3, 85748 Garching
Email: {ruepp, burschka}@in.tum.de

***Abstract*— We present a method for vision-based recovery of three-dimensional structures through simultaneous model reconstruction and validation 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 minimal texture due to partial homogenity of the surfaces. Additionally, we describe an efficient method facilitating Levenberg-Marquardt optimization of complex compositional functions.**

I. 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 [1] features, Good Features to Track [2], edges 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. [3] 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 to achieving this goal, we developed the method presented in this paper, which allows simultaneous model reconstruction and validation 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.

Our method belongs to the family of bundle-adjustment techniques. An in-depth survey of the original bundle-adjustment method is given in the book by Hartley and Zisserman [4]. The paper by Triggs et al. [5] provides a good overview of bundle adjustment variants and related methods.

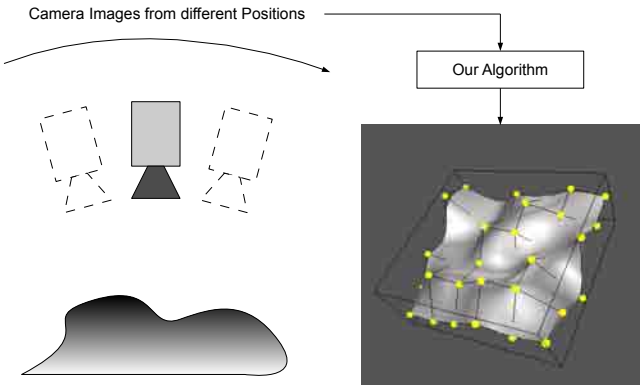

Fig. 1. Schematic overview of the problem addressed by our algorithm.

There is also a more recent paper evaluating the status of real-time bundle adjustment methods [6].

Since we are working only with intensity values, we also evaluated other approaches for intensity based tracking algorithms. Tracking and matching of fixed point clouds has been investigated by Sepp et al. [7], and is also related to the problem considered herein. The tracking methodology is very similar to that used herein, but we are using a monocular camera instead and determining the parameters of a surface model during the process.

A number of offline methods for model-based bundle-adjustment have been described with applications to face modeling [8, 9]. 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.

II. Problem Statement

We are interested in recovering and validating the structure of a 3D object on-line from a stream of monocular camera images. The object we are interested in must be static, and it must be possible to represent the object by means of a parametric surface model. 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. The concept is visualized in Figure 1.

The basic idea is as follows: In traditional bundle adjustment, coordinates of 3D points that are associated with features are recovered from a set of 2D feature position

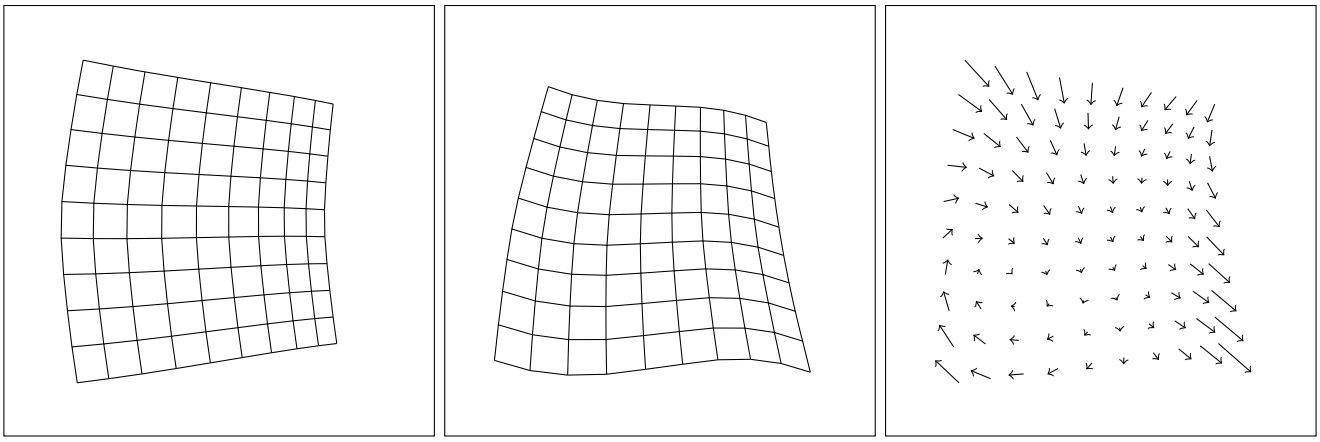

Fig. 2. Left, middle: Surface under two different camera positions. Right: Warping of surface coordinates from left to right image.

measurements. This approach will obviously work only if a feature detecting scheme can be used at all. In our case, we do not assume that robust feature extraction is possible, and thus we do not work with 2D positions, but with image intensities.

III. Approach

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 through use of reliable feature detection methods. Unfortunately, this assumption does not hold in the situation described above: We assume that the scene we are looking at does not exhibit a lot of structure, and we expect it to be very difficult to reliably detect and track features. Using a point cloud model would thus be problematic, since the position of a point can only be determined if the point can be identified reliably, which is not the case.

A better suited model would be a parametric surface of type $S : \mathbb{R}^k \times \mathbb{R}^2 \rightarrow \mathbb{R}^3$. 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. This loss of generality is a compromise that is necessary in the difficult situation of 3D reconstruction in scenes with low structure.

Inspired by the method of Ramey et al. [3], we do not directly model the scene as a 3D surface. Instead, we choose to 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 $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ 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.

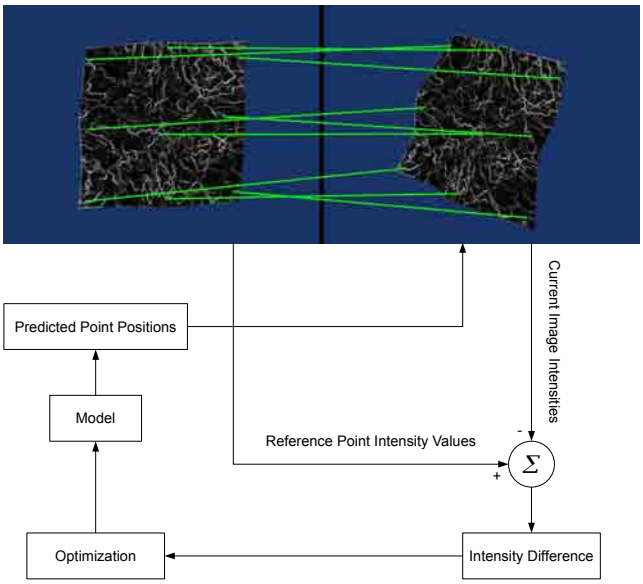

Fig. 3. Overall structure of the algorithm.

The idea of our approach is now basically the same as in traditional bundle adjustment: Using a nonlinear optimization technique, we are able to compute parameters for the warping function that best explain the observations. Thus, we are able to determine a good approximation of the warping function itself. Figure 3 shows the concept.

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 picked according to a weak criterium that will be described later. These pixels are selected 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. In the following, we will derive the image warping function step by step. Before we start with the mathematical part, 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 $\mathbb{R}^k \times \mathbb{R}^2 \rightarrow \mathbb{R}$ 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 $\mathbf{q}_n \in \mathbb{R}^4$.
- $T(\mathbf{t}, \mathbf{q}, \mathbf{p}) : \mathbb{R}^3 \times \mathbb{R}^4 \times \mathbb{R}^3 \rightarrow \mathbb{R}^3$ 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

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

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 be parameterized by depth λ as

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

Then, the full 3D model surface is

$$r_{u,v}(S_{\mathbf{d}}(u, v)) = S_{\mathbf{d}}(u, v) \cdot \left(\frac{u - c_x}{f_x}, \frac{v - c_y}{f_y}, 1 \right)^T$$

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 following relationship to hold for all model surface coordinates (u, v):

$$I_n(\pi(T(\mathbf{p}_n, r_{u,v}(S_{\mathbf{d}}(u, v))) = I_0(u, v)$$

Of course, we do not have a model, and we do not know the camera position, but we want to determine them. Thus, we assume that the correct camera position and the correct model parameters together minimize the absolute difference, or equally the squared difference in intensity values:

$$(I_n(\pi(T(\mathbf{p}_n, r_{u,v}(S_{\mathbf{d}}(u, v))) - I_0(u, v))^2$$

Obviously, it will be impossible to determine camera and model parameters by comparing intensity values of only one point seen in two images, we need to take more points into account. However, it is also, due to computational complexity, not advisable to compare intensities of all pixels of the model surface. We will make a compromise and try to find parameters that minimize the intensity differences of the m reference points. The corresponding cost function $c(\mathbf{d}, \mathbf{p}_n)$ can be defined as

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

Defining a vector-valued function of image intensities, the cost function can be written in a more concise way. If we define

$$c(\mathbf{d}, \mathbf{p}_n) = \begin{pmatrix} I_n(\pi(T(\mathbf{p}_n, r_{u_1,v_1}(S_{\mathbf{d}}(u_1, v_1))) - I_0(u_1, v_1) \\ I_n(\pi(T(\mathbf{p}_n, r_{u_2,v_2}(S_{\mathbf{d}}(u_2, v_2))) - I_0(u_2, v_2) \\ \vdots \\ I_n(\pi(T(\mathbf{p}_n, r_{u_m,v_m}(S_{\mathbf{d}}(u_m, v_m))) - I_0(u_m, v_m) \end{pmatrix}$$

then $c(\mathbf{d}, \mathbf{p}_n)^T c(\mathbf{d}, \mathbf{p}_n)$ is the value of the summed squared intensity differences, and hence equivalent to the cost function specified above. 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.

But, there are two more minor issues that we need to take care of: Reconstruction of three-dimensional structures from monocular image sequences is always only possible up to scale, but we want at least to keep the scale constant. Furthermore, a quaternion describing a rotation must have unit length, and we need to enforce that somehow.

Keeping the scale constant over the image sequence can be achieved by simply adding a constraint that fixes the depth of one of the reference points to some fixed value. Let that depth value be denoted by $s \in \mathbb{R}$, and assume that, without loss of generality, we fix the depth of the first reference point. Then, the additional constraint to add to above constraints would be $S_{\mathbf{d}}(u_1, v_1) = c$. In our optimization formulation, we would then need to minimize the squared difference $(S_{\mathbf{d}}(u_1, v_1) - c)^2$.

Similarily, for enforcing unit length of the rotation quaternion, we add the constraint $|\mathbf{q}_t| = 1$, or the constraint of minimizing $(|\mathbf{q}_t| - 1)^2$. Overall, the total objective function to be optimized can now be stated as

$$o(\mathbf{d}, \mathbf{p}_n) = \begin{pmatrix} c(\mathbf{d}, \mathbf{p}_n) \\ S_{\mathbf{d}}(u_1, v_1) - c \\ |\mathbf{q}_n| - 1 \end{pmatrix}$$

By optimizing camera parameters and depth map parameters according to above objective function, we can, for each image, determine a 3D model that best explains the image measurement.

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 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. The more reference points are used, the better the problem will be constrained.

IV. Efficient Optimization

To actually recover the model parameters from the scene, we need some method to minimize the cost function described above. Typically, the Levenberg-Marquardt method [10, 11, 12] is applied to such problems. That method is useful for minimizing nonlinear functions, and it basically works through solving a linear system, the so-called *augmented normal equations*. The basic idea is as follows: If $f : \mathbb{R}^n \to \mathbb{R}$ is the function to be minimized, ∇f is its gradient and H_f its Hessian matrix, then the function can be approximated by means of the Taylor expansion around the current parameter vector p through

$$f(p + \delta) \approx f(p) + \delta^T \nabla f(p) + \frac{1}{2} \delta^T H_f \delta.$$

We then proceed by minimizing the approximated term through differentiating w.r.t. δ and setting the result equal to 0:

$$\nabla f(p) + H_f \delta = 0 \Leftrightarrow H_f \delta = -\nabla f(p)$$

In our case, we have $f(p) = o(p)^T o(p)$. The equation stated above then becomes

$$H_{o(p)^T o(p)} \delta = -\nabla (o(p)^T o(p))$$

By approximating the Hessian $H_{o(p)^T o(p)}$ with $J_o J_o^T$ and using basic calculus and the chain rule, this can finally be rewritten as

$$J_o J_o^T \delta = -J_o^T o(p),$$

where J_o denotes the Jacobian of the objective function defined above. This is the regular system of normal equations. The so-called augmented normal equations are then obtained by adding a so-called damping term λI:

$$(J_o J_o^T + \lambda I)\delta = -J_o^T o(p).$$

That term basically allows the method to interpolate between gradient descent steps and Gauss-Newton steps. This equation system is solved for δ several times until convergence. A detailed description and analysis of the method is provided in Hartley and Zisserman's book [4]. We will from now on focus on the key part of the algorithm, which is efficient and accurate computation of the Jacobian J_o.

For computing the Jacobian, we took three different approaches into consideration: Numerical approximation using finite differences, code generation using symbolic computation, and Automatic Differentiation [13, 14]. Approximation using finite differences has been shown to be both inefficient and inaccurate as compared to the other methods, which disqualified the method for our purposes.

Symbolic differentiation works through specifying the function of interest in a Computer Algebra System, which will then be able to compute the symbolic Jacobian of that function. After this step has been performed, efficient programming language code can be generated from the symbolic Jacobian. Still, purely symbolic differentiation is problematic for large problems, because the symbolic computation alone can be very slow. Even for rather simple problems, the symbolic computation might take up to days, which is not acceptable.

Automatic Differentiation is a method to numerically evaluate the derivative of a function specified by a computer program. It treats a computer program that implements a vector-valued function $y = F(x)$ as a composition of a sequence of elementary functions. Each one of those functions can be trivially differentiated using a look-up table. The derivative of the composition can also be evaluated easily by applying the chain rule from derivative calculus. This process yields highly accurate derivatives. Actually, a symbolic computation step is inherently used in this method as well, but because it is applied only at the most basic level, the computational problems of symbolic computation are avoided.

Since it is fast and accurate, the method of choice would have been Automatic Differentiation. There was only one problem: The implementation of the B-Spline surface functions were not done by us, but were part of third-party libraries[1]. Compiling this library with Automatic Differentiation

[1] http://www.sintef.no/Projectweb/Geometry-Toolkits/SISL/

support would have required us to make really fundamental changes to it, which is something we wanted to avoid. However, the library already provides functions for computing the required derivatives, so we wanted to exploit that.

Instead of directly applying Automatic Differentiation, we hence adopted a slightly different idea. Basically, the Jacobian of a compositional function $f = f_1 \circ f_2 \circ \ldots \circ f_n$ can be computed as matrix chain product of the Jacobians of the individual functions:

$$J_f = J_{f_1} \cdot J_{f_2} \cdot \ldots \cdot J_{f_n}$$

This idea can also be applied to our cost function, since it can be interpreted as a composition of several functions. The separate functions that we have used to define it are quite simple and computation of their Jacobians is straightforward. At first sight, one might think that this way of computing J_f is not very efficient, since matrix multiplications are usually very costly. But, looking at the structure of the Jacobians of the used functions, we see that all of the Jacobians exhibit a high degree of sparsity. Indeed, it can easily be shown that the number of nonzero entries is actually linear in the number of reference points chosen.

Since we are dealing with a matrix chain product, and matrix products are associative, it is also important to take into account the bracketing, i.e., the order of evaluation of multiplications for computing the overall matrix product. It is well-known that for dense matrices, the bracketing can make a tremendous difference in computation time, and there exists an algorithm that efficiently computes an optimal matrix bracketing based on dynamic programming [15, 16].

In our case however, we are not dealing with dense matrices, but with sparse matrices. Fortunately, the approach developed for optimal bracketing of a dense matrix chain product can easily be modified to work with sparse matrices as well.

The key function for determining the optimal bracketing for dense matrix chain product is the computation of the cost of multiplying two matrices, where only elementary multiplications are counted. Let A_1 and A_2 be two matrices with dimensions of $n_1 \times n_2$ and $n_2 \times n_3$, respectively. Then the total cost of computing the matrix product would be $n_1 n_2 n_3$.

For sparse matrices, the multiplication cost depends on the actual sparsity structure of the involved matrices. Fortunately, all of the functions used in the objective function have a static Jacobian sparsity structure that does not depend on the parameters. Thus, it suffices to compute the bracketing only once, at the beginning of the algorithm. Consider the following example with small matrices A, B that have sparsity structures S_A, S_B. You will see that the product of the sparsity structure matrices contains, for each entry of the product matrix, the number of multiplications needed to compute that entry:

$$A = \begin{pmatrix} 1 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}, B = \begin{pmatrix} 1 & 3 \\ 0 & 1 \\ 1 & 0 \end{pmatrix},$$

$$S_A = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, S_B = \begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}; S_A \cdot S_B = \begin{pmatrix} 1 & 2 \\ 1 & 0 \end{pmatrix}$$

Then the overall multiplication cost is obviously the sum of entries of $S_A \cdot S_B$. Thus, if we replace the cost measurement used in the dense chained matrix multiplication algorithm with this cost function, we will retrieve an algorithm that computes the optimal bracketing for sparse matrix multiplication. Note that since the cost computation now relies on the sparsity structure of the involved matrices, it is required that the sparsity structures of all subchains are computed. This in turn means that the preparation step is quite costly, but it pays off later on. Another option for cost measurement would be to use a heuristic, such as the one developed by Cohen [17].

It should be noted that our approach deals with the same structural form of matrices as the method developed by Griewank and Naumann [18]. Their method has been shown to be very efficient for this type of problem. Roughly speaking, they are using Automatic Differentiation for computing the individual Jacobians of small sub-functions, then applying sparse matrix chain multiplication with optimal bracketing to compute the overall Jacobian. Since the problem of optimal Jacobian accumulation (computing the Jacobian with minimal computational expense) has been shown to be NP-complete by Naumann [19], this is not the optimally efficient solution, but can be interpreted as a heuristic approach to solving the problem.

After the computation of the Jacobian is finished, the augmented normal equations are solved by computing $(JJ^T + \lambda I)$ and using a sparse LDL^T Cholesky transformation on the resulting matrix. Apart from that, the Levenberg-Marquardt method is used in its standard form.

V. 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 directly into our optimization approach. However, the idea has also been used by Lucas and Kanade [20] for their optical flow algorithm, which is well-established and implementations of which are readily available.

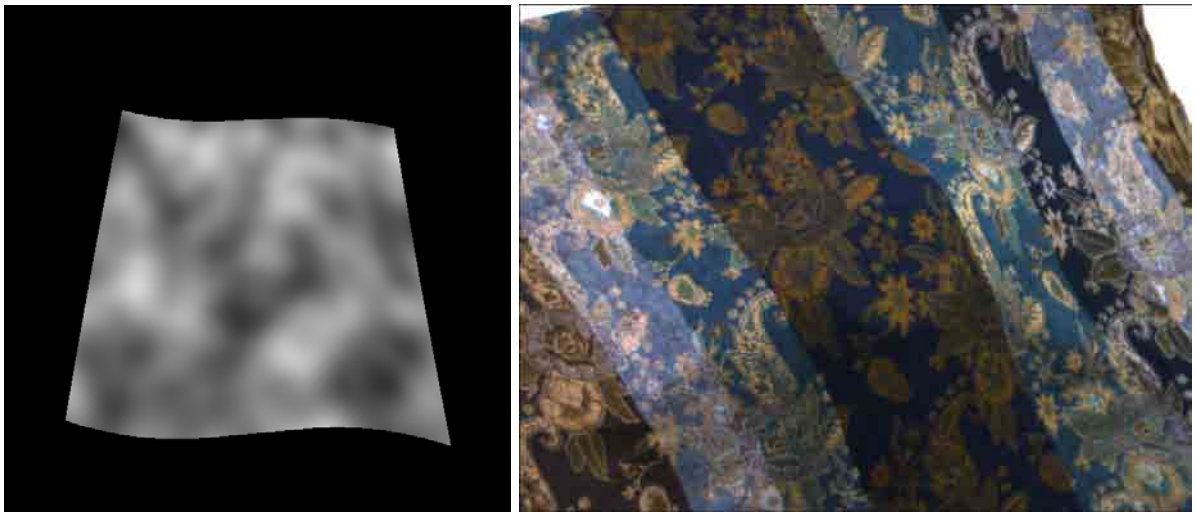

Fig. 4. Left: Sample image from artificial sequence, Right: Sample image from real-world sequence.

Thus, instead of incorporating a pyramidal approach directly into our method, we chose to implement a two-step approach: 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 cost function that we use for that optimization is just a simplified version of the cost function for the intensity based optimization, namely

$$\begin{pmatrix} \pi(T(\mathbf{p}_n, r_{u_1,v_1}(S_{\mathbf{d}}(u_1,v_1))) - (u'_1, v'_1) \\ \pi(T(\mathbf{p}_n, r_{u_2,v_2}(S_{\mathbf{d}}(u_2,v_2))) - (u'_2, v'_2) \\ \vdots \\ \pi(T(\mathbf{p}_n, r_{u_n,v_n}(S_{\mathbf{d}}(u_m,v_m))) - (u'_m, v'_m) \\ S_{\mathbf{d}}(u_1,v_1) - c \\ |\mathbf{q}_n| - 1 \end{pmatrix}.$$

Note that this is basically the original cost function, where the mapping from 2D coordinates to intensity values by application of I_n resp. I_0 has been removed.

In the next step, we apply the original intensity based optimization process to realign the points to the reference intensity values. 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.

Overall, our algorithm performs according to the following scheme:

1) Show the reference frame to the user, allowing him to mark the region of interest in the image.
2) Choose some reference points from the region of interest.
3) Initialize model parameters to represent a plane.
4) For each new image:
   a) Compute optical flow, optimize parameters according to results.
   b) Optimize parameters based on intensity values to prevent drift.

VI. 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 4 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 5 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.

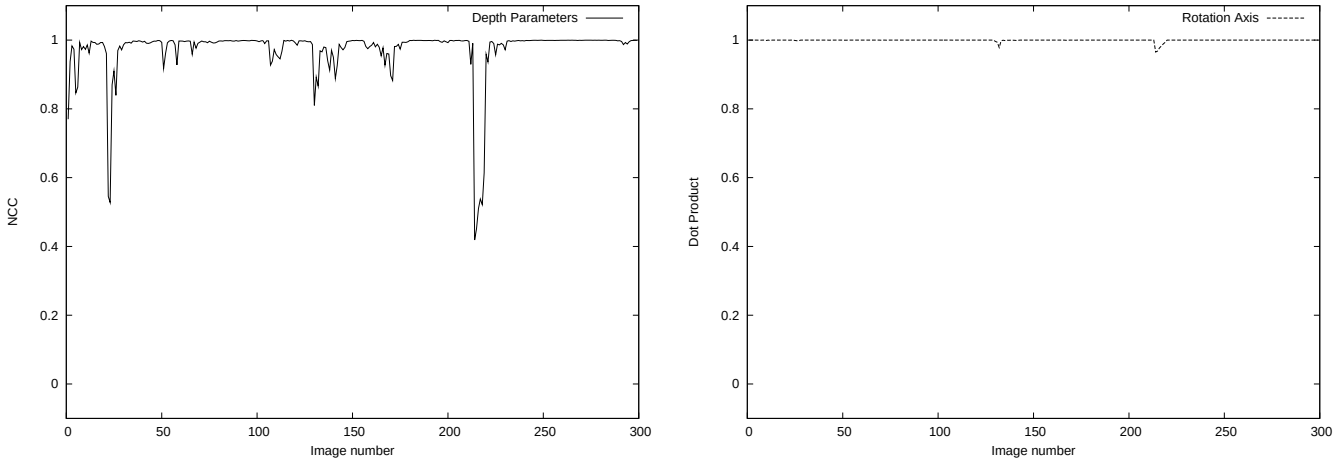

Fig. 5. Left: Plot showing comparison between ground truth depth map parameters and recovered depth map parameters. Right: Analogous comparison of camera parameters.

The right plot in Figure 5 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 4. Figure 6 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 12×12 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 4-5 frames per second were achieved. The major time spent during reconstruction was due to intensity-based optimization. The convergence of the intensity-based optimization was rather slow, which is probably due to the non-convex nature of the cost function in case of large displacements of the tracked pixels to the optimal position. Still, the performance is promising, and we expect it to be possible to further improve performance by pursuing more elaborate optimization schemes.

VII. 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.

Another important issue is the fusion of optimization results to achieve convergence of the reconstructed model. Until now, the model parameters are optimized in each step, starting with the reconstruction results from the previous step. This is obviously not efficient, since the algorithm should be able to accumulate knowledge from the images it has seen, so that the confidence in depth parameters rises over time. This would prevent the problem of the temporary decrease in model quality that we have seen in the results section. A possible idea is to treat the results from our algorithm as measurements for a Kalman filter [**?**] that determines the model that has the highest likelihood.

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.

Until now, we have only used surface models with a fixed level of detail that is uniform for the whole surface. The level of detail is determined by the number of parameters used. In the case of B-Spline surfaces, this directly corresponds to the number of control points, and a higher number of control points would allow us to model more complex surfaces. In nature, however, surfaces often exhibit varying degrees of complexity in different locations: They might have low complexity in one part, but another part might be very com-

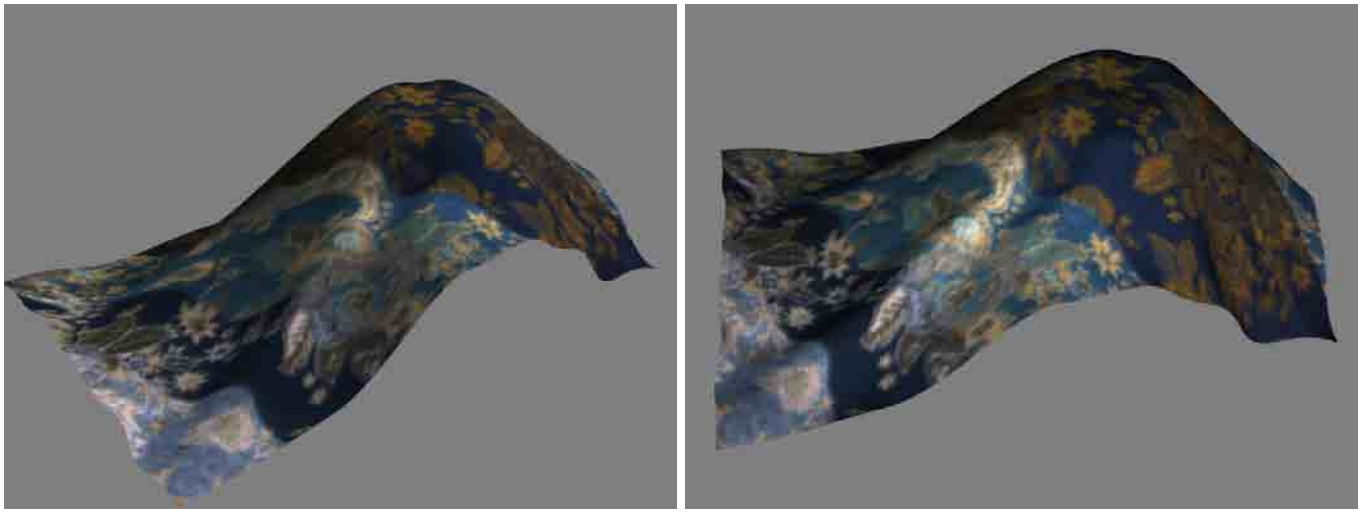

Fig. 6. Reconstruction result from real-world scenario.

plex. Consequently, a more appropriate model would allow to dynamically increase the resolution in some parts, while the resolution in other parts might be decreased. To be able to exploit this idea, we also need to find a method for detecting such areas that need to be modeled with higher resolution.

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.

ACKNOWLEDGMENT

This work was supported by EU IST-FP7 IP project GRASP.

REFERENCES

[1] D. G. Lowe, "Object recognition from local scale-invariant features," in *ICCV*, 1999, pp. 1150–1157.

[2] J. Shi and C. Tomasi, "Good features to track," in *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, 1994, pp. 593 – 600.

[3] N. A. Ramey, J. J. Corso, W. W. Lau, D. Burschka, and G. 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.

[4] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.

[5] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment - a modern synthesis," in *ICCV '99: Proceedings of the International Workshop on Vision Algorithms*. London, UK: Springer-Verlag, 2000, pp. 298–372.

[6] C. Engels, H. Stewénius, and D. Nistér, "Bundle adjustment rules," in *Photogrammetric Computer Vision (PCV)*. ISPRS, Sep. 2006.

[7] W. Sepp, "A direct method for real-time tracking in 3-d under variable illumination," in *DAGM-Symposium*, ser. Lecture Notes in Computer Science, W. G. Kropatsch, R. Sablatnig, and A. Hanbury, Eds., vol. 3663. Springer, 2005, pp. 246–253.

[8] 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*, vol. 1, 1999, pp. 46–53 vol.1.

[9] Y. Shan, Z. Liu, and Z. Zhang, "Model-based bundle adjustment with application to face modeling," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 2, 2001, pp. 644–651 vol.2.

[10] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly Journal of Applied Mathmatics*, vol. II, no. 2, pp. 164–168, 1944.

[11] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *SIAM Journal on Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963. [Online]. Available: http://link.aip.org/link/?SMM/11/431/1

[12] M. Lourakis, "levmar: Levenberg-marquardt nonlinear least squares algorithms in C/C++," [web page] `http://www.ics.forth.gr/~lourakis/levmar/`, Jul. 2004.

[13] L. B. Rall, *Automatic Differentiation: Techniques and Applications*, ser. Lecture Notes in Computer Science. Berlin: Springer, 1981, vol. 120.

[14] A. Griewank and A. Walther, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, 2nd ed., ser. Other Titles in Applied Mathematics. Philadelphia, PA: SIAM, 2008, no. 105.

[15] T. C. Hu and M. T. Shing, "Computation of matrix chain products. part ii," *SIAM J. Comput.*, vol. 13, no. 2, pp. 228–251, 1984.

[16] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Second Edition*. The MIT Press and McGraw-Hill Book Company, 2001.

[17] E. Cohen, "On optimizing multiplications of sparse matrices," in *Proceedings of the 5th International IPCO Conference on Integer Programming and Combinatorial Optimization*. London, UK: Springer-Verlag, 1996, pp. 219–233.

[18] A. Griewank and U. Naumann, "Accumulating jacobians as chained sparse matrix products," *Math. Program.*, vol. 95, no. 3, pp. 555–571, 2003.

[19] U. Naumann, "Optimal jacobian accumulation is np-complete," *Math. Program.*, vol. 112, no. 2, pp. 427–441, 2007.

[20] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision (darpa)," in *Proceedings of the 1981 DARPA Image Understanding Workshop*, April 1981, pp. 121–130.