Rigid 3D Geometry Matching for Grasping of Known Objects in Cluttered Scenes

Chavdar Papazov*, Sami Haddadin†, Sven Parusel†, Kai Krieger†, Darius Burschka*

*Robotics and Embedded Systems
Technische Universität München (TUM)
Boltzmannstr. 3, D-85748 Garching, Germany
{papazov, burschka}@in.tum.de

†Institute of Robotics and Mechatronics
German Aerospace Center (DLR)
P.O. Box 1116, D-82230 Wessling, Germany
{sami.haddadin, sven.parusel, kai.krieger}@dlr.de

Abstract—In this paper, we present an efficient 3D object recognition and pose estimation approach for grasping procedures in cluttered and occluded environments. In contrast to common appearance-based approaches, we rely solely on 3D geometry information. Our method is based on a robust geometric descriptor, a hashing technique and an efficient, localized RANSAC-like sampling strategy. We assume that each object is represented by a model consisting of a set of points with corresponding surface normals. Our method simultaneously recognizes multiple model instances and estimates their pose in the scene. A variety of tests shows that the proposed method performs well on noisy, cluttered and unsegmented range scans in which only small parts of the objects are visible. The main procedure of the algorithm has a linear time complexity resulting in a high recognition speed which allows a direct integration of the method into a continuous manipulation task. The experimental validation with a 7-degrees-of-freedom Cartesian impedance controlled robot shows how the method can be used for grasping objects from a complex random stack. This application demonstrates how the integration of computer vision and soft-robotics leads to a robotic system capable of acting in unstructured and occluded environments.

Fig. 1. A robot operating in a household environment.

1 Introduction

Robot manipulation tasks in non-industrial environments cannot rely on hard-coded knowledge about the scene structure. Since especially human actions modify the environment in a way which cannot be foreseen, a vision-based object recognition and localization system is very useful for providing the necessary updates of the scene knowledge. In recent years, advances in 3D geometry acquisition technology have led to a growing interest in object recognition and pose estimation techniques which operate on three-dimensional data. Furthermore, the knowledge of the 3D geometric shape and the pose of an object greatly facilitates the execution of a stable grasp. The 2D appearance of an object may not provide reliable information about its pose in space because surface texture elements may be misaligned (as it often happens to labels of household objects). Furthermore, 2D techniques have to deal with changes in viewpoint and illumination.

The 3D object recognition and pose estimation problem can loosely be defined as follows. Given a set of object models and a scene, the task is to identify the objects present in the scene and to estimate their position and orientation. The output of an object recognition and pose estimation algorithm is a list of recognized model instances each one with a corresponding transform which aligns the model instance to the scene. For the sake of simplicity, in the rest of the text, we mean by "object recognition" *both* object identification and pose estimation. Furthermore, we

discuss a special instance of the problem, given by the following assumptions.

1) Each model is a finite set of points with corresponding surface normals.
2) Each model represents a non-transparent object.
3) The scene is a range image.
4) Each transform which aligns a recognized model instance to the scene is a proper rigid transform.

Even under these assumptions the problem still remains challenging for several reasons: it is a priori not known which objects are present in the scene; usually, there are scene parts not belonging to any of the objects of interest, i.e., there is background clutter; the input is typically corrupted by noise and outliers; the objects are only partially visible due to occlusions and scan device limitations.

1.1 Contributions and Overview

This paper demonstrates how our original 3D object recognition approach presented in [Papazov and Burschka, 2010] can be used to support a manipulation task. We introduce a vision-based framework that allows a robotic manipulator to grasp objects in unstructured, dynamically changing environments.

Our object recognition approach operates directly on unsegmented point clouds provided by a range scanner. This does not require scene segmentation which may be quite time consuming. More specifically, we make the following contributions. (i) A new efficient, localized RANSAC-like sampling strategy is introduced. (ii) We use a hash table for rapid retrieval of pairs of oriented model points which are similar to a sampled pair of oriented scene points. This allows to efficiently generate object and pose hypotheses. (iii) We provide a complexity analysis of our sampling strategy and derive a formula for the number of iterations required to recognize the objects with a predefined success probability.

The proposed accelerations in our vision processing allow a seamless integration into a grasping framework, where the recognition interleaves with the actual manipulation task without causing noticeable delays in the overall process. The method shows its potential in a complex experimental use-case. We employ the DLR Lightweight Robot III (LWR-III) [Albu-Schäffer et al., 2007], which is equipped with a Cartesian impedance control method and is able to react to environment disturbances and to process faults caused by unexpected contact forces in real-time. Using the proposed 3D object recognition method, impedance control with reactive recovery strategies, and a simple grasp planner the robot quickly and robustly grasps objects from unsorted and cluttered piles. Furthermore, in case of failures, it reacts accordingly and continues the process if possible[1].

1. These experiments are enclosed as multimedia extensions (see Extensions 1 and 2 in Appendix A).

The rest of the paper is organized as follows. Previous work is reviewed in Section 2. In Section 3, we establish some notation and explain in more details two algorithms important to our work. Our 3D object recognition approach is introduced in Section 4. In Section 5, the robot and its overall control concept for robust and sensitive grasping is described. Section 6 presents experimental results. Conclusions and possible future topics of research are drawn in the final Section 7 of the paper.

2 Related Work

Object recognition is closely related to object classification/shape retrieval. However, the latter methods only compute the similarity between shapes and not an aligning transform. Furthermore, it is assumed that the input represents a single shape whereas we handle range images containing multiple objects and background clutter.

The so-called voting approaches represent one class of object recognition methods. Well-known representatives are the generalized Hough transform [Ballard, 1981], pose clustering [Stockman, 1987], geometric hashing [Lamdan and Wolfson, 1988] and tensor matching [Mian et al., 2006]. Although these methods perform well on complex scenes they tend to be costly and their integration into a real-world grasping system will lead to long delays in the overall processing loop.

Another way to tackle the problem is to model an object as an assembly of basic shapes (primitives) and to recover these shapes and their spatial relationships from an input scene. Many types of primitives can be used within this part-based framework: generalized cylinders [Binford, 1971], superquadrics [Barr, 1981], implicit polynomials [Keren et al., 1994], geometric primitives [Taylor and Kleeman, 2003], and parametric shapes [Schnabel et al., 2007]. Methods for efficient recovering of superquadrics from range data were introduced in [Solina and Bajcsy, 1990], [Dickinson et al., 1997] and [Biegelbauer et al., 2010]. However, despite their efficiency, the part-based approaches are limited to a certain shape class, namely, the one which can be described by the chosen set of primitives.

The feature-based methods belong to a further class of object recognition approaches. In a first step, point-wise correspondences between the models and the scene are computed, usually using local geometric descriptors. Next, the aligning transform is calculated based on the established correspondences. A list of geometric descriptors includes, without being nearly exhaustive, spin images [Johnson and Hebert, 1999], local feature histograms [Hetzel et al., 2001], 3D/harmonic shape context [Frome et al., 2004], intrinsic isometry invariant descriptors [Sun et al., 2009] and manifold harmonic bases [Wu et al., 2010]. All feature-based algorithms rely on the assumption that the objects to

be recognized have distinctive feature points, i.e., points with rare descriptors. However, especially for simple shapes, this assumption does not always hold and the methods degenerate to brute force search [Aiger et al., 2008].

In [Grewe and Kak, 1995], a hashing technique similar to ours was proposed. It is a learning-based method employing a multiple-attribute hash table for efficient 3D object recognition. On the positive side, attribute uncertainties are taken into account and the number of attributes as well as the size of the hash table bins are calculated automatically. However, the system cannot handle free-form objects and in the presented experimental results only objects composed of single-colored surfaces are used. Furthermore, the method relies on a segmentation to identify the planar or cylindrical surface patches the objects are made of.

A further hashing technique was proposed in [Matei et al., 2006]. Based on a hash table, a fast indexing into a collection of geometry descriptors of *single* model points is performed. In contrast, our hash table stores descriptors of *pairs* of oriented model points (called doublets). This allows to efficiently query the model doublets similar to a sampled scene doublet and it makes it very easy to compute the aligning rigid transform since it is uniquely defined by two corresponding doublets. Moreover, in [Matei et al., 2006], multiple range images are aligned to each other in order to build a more complete scene representation and a foreground/background segmentation is executed. In contrast, our method operates on a single range image and does not require segmentation. Furthermore, the test scenes used in [Matei et al., 2006] contain a single object and some background clutter.

3 Notation and Basic Algorithms

An oriented point $\mathbf{u} = (\mathbf{p}_u, \mathbf{n}_u)$ consists of a point $\mathbf{p}_u \in \mathbb{R}^3$ and a corresponding surface normal $\mathbf{n}_u \in \mathbb{R}^3$, $\|\mathbf{n}_u\| = 1$. Accordingly, an oriented point pair $(\mathbf{u}, \mathbf{v})$ is a pair of two oriented points $\mathbf{u} = (\mathbf{p}_u, \mathbf{n}_u)$ and $\mathbf{v} = (\mathbf{p}_v, \mathbf{n}_v)$.

3.1 Fast Surface Registration

In short, rigid surface registration consists of computing a rigid transform which aligns two surfaces. Assume $\mathbf{S}$ is a surface represented by a set of oriented points. According to [Winkelbach et al., 2006], 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 computed as

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

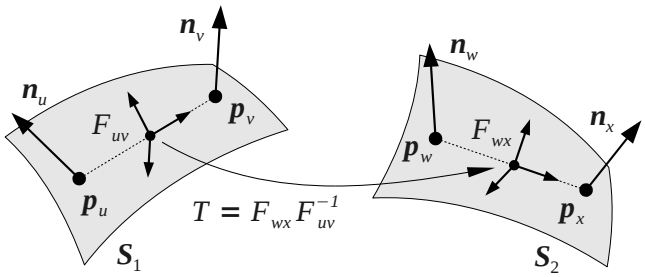


Fig. 2. Computing the rigid transform T which aligns $\mathbf{S}_1$ to $\mathbf{S}_2$ based on the local coordinate systems F_{uv} and F_{wx} of the oriented point pairs $(\mathbf{u}, \mathbf{v})$ and $(\mathbf{w}, \mathbf{x})$, respectively. See text for details on F_{uv} and F_{wx}.

where $\angle(\mathbf{a},\, \mathbf{b})$ is the angle between the vectors $\mathbf{a}$ and $\mathbf{b}$. In order to register two surfaces $\mathbf{S}_1$ and $\mathbf{S}_2$, each one represented by a set of oriented points, the method proceeds as follows. It samples uniformly oriented point pairs $(\mathbf{u}, \mathbf{v}) \in \mathbf{S}_1 \times \mathbf{S}_1$ and $(\mathbf{w}, \mathbf{x}) \in \mathbf{S}_2 \times \mathbf{S}_2$ and computes and stores their descriptors $f(\mathbf{u}, \mathbf{v})$ and $f(\mathbf{w}, \mathbf{x})$ in a four-dimensional hash table. This process continues until a collision occurs, i.e., until $f(\mathbf{u}, \mathbf{v})$ and $f(\mathbf{w}, \mathbf{x})$ end up in the same hash table cell. Computing the rigid transform T which aligns $(\mathbf{u}, \mathbf{v})$ to $(\mathbf{w}, \mathbf{x})$ gives a transform hypothesis which registers $\mathbf{S}_1$ to $\mathbf{S}_2$. Fig. 2 illustrates the alignment. More formally,

$$T = F_{wx} F_{uv}^{-1} \tag{2}$$

is computed based on the pairs' local coordinate systems, each one represented by a 4×4 matrix (homogeneous coordinates) F_{uv} respectively F_{wx}. We have

$$F_{uv} = \begin{pmatrix} \frac{\mathbf{p}_{uv} \times \mathbf{n}_{uv}}{\|\mathbf{p}_{uv} \times \mathbf{n}_{uv}\|} & \frac{\mathbf{p}_{uv}}{\|\mathbf{p}_{uv}\|} & \frac{\mathbf{p}_{uv} \times \mathbf{n}_{uv} \times \mathbf{p}_{uv}}{\|\mathbf{p}_{uv} \times \mathbf{n}_{uv} \times \mathbf{p}_{uv}\|} & \frac{\mathbf{p}_u + \mathbf{p}_v}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3}$$

where $\mathbf{p}_{uv} = \mathbf{p}_v - \mathbf{p}_u$ and $\mathbf{n}_{uv} = \mathbf{n}_u + \mathbf{n}_v$. F_{wx} is defined analogously by replacing the indices u and v in (3) with w and x, respectively. The transform hypothesis T generated in this way is evaluated by transforming the points of $\mathbf{S}_1$, i.e., $\mathbf{p}_i' = T\mathbf{p}_i$, $\forall \mathbf{p}_i \in \mathbf{S}_1$ and counting those $\mathbf{p}_i'$ which fall within a certain ϵ-band of $\mathbf{S}_2$.

According to [Winkelbach et al., 2006], this process of generating and evaluating hypotheses is repeated until either of the following stopping criteria is met: (i) a hypothesis is good enough, (ii) a predefined time limit is reached or (iii) all combinations are tested. Unfortunately, non of these criteria is well-grounded: the first two are ad hoc and the third one is computationally infeasible. In contrast, we derive the number of iterations required to recognize model instances with a pre-defined success probability. Furthermore, we modify this technique in a way which allows for the *simultaneous* matching of all object models to the scene.

3.2 RANSAC

RANSAC [Fischler and Bolles, 1981] is an iterative approach for model recognition. It uniformly draws minimal point sets from the scene and computes a transform which aligns the model with the minimal point set. A minimal point set is the smallest point set which uniquely determines a given type of transform. In the case of rigid transforms, it is a point triple. The score of the aligning transform is the number of transformed model points which lie within an ϵ-band of the scene. After a certain number of trials the model is considered to be recognized at the locations defined by the transforms which achieved a score higher than a predefined threshold.

The probability P_S of recognizing the model in N trials equals the complementary of N consecutive failures [Schnabel et al., 2007], i.e.,

$$P_S = 1 - (1 - P_M)^N, \tag{4}$$

where P_M is the probability of recognizing the model in a single iteration. Solving for N gives the number of trials needed to recognize the model in the scene:

$$N = \frac{\ln(1 - P_S)}{\ln(1 - P_M)}. \tag{5}$$

Note that since $P_S \approx 1$ and $P_M \approx 0$, one can safely assume that $N \geq 1$.

The RANSAC approach is conceptually simple, general and robust against outliers. Unfortunately, its direct application to the 3D object recognition problem is computationally expensive. In order to compute an aligning rigid transform, we need two corresponding point triples—one from the model and one from the scene. Assuming that the model is completely contained in the scene, the probability of drawing two such triples in a single trial is $P_M(n) = \frac{3!}{(n-2)(n-1)n}$, where n is the number of scene points. Since $P_M(n)$ is a small number we can approximate the denominator in (5) by its Taylor series $\ln(1 - P_M(n)) = -P_M(n) + O(P_M(n)^2)$ and obtain the number of trials as a function of the number of scene points:

$$N(n) \approx \frac{-\ln(1 - P_S)}{P_M(n)} = O(n^3). \tag{6}$$

In the next section of the paper, we will show that using oriented point pairs and a localized sampling strategy leads to a reduction of the time complexity from $O(n^3)$ to $O(n)$.

4 Rigid 3D Geometry Matching

Our object recognition method consists of two phases. The first one, the model preprocessing, is performed offline. It is executed only once and does not depend on the scenes in which the objects have to be recognized. The online recognition is the second phase. It is executed on the scene using the model representation computed in the offline phase. In the rest of this section, we describe both phases in detail and discuss the time complexity of our algorithm.

4.1 Model Preprocessing Phase

We assume that each object model is represented by a finite set $\mathbf{M} = \{\mathbf{u}_i = (\mathbf{p}_u, \mathbf{n}_u)_i\}_{i=1}^m$ of oriented points. For a given object model $\mathbf{M}$, we sample the pairs of oriented points $(\mathbf{u}, \mathbf{v}) = ((\mathbf{p}_u, \mathbf{n}_u), (\mathbf{p}_v, \mathbf{n}_v)) \in \mathbf{M} \times \mathbf{M}$ having $\mathbf{p}_u$ and $\mathbf{p}_v$ approximately d units apart from each other. For each such pair, the descriptor $f(\mathbf{u}, \mathbf{v}) = (f_2(\mathbf{u}, \mathbf{v}),\ f_3(\mathbf{u}, \mathbf{v}), f_4(\mathbf{u}, \mathbf{v}))$ is computed according to (1) and stored in a three-dimensional hash table. Note that f_1 is not part of the descriptor since a fixed distance d is used. In contrast to [Winkelbach et al., 2006], *not* all pairs of oriented points are considered, but only those with $\|\mathbf{p}_u - \mathbf{p}_v\| \in [d - \delta_d, d + \delta_d]$, for a given tolerance value δ_d. This has several advantages. It reduces the space complexity from $O(m^2)$ to $O(m)$, where m is the number of model points (this empirical measurement is further discussed in [Aiger et al., 2008]). Using a large d results in wide-pairs which allow a more stable computation of the aligning rigid transform than narrow-pairs do [Aiger et al., 2008]. Furthermore, a larger d leads to fewer pairs which means that computing and storing descriptors of wide-pairs results in less populated hash table cells. Thus, we will have to test fewer transform hypotheses in the online recognition phase and will save computation time.

However, the pair width d should not be too large since occlusions in real world scenes would prevent sampling a pair with points from the same object. For a typical value for d, there are still many pairs with similar descriptors which leads to hash table cells with too many entries. We avoid this overpopulation, by removing as many of the most populated cells to keep only a fraction K of the original number of pairs (in our implementation $K = 0.4$). This results in an information loss about the object shape which we take into account in the online phase of the algorithm.

In order to compute the final representation of all models $\mathbf{M}_1, \ldots, \mathbf{M}_q$, each $\mathbf{M}_i$ is processed in the way described above using the *same* hash table. In this way, a simultaneous recognition of all models is possible instead of sequentially matching each one of them to the scene. Furthermore, in order to keep track of which pair belongs to which model, every hash table cell stores the pairs in separate model-specific lists.

4.2 Online Recognition Phase

The input to the online recognition algorithm is a set $\mathcal{M} = \{\mathbf{M}_1, \ldots, \mathbf{M}_q\}$ of object models and a scene range image $\mathbf{S}$. The output is a list $\mathcal{T} = \{(\mathbf{M}_{k_1}, T_1), \ldots, (\mathbf{M}_{k_r}, T_r)\}$, where $\mathbf{M}_{k_j} \in \mathcal{M}$ is a recognized model instance and T_j is a proper rigid transform (an element of the special Euclidean group $SE(3)$) aligning $\mathbf{M}_{k_j}$ to the scene. Before turning to the details, it is advisable to read Algorithm 1 although some of the steps may not be completely clear at this point. In the rest of this section, the lines we are referring to are the lines of Algorithm 1.

input : a set $\mathcal{M} = \{\mathbf{M}_1, ..., \mathbf{M}_q\}$ of object models; a scene range image $\mathbf{S}$;
output: a list $\mathcal{T} = \{(\mathbf{M}_{k_1}, T_1), ..., (\mathbf{M}_{k_r}, T_r)\}$, with $\mathbf{M}_{k_j} \in \mathcal{M}$ and $T_j \in SE(3)$;

// 1) initialization
1 compute an octree for the scene $\mathbf{S}$ to produce a modified scene $\mathbf{S}^*$;
2 $\mathcal{T} \leftarrow \emptyset$; // an empty solution list
// 2) number of iterations
3 compute the number N of iterations;
4 **repeat** N *times*
// 3) sampling
5 sample a $\mathbf{p}_u$ uniformly from $\mathbf{S}^*$;
6 $\mathbf{L} = \{\mathbf{x} \in \mathbf{S}^* \ : \ \|\mathbf{x} - \mathbf{p}_u\| \in [d - \delta_d, d + \delta_d]\}$;
7 sample a $\mathbf{p}_v$ uniformly from $\mathbf{L}$;
// 4) normal estimation
8 estimate normals $\mathbf{n}_u$ at $\mathbf{p}_u$ and $\mathbf{n}_v$ at $\mathbf{p}_v$;
9 $(\mathbf{u}, \mathbf{v}) = ((\mathbf{p}_u, \mathbf{n}_u), (\mathbf{p}_v, \mathbf{n}_v))$;
// 5) hash table access
10 $f_{\mathbf{uv}} = (f_2(\mathbf{u}, \mathbf{v}), \ldots, f_4(\mathbf{u}, \mathbf{v}))$; // see (1)
11 access the model hash table cell at $f_{\mathbf{uv}}$ **and**
12 get its oriented model point pairs $(\mathbf{u}_j, \mathbf{v}_j)$;
// 6) generate and test
13 **foreach** $(\mathbf{u}_j, \mathbf{v}_j)$ **do**
14 get the model $\mathbf{M}$ of $(\mathbf{u}_j, \mathbf{v}_j)$;
15 compute the rigid transform T that aligns $(\mathbf{u}_j, \mathbf{v}_j)$ to $(\mathbf{u}, \mathbf{v})$; // see (2)
16 **if** μ *accepts* $(\mathbf{M}, T)$ **then**
17 $\mathcal{T} \leftarrow \mathcal{T} \cup (\mathbf{M}, T)$;
18 **end**
19 **end**
20 **end**
// 7) removing conflicting hypotheses
21 remove conflicting hypotheses from $\mathcal{T}$;

Algorithm 1: Online recognition phase.

Searching for closest points (line 8) and for points lying on a sphere around a given point (line 6) have to be performed very often in the online recognition phase. Thus, a fast execution of these operations is of great importance for the runtime of the algorithm. An efficient way to achieve this is to use an octree [de Berg et al., 2000].

Step 1) Initialization

In step 1 of the algorithm, an octree with a fixed leaf size L (the edge length of a leaf) is constructed for the input scene points. The full octree leaves (the ones containing at least one point) are voxels ordered in a regular axis-aligned 3D grid and have unique integer coordinates. Two full leaves are considered neighbors if their corresponding integer coordinates differ by not more than 1. Next, a down-sampled scene $\mathbf{S}^*$ is created by setting its points to be the centers of mass of the full octree leaves. The center of mass of a full leaf is the average of the points it contains. In this way, a one-to-one correspondence between the points in $\mathbf{S}^*$ and the full octree leaves is established. Two points in $\mathbf{S}^*$ are neighbors if the corresponding leaves are neighbors.

Step 2) Number Of Iterations

In this step, the number N of iterations is estimated such that all objects in the scene will be recognized with a certain user-defined probability. This will be explained in detail in Section 4.3.

Step 3) Sampling

As in classic RANSAC, we sample minimal sets from the scene. In our case, since we use normals, a minimal set consists of two oriented points. In contrast to RANSAC, they are *not* sampled uniformly and independently of each other. Only the first point, $\mathbf{p}_u$, is drawn uniformly from $\mathbf{S}^*$. The second one, $\mathbf{p}_v$, is drawn uniformly from the scene points in $\mathbf{S}^*$ which are approximately within a distance d from $\mathbf{p}_u$. To achieve this, we first retrieve the set $\mathbf{L}$ of all full leaves intersecting the sphere with center $\mathbf{p}_u$ and radius d, where d is the pair width used in the offline phase (see Section 4.1). This can be performed very efficiently due to the hierarchical structure of the octree. Finally, a leaf is drawn uniformly from $\mathbf{L}$ and $\mathbf{p}_v$ is set to be its center of mass.

Step 4) Normal Estimation

We estimate the normals $\mathbf{n}_u$ and $\mathbf{n}_v$ at the points $\mathbf{p}_u$ and $\mathbf{p}_v$ by performing a PCA: $\mathbf{n}_u$ and $\mathbf{n}_v$ are set to be the eigenvectors corresponding to the smallest eigenvalues of the covariance matrix of the points in the neighborhood of $\mathbf{p}_u$ and $\mathbf{p}_v$. The result of this step is the oriented scene point pair $(\mathbf{u}, \mathbf{v}) = ((\mathbf{p}_u, \mathbf{n}_u), (\mathbf{p}_v, \mathbf{n}_v))$.

Step 5) Hash Table Access

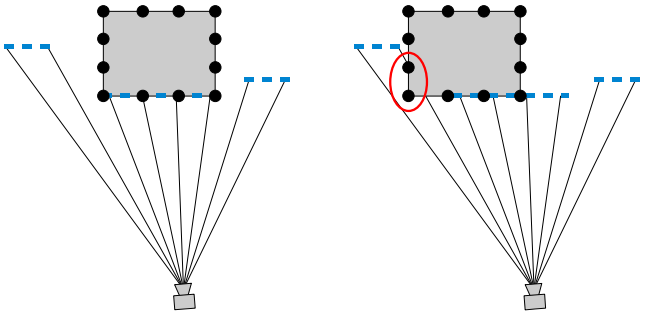
In line 10, $f_{\mathbf{uv}} = (f_2(\mathbf{u}, \mathbf{v}), f_3(\mathbf{u}, \mathbf{v}), f_4(\mathbf{u}, \mathbf{v}))$ is computed according to (1). Next, in lines 11 and 12, $f_{\mathbf{uv}}$ is used as a key to the model hash table to retrieve all model pairs $(\mathbf{u}_j, \mathbf{v}_j)$ similar to $(\mathbf{u}, \mathbf{v})$.

Step 6) Generate and Test

For each $(\mathbf{u}_j, \mathbf{v}_j)$, its model $\mathbf{M}$ is retrieved (line 14) and the rigid transform T which aligns $(\mathbf{u}_j, \mathbf{v}_j)$ to $(\mathbf{u}, \mathbf{v})$ is computed according to (2) (line 15). This results in the hypothesis that the model $\mathbf{M}$ is in the scene at the location defined by T. Finally, the hypothesis is saved in the solution list $\mathcal{T}$ if it is accepted by the acceptance function μ (line 16).

Acceptance Function

μ consists of a visibility term and a penalty term. Similar to RANSAC, the visibility term, μ_V, is proportional to the number m_V of transformed model points which fall within a certain ϵ-band of the scene. More precisely, we set $\mu_V(\mathbf{M}, T) = m_V/m$, where m



(a) $\mu_V = 4/12,\ \mu_P = 0.$ (b) $\mu_V = 3/12,\ \mu_P = 2/12.$

Fig. 3. A 2D top schematic view of the same scene (blue dashed line) with two different model hypotheses (models are shown as gray boxes). The lines of sight are shown as thin black lines originating from the scanning device. In (a), 4 (out of 12) model points match the scene and no model points are occluding scene points. In (b), 3 model points match the scene and 2 model points (marked by the ellipse) are occluding scene points. The resulting values for μ_V and μ_P are shown below the corresponding figure.

is the total number of model points. μ_V is an approximation of the visible object surface area expressed as a fraction of the total object surface area. Thus, μ_V can be interpreted as an estimation of the object visibility in the scene.

In contrast to RANSAC, our algorithm contains an additional penalty term, μ_P, which is proportional to the number of transformed model points which occlude the scene. Obviously, a correctly recognized and localized non-transparent object should not occlude any visible scene points when the scene is viewed from the viewpoint of the range scanner. In other words, if we view the scene from the perspective of the scanning device, we will not be able to see scene points lying behind the localized model since we cannot see through non-transparent surfaces. The penalty term penalizes hypotheses which violate this condition. It is computed by counting the number m_P of transformed model points which are between the projection center of the range image and a range image pixel and thus are "occluding" reconstructed scene points. We set $\mu_P(\mathbf{M}, T) = m_P/m$, where m is the number of model points.

For $(\mathbf{M}, T)$ to be accepted by μ as a valid hypothesis it has to fulfill

$$\mu_V(\mathbf{M}, T) > V \text{ and } \mu_P(\mathbf{M}, T) < P, \tag{7}$$

where $V \in [0, 1]$ is a visibility and $P \in [0, 1]$ a penalty threshold. In Fig. 3, a simple scene is shown with two different model hypotheses and the corresponding values for μ_V and μ_P.

The visibility threshold is one of the most crucial parameters in the algorithm. In the experimental part of the paper, we examine how this threshold affects the recognition and the false positives rates of our method. In the case of perfect data, the penalty threshold P should be 0. However, since we are dealing with real range images, we use $P = 0.05$.

Step 7) Removing Conflicting Hypotheses

A hypothesis $(\mathbf{M}, T)$ "explains" a subset $\mathbf{P} \subset \mathbf{S}^*$ if there are points from $T(\mathbf{M})$ lying in the octree leaves corresponding to $\mathbf{P}$. After accepting $(\mathbf{M}, T)$, the points explained by it are *not* removed from $\mathbf{S}^*$ because there could be a better hypothesis, i.e., one which explains a superset of $\mathbf{P}$. We call hypotheses conflicting if the intersection of the point sets they explain is non-empty. In other words, conflicting hypotheses transform their models such that they intersect in space.

Since the scene points explained by the accepted hypotheses are not removed from $\mathbf{S}^*$, there are many conflicting ones in the solution list $\mathcal{T}$ after the execution of the main loop (lines 4 to 20) of Algorithm 1. To filter the weak hypotheses, we construct a so-called conflict graph. Its nodes are the hypotheses in $\mathcal{T}$ and an edge is connecting two nodes if the hypotheses are conflicting ones.

To produce the final output, the solution list is filtered by performing a non-maximum suppression on the conflict graph. We borrow this technique from image processing. To perform a non-maximum suppression on a gray-scale image, the pixel under observation is set to zero (it is suppressed) if its value is not a maximum in a window placed around that pixel. In this case, the window defines the neighborhood of each pixel. In the case of our conflict graph, the neighborhood is defined by the graph structure. Using the neighborhood of each node, we perform non-maximum suppression essentially in the same way as in image processing: a node η is suppressed if there is a better one in its neighborhood, i.e., a node which explains more scene points than η.

4.3 Time Complexity

The dominating factor in the complexity of the proposed method is the number N of iterations needed to recognize all models with a predefined success probability (see the main loop of Algorithm 1, lines 4 to 20). In the following, we discuss the dependency of N on the number of scene points.

Consider a scene $\mathbf{S}^*$ consisting of $|\mathbf{S}^*| = n$ points and a model instance $\mathbf{M}$ therein consisting of $|\mathbf{M}| = m$ points. In Section 3.2 on RANSAC, we derived the number $N = \frac{\ln(1-P_S)}{\ln(1-P_M)}$ of iterations required to recognize $\mathbf{M}$ with a predefined success probability P_S, where P_M is the probability of recognizing $\mathbf{M}$ in a single iteration. Again in Section 3.2, we obtained $P_M \approx 1/n^3$ which resulted in the cubic time complexity of RANSAC. In the following, we show that our sampling strategy and the use of the model hash table lead to a significant increase of P_M and thus to a reduction of the complexity.

If $P(\mathbf{p}_u \in \mathbf{M}, \mathbf{p}_v \in \mathbf{M})$ denotes the probability that both points are sampled from $\mathbf{M}$ (lines 5 and 7 of Algorithm 1), then the probability of recognizing $\mathbf{M}$ in a single iteration is

$$P_M = KP(\mathbf{p}_u \in \mathbf{M}, \mathbf{p}_v \in \mathbf{M}), \tag{8}$$

with K being the fraction of oriented point pairs for which the descriptors are stored in the model hash table (see Section 4.1). Using conditional probability and the fact that $P(\mathbf{p}_u \in \mathbf{M}) = m/n$ we can rewrite (8) to obtain

$$P_M = (m/n)KP(\mathbf{p}_v \in \mathbf{M} | \mathbf{p}_u \in \mathbf{M}). \tag{9}$$

$P(\mathbf{p}_v \in \mathbf{M}|\mathbf{p}_u \in \mathbf{M})$ denotes the probability to sample $\mathbf{p}_v$ from $\mathbf{M}$ given that $\mathbf{p}_u \in \mathbf{M}$. Recall from Section 4.2 that $\mathbf{p}_v$ depends on $\mathbf{p}_u$ because it is sampled uniformly from the set $\mathbf{L}$ of scene points which are close to the sphere $\mathcal{S}_d(\mathbf{p}_u)$ with center $\mathbf{p}_u$ and radius d, where d is the pair width used in the offline phase. Assuming that the visible object part has an extent larger than $2d$ and that the reconstruction is not too sparse, $\mathbf{L}$ contains points from $\mathbf{M}$. In this case, $P(\mathbf{p}_v \in \mathbf{M}|\mathbf{p}_u \in \mathbf{M}) = |\mathbf{L} \cap \mathbf{M}|/|\mathbf{L}|$ is well-defined and greater than zero.

Let us discuss $C := |\mathbf{L} \cap \mathbf{M}|/|\mathbf{L}|$. It depends on the scene clutter, the number of outliers and the extent and shape of the visible object part. If all scene points originate from known objects (in particular there is no background) and if the objects are well separated then $|\mathbf{L} \cap \mathbf{M}| = |\mathbf{L}|$ since the sphere $\mathcal{S}_d(\mathbf{p}_u)$ intersects scene octree leaves containing only points from the object $\mathbf{p}_u$ belongs to. In this extreme case, $C = 1$. On the other hand, occluded scenes with many outliers can be constructed in which $\mathcal{S}_d(\mathbf{p}_u)$ intersects only objects other then the one $\mathbf{p}_u$ belongs to. This leads to $C = 0$ and simply means that the object is too occluded to be recognized.

In our implementation, we estimate C by $1/4$. This accounts for up to 75% outliers and scene clutter. Thus, we obtain for P_M as a function of n (the number of scene points)

$$P_M(n) = (m/n)KC. \tag{10}$$

Approximating the denominator $\ln(1 - P_M(n))$ in (5) by its Taylor series $-P_M(n) + O(P_M(n)^2)$ we obtain for the number of iterations

$$N(n) \approx \frac{-\ln(1 - P_S)}{P_M(n)} = \frac{-n \ln(1 - P_S)}{mKC} = O(n). \tag{11}$$

This shows that the number of iterations depends linearly on the number n of scene points. Furthermore, Eq. (11) provides means for computing the number of iterations required to recognize the model instances with the desired success probability P_S.

5 Object Manipulation

The object recognition is only one link in the overall processing chain. In order to enable the robot to perform the recognition together with the object manipulation in a loop and to recover from faulty grasps, the overall process is controlled by a hybrid state automaton. The scheme is based on the work in [Haddadin et al., 2009], [Parusel et al., 2011], [Fuchs et al., 2010] and relies on the disturbance observer introduced in [Haddadin et al., 2008]. The high-level schematic is shown in Fig. 4 and consists of the following phases:

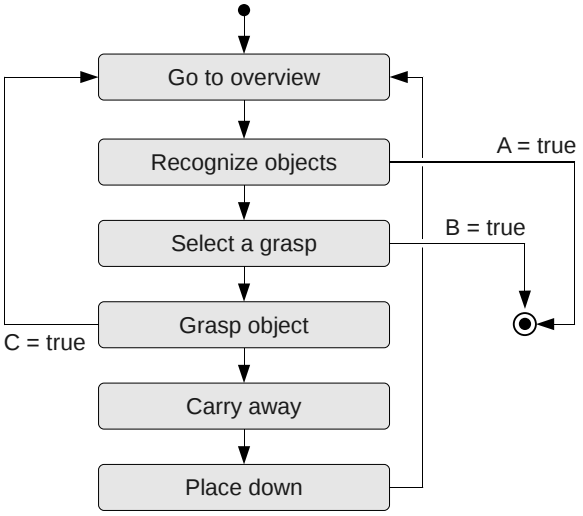


Fig. 4. Hybrid state automaton for controlling the overall object manipulation process. The logical clauses A, B and C defining the transition conditions, are the following: $A = no - object$, $B = no - grasp$ and $C = collision \vee failed - grasp$. A grasp is considered as a failure if the robot gripper completely closes.

1) Go to overview: The robot moves to an overview pose in order to avoid scene occlusion.
2) Recognize objects: The object recognition method recognizes the objects in the scene (see Section 4).
3) Select a grasp: Select an object (from the list of recognized ones) together with a suitable grasp (see Section 5.1).
4) Grasp object: The robot performs the grasp on the selected object (see Section 5.2).
5) Carry away: The robot carries the object to the place designated for it.
6) Place down: The robot softly and safely puts the object on its place (see Section 5.2).

Next, the phases 3), 4) and 6) are explained in detail. Phase 5), bringing an object to a specified location, is out of the scope of this paper and will not be discussed any further.

5.1 Grasp Selection

The first step in the manipulation chain is the selection of an object (from the list of recognized objects returned by the recognition method) together with a suitable grasp. Each object in the database is associated with a finite set of plausible grasps. A grasp

G (also called grasp frame) consists of an orientation and a position of the robot end effector relative to the object. The orientation is represented by a 3×3 rotation matrix. Its last column is a vector, called the approach vector $\mathbf{v}_{appr}$, which is aligned with the z-axis of the end effector and points towards the object.

The idea of the grasp selection is to go for the up most object, i.e., the one whose center of mass has the largest z-coordinate. This is measured according to the world coordinate system which has its z-axis, $\mathbf{z}_w$, perpendicular to the table the objects are placed on. (More details on the scene setup will be given in Section 6.3.) Next, among the grasp frames associated with the selected object, the one with the lowest

$$\mathrm{cost}(G) = w_1 \mathrm{crit}_1(G) + w_2 \mathrm{crit}_2(G), \tag{12}$$

is chosen, where

$$\mathrm{crit}_1(G) = \angle(-\mathbf{v}_{appr}, \mathbf{z}_w) \tag{13}$$

is the alignment of the end effector with respect to the z-axis of the world coordinate system and

$$\mathrm{crit}_2(G) = |\varphi_0^{wrist} - \varphi_{req}^{wrist}|, \tag{14}$$

is the absolute difference between φ_0^{wrist}, a desired (neutral) wrist orientation, and φ_{req}^{wrist}, the one required to perform the grasp. In our implementation, we set $w_1 = 3$ and $w_2 = 1$ (see (12)) which makes the end effector alignment more important than the wrist orientation. Note that (12) uses the negative of $\mathbf{v}_{appr}$ such that it shows in the same direction as $\mathbf{z}_w$. Furthermore, the selected grasp is discarded if it is "too parallel" to the table plane, i.e., if $\mathrm{crit}_1(G) \geq \varphi_z$ (we set $\varphi_z = 30°$). If this is the case, then the next lower object is inspected.

The selected grasp frame G is projected 0.1 m along the approach vector for acquiring the grasping motion.

5.2 Impedance Control and Collision Detection for Sensitive Grasping and Placing

To achieve robust and careful grasping and placing, we employ the Cartesian impedance controlled LWR-III [Albu-Schäffer et al., 2007]. Especially its soft-robotics features are crucial for such a delicate task. Since it is equipped with torque sensors in every joint, both impedance and accurate position control are possible at the same time. In order not to damage its environment and in particular the objects it is supposed to manipulate, the robot should be able to quickly detect collisions and safely react to them. The collision detection method we use was introduced and evaluated in [De Luca et al., 2006], [Haddadin et al., 2008]. It provides not only binary contact information but also an accurate estimation of the external torques. Based on this additional input, we employ a decision algorithm [Haddadin et al., 2009] which enables the robot to react in an appropriate manner to unexpected interaction forces. For example, such forces occur when the object pose estimated by the object recognition algorithm is too imprecise such that the robot collides with the object as it reaches for it. However, since the robot quickly detects the collision it is able to stop before damaging the object. After that, the robot moves to a well-defined position and restarts the recognition process (as depicted in Fig. 4).

In order to demonstrate the importance of impedance control and collision detection for a safe object manipulation, we conducted a series of object grasping and placing experiments with different impedances and with distinct collision behavior.

Fig. 5 and 6 illustrate the robot behavior while grasping and placing an object using different stiffness values and tip velocities with and without collision detection. The plots depict the z-coordinates of the desired (dashed curves) and measured (continuous curves) contact force F_{ext_z}, position z and velocity $\dot{z}$. In both figures, the left columns are obtained for a high stiffness value while the right ones for a low value. The red curves indicate a lower tip velocity (about 0.1 m/s) while the blue ones a higher velocity (about 0.8 m/s). The upper row is obtained without collision detection and reaction, while the lower one visualizes the binary collision detection signal, $hf1$, together with a respective response which obviously leads to different curves.

If collision detection and handling is activated, a contact is classified as a collision if the magnitude of the contact force exceeds a certain limit. According to the figures, a high stiffness always leads to very high contact forces which might destroy the object to be handled. In contrast, low stiffness and collision detection contribute to a significant reduction of contact forces and practically eliminate the risk of damaging the object if a planned grasp is misaligned.

Fig. 7 shows an image sequence of grasping a bottle with low stiffness. We intentionally degraded the pose estimation accuracy such that the end effector collided with the bottle. Nevertheless, because of the impedance control, the end effector was able to adjust and successfully grasped the bottle.

Extension 1 exemplary shows some of the experiments described in this section.

6 Experimental Results

In this Section, we experimentally validate the proposed geometry matching algorithm (Section 6.1), the impedance controlled grasping strategy (Section 6.2) and the grasping capabilities of the overall system (Section 6.3). The object models involved in the tests are shown in Fig. 8.

The algorithm presented in this paper is implemented in C++ and all tests were performed on a Linux PC with 4GB RAM and an Intel Xeon 2.67GHz CPU with four cores.

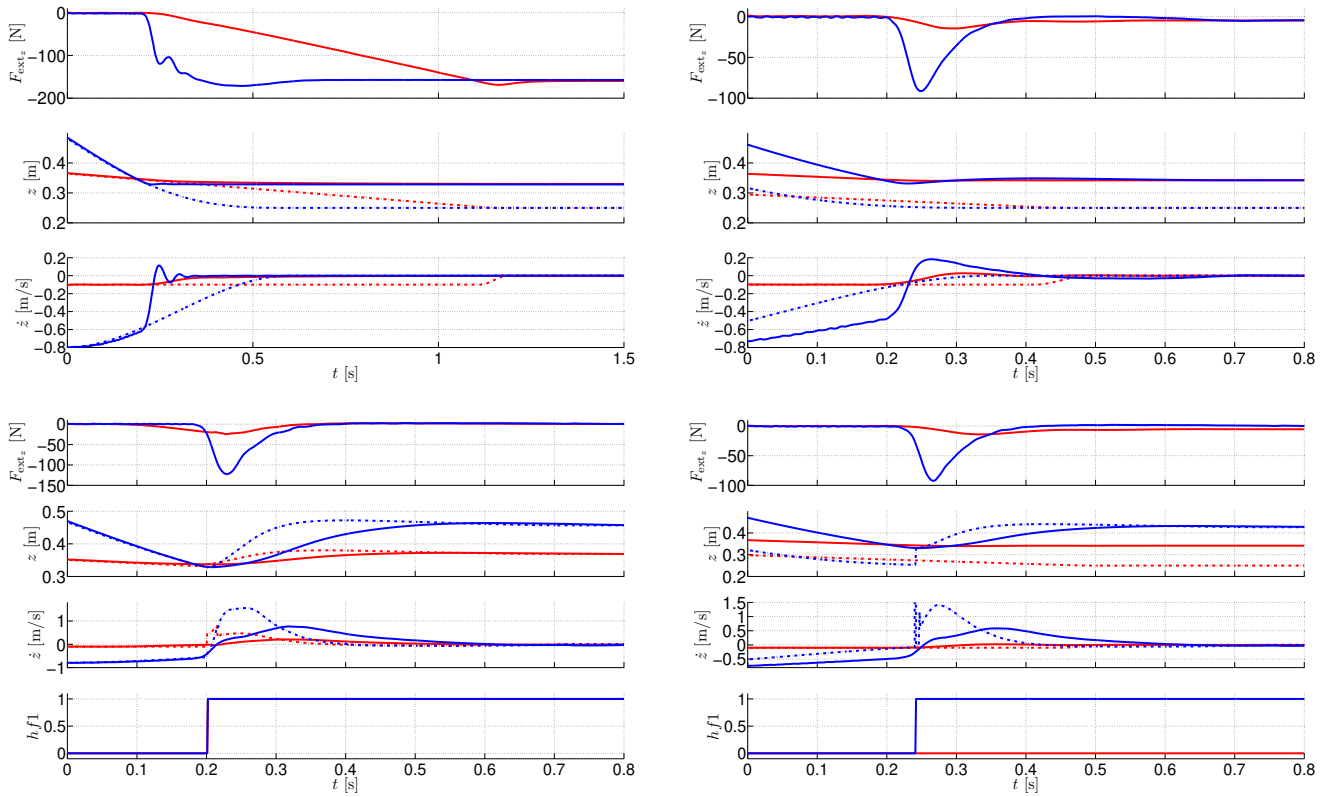


Fig. 5. Grasping behavior with high stiffness (left) and low stiffness (right) without (top) and with (bottom) collision detection and reaction.

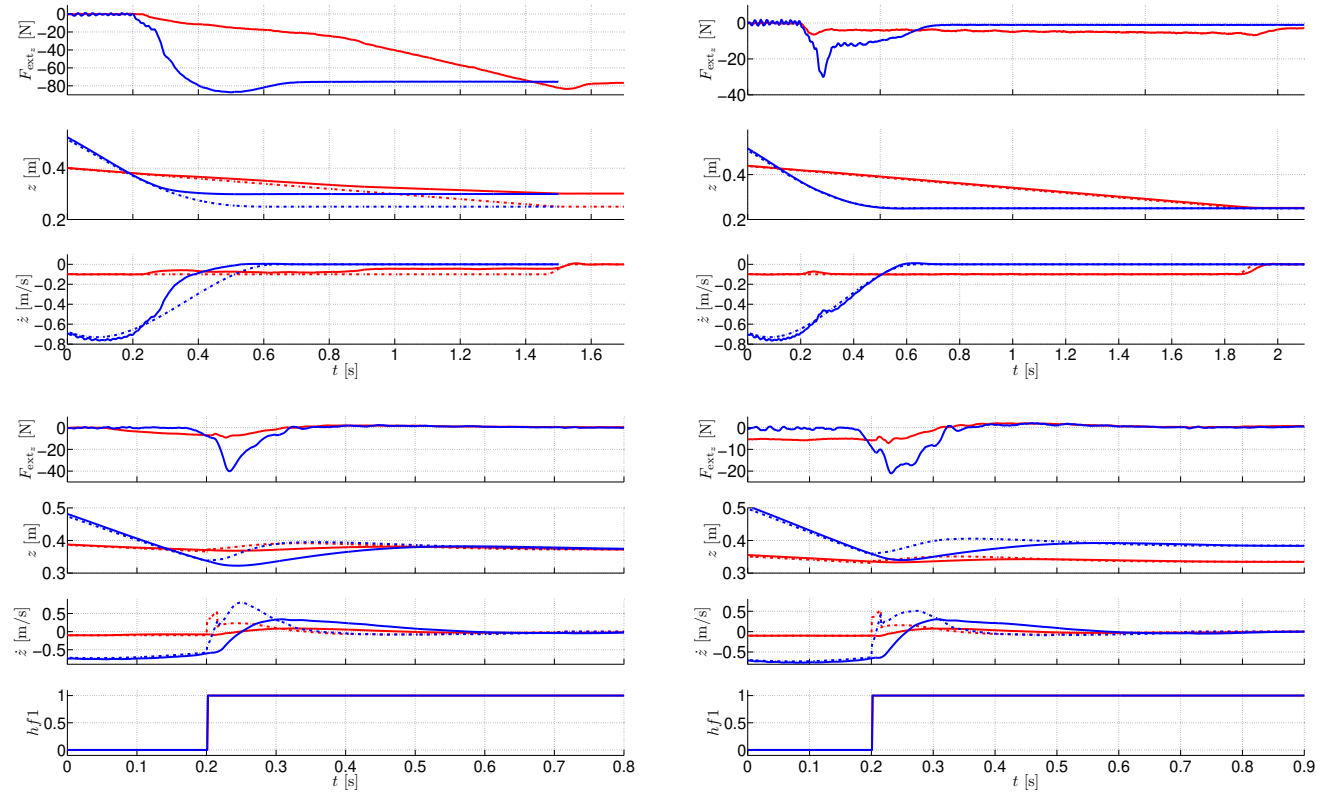


Fig. 6. Placing behavior with high stiffness (left) and low stiffness (right) without (top) and with (bottom) collision detection and reaction.

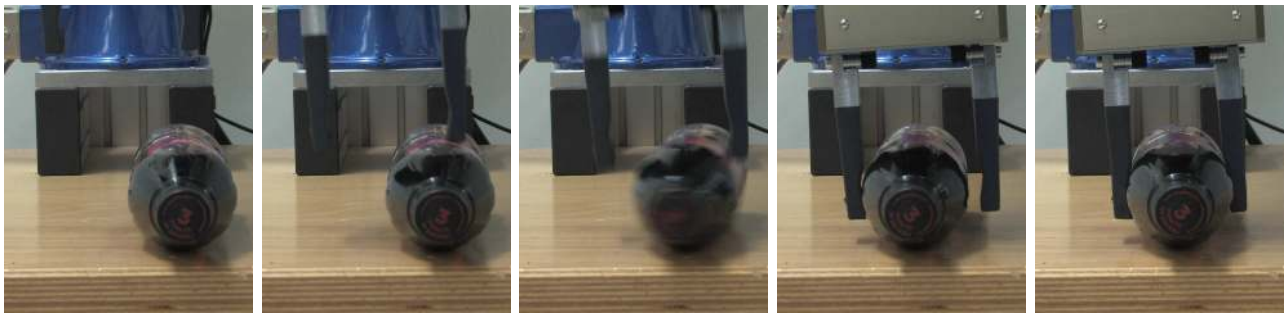


Fig. 7. Grasping behavior with low stiffness. Note that despite the imprecise object pose the robot is able to grasp the bottle.

Fig. 8. (a) The models used in the tests of the geometric matching algorithm (models provided by [Mian et al., 2006]). (b) The models involved in the grasping test scenarios (our own scans made with a low-cost light intersection-based scanning device).

6.1 Rigid 3D Geometry Matching

The scenes used in the following test cases were digitized with a Minolta VIVID 910 scanner [Konica Minolta, 2011] and provided by [Mian et al., 2006]. Examples and more information about the scenes will be given in the following subsections.

6.1.1 Matching a Single Object in Occluded Scenes

In the first test scenario, we examined how the success rate and the false positives rate of the geometry matching algorithm depend on the most important parameter, namely, the visibility threshold (introduced in Section 4.2) and the actual object occlusion in a scene. According to [Johnson and Hebert, 1999], the occlusion of an object model is given by

$$occlusion = 1 - \frac{\text{visible model surface area}}{\text{total model surface area}}. \quad (15)$$

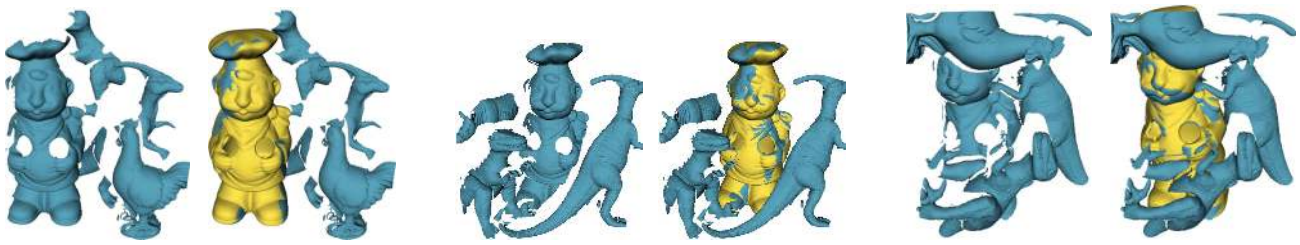
The aim of this test was to establish a value for the visibility threshold which, on the one hand, results in a high success rate even in highly occluded scenes, and on the other hand leads to as few as possible false positives. In this experiment, only the model of the Chef (Fig. 8(a)) was used for matching in three different scenes each one containing a total of three or four objects. The Chef was present in each scene at different locations and at different levels of occlusion (self-occlusion as well as occlusion caused by the other objects). The three test scenes together with typical recognition results are shown in Fig. 9.

Since the matching algorithm is a probabilistic one, we ran 100 trials on each scene and computed the recognition (success) rate and the mean number of false positives in the following way. We visually inspected the result of each trial. If object $\mathbf{A}$ (in this case only the Chef) was recognized k times ($0 \leq k \leq 100$), then the recognition rate for $\mathbf{A}$ is $k/100$. The mean number of false positives is $(k_1 + ... + k_{100})/100$, where k_i is the number of false alarms in the i-th trial.

The results of the test are summarized in Fig. 10. As expected, the visibility threshold had to fall below a certain value, namely, $1 - occlusion$ in order to achieve a positive recognition rate. More importantly, the plots suggest that the number of false positives practically does not depend on the actual level of occlusion but mainly on the visibility threshold: in all three cases it starts to grow when the visibility threshold falls below 0.15. In summary, it can be said that the method achieved a recognition rate of 1.0 in highly occluded scenes (up to 85% occlusion) at the cost of no false positives. In order to handle more occlusion the visibility threshold had to fall below 0.15 which gave rise to some false positives.

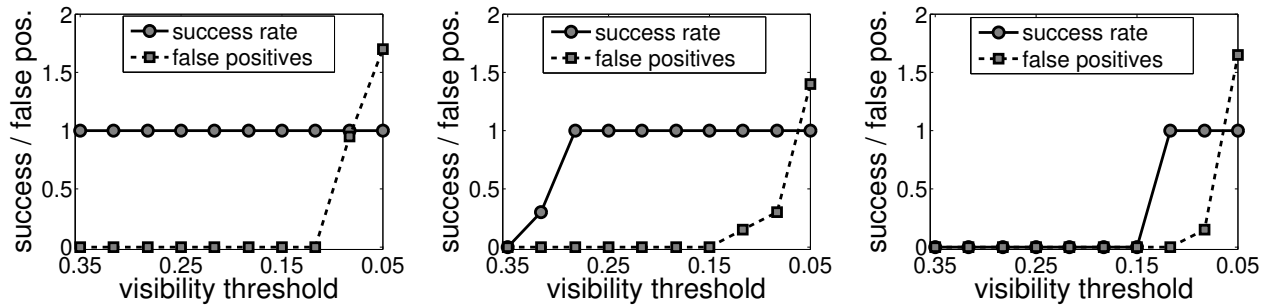
6.1.2 Matching Multiple Objects in Noisy Scenes

In this scenario, we tested the algorithm under varying noisy conditions. The four models involved are shown in Fig. 8(a) and the noise-free scene is shown in



(a) 62.3% object occlusion (b) 70.4% object occlusion (c) 86.2% object occlusion

Fig. 9. The test scenes used for the Chef model matching. The level of occlusion for the Chef is indicated for each scene. On the left of each subfigure, the input scene is shown as a blue mesh, whereas on the right, the recognized Chef model is placed at the location computed by our algorithm and rendered as a yellow mesh.



(a) 62.3% object occlusion (b) 70.4% object occlusion (c) 86.2% object occlusion

Fig. 10. The success rate and the mean number of false positives as functions of the visibility threshold for three different scenes each one containing the Chef model at an occlusion level of (a) 62.3%, (b) 70.4% and (c) 86.2%.

(a) Noise-free (b) $\sigma^2 = 0.25$ (c) $\sigma^2 = 1.0$

Fig. 11. (a) Noise-free scene. (b), (c) Typical recognition results for data sets degraded by zero-mean Gaussian noise for different variance σ^2 which is given as percentage of the bounding box diagonal length of the scene. On the left side of each subfigure, only the noise-corrupted scene is shown, whereas the right side shows the scene plus the recognized models placed at the locations estimated by the matching algorithm.

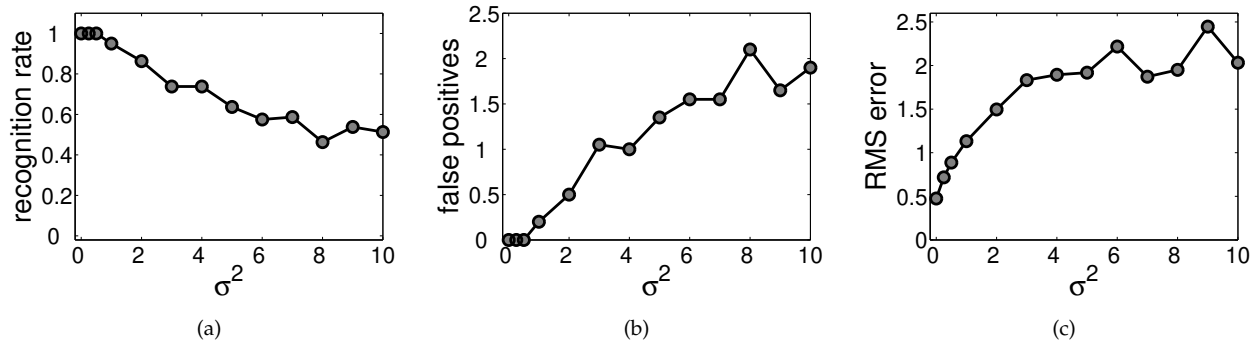
Fig. 11(a). We degraded the noise-free scene with zero-mean Gaussian noise with different variance values σ^2. Again, 100 recognition trials on each noisy scene were performed and the recognition rate, the mean number of false positives and the Root Mean Square error (RMSe) were computed as functions of σ^2.

For two point sets $\mathbf{P}$ and $\mathbf{Q}$ and a transform T the RMS error measures how close each point $\mathbf{q}_i \in \mathbf{Q}$ comes to its corresponding point $\mathbf{p}_i \in \mathbf{P}$ after transforming $\mathbf{Q}$ by T [Gelfand et al., 2005]. The smaller the error the closer the alignment between the point sets. More formally,

$$\mathrm{RMSe}(\mathbf{P}, \mathbf{Q}, T) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \|\mathbf{p}_i - T(\mathbf{q}_i)\|^2}, \qquad (16)$$

with N being the number of points. Since the ground truth location of each model in the test scene is known, the RMS error of the rigid transform computed by our method was easily calculated[2]. Note that we did *not* compute the RMS error for the transformed model and the scene but for the transformed model and the same model placed at the ground truth location. Fig. 11(b) and (c) exemplary show typical recognition results for two of the twelve noisy scenes. The results of the tests are reported in Fig. 12.

2. The ground truth rigid transform for the models is available on http://www.csse.uwa.edu.au/~ajmal/recognition.html



(a) (b) (c)

Fig. 12. (a) Recognition rate, (b) mean number of false positives and (c) RMS error as functions of the variance σ^2 of Gaussian noise. Note that the RMS error is computed for the successful trials only. Both σ^2 and the RMS error are given as percentage of the bounding box diagonal length of the scene.

6.1.3 Comparison with Spin Images and Tensor Matching

Next, we compared the recognition rate of our geometry matching algorithm with the spin images [Johnson and Hebert, 1999] and the tensor matching [Mian et al., 2006] approaches on the same 50 data sets used in [Mian et al., 2006]. This made a direct comparison possible without the need of re-implementing either of the two algorithms. The models of the four toys involved in the tests are shown in Fig. 8(a). The toys (not necessarily all four of them) are present in the scenes in different positions and orientations. Since each scene was digitized with a laser range finder from a single viewpoint the back parts of the objects were not visible. Furthermore, the toys were usually placed such that some of them occluded others which made the visible object parts even smaller. Four (out of the 50) test scenes are shown in Fig. 9 and Fig. 11(a). Again, we ran 100 recognition trials on each scene and computed the recognition rate for each object in the way described in Section 6.1.1. Since the occlusion of every object in the test scenes was known we report the recognition rate for each object as a function of its occlusion. The result of the comparison is summarized in Fig. 13(a). Note that the chef was recognized in all trials, even in the case of occlusion over 91%. The blue dots represent the recognition rate in the three chicken test scenes in which our method performed worse than the other algorithms. This was due to the fact that in these scenes only the chicken's back part was visible which contains strongly varying normals which made it difficult to compute a stable aligning transform.

Our method needed in average about 7.5 seconds for the recognition of the objects in each scene and sampled about 450 oriented point pairs per scene. For a comparison, 250 tensors, respectively, 4000 spin images per scene were used in the experiments performed in [Mian et al., 2006].

6.1.4 Runtime

We experimentally validated the linear time complexity of the matching algorithm in the number of scene points. Eleven different data sets were involved in this test case — a subset from the scenes used in the comparison test case (Section 6.1.3). Note that we did *not* take a single data set and down/up-sampled it to get the desired number of points. Instead, we chose eleven *different* scenes with varying scene extent, number of points and number of objects. This suggests that the results will hold for arbitrary scenes. We report the results of this test in Fig. 13(b).

Note that the iterations of the main loop of the matching algorithm (lines 4 to 20 of Algorithm 1) can be executed independently of each other which makes it possible to run them in parallel. This is a very important issue since parallel computing has become the dominant paradigm in computing architectures, mainly in the form of multicore processors [Asanovic et al., 2006]. In Fig. 13(c), we report the processing time as a function of the used CPU cores. Note that the parallel execution on four cores runs more than three times faster than on a single core. This indicates a great potential for further speed-up when more CPU cores become available.

6.2 "Blind" Grasping with an Impedance Controlled Robot

In contrast to Section 5.2, where the importance of impedance control and collision detection for a safe object manipulation was demonstrated, in this section, we conducted a series of grasping experiments with the aim to find a set of impedance parameters that maximizes the grasping success in the presence of simulated object pose errors. The robot altered its Cartesian translation stiffness in y-direction (denoted by $K_{t,y}$) and its rotation stiffness in x-direction (denoted by $K_{r,x}$). These directions are the lateral compliance along the gripper motion and the rotation perpendicular to this. Due to the inherent structure of the gripper, they are the significant parameters

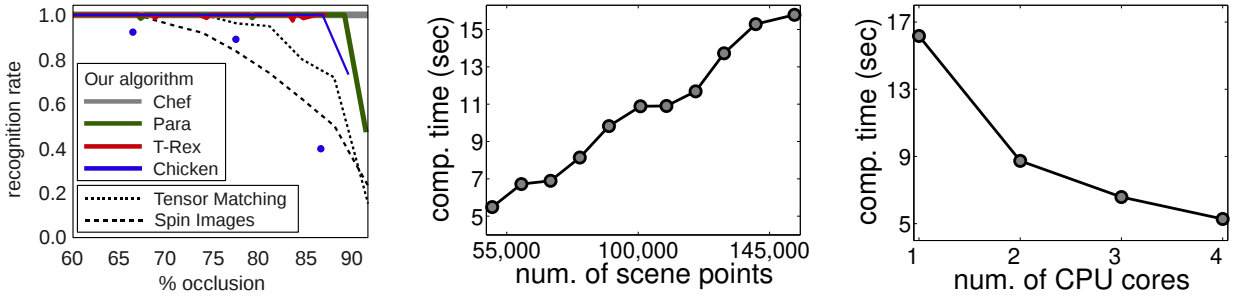


Fig. 13. (a) Comparison with spin images [Johnson and Hebert, 1999] and tensor matching [Mian et al., 2006]. The recognition rate of our algorithm for each object as a function of its occlusion is indicated by the continuous lines. The dashed lines give the recognition rate of the spin images and the tensor matching approaches on the same scenes as reported in [Mian et al., 2006]. Note that our algorithm outperforms both other methods. (b) Computation time as a function of the number of scene points for the simultaneous recognition of seven models. The line indicates a linear complexity. (c) Runtime as a function of the number of used CPU cores for the recognition of seven models in a scene consisting of around 60,000 points.

	$K_{t,y}$[N/m]	$K_{r,x}$ [Nm/rad]	success [%]
1	200	20	60
2	200	75	80
3	200	200	90
4	750	20	70
5	750	75	80
6	750	200	40
7	2000	20	50
8	2000	75	60
9	2000	200	70

TABLE 1
Grasping success with varying stiffness for a translational object pose error of 1.5 cm.

Fig. 14. The setup of the vision-based grasping experiments. The robot has grasped a green soda club bottle and is about to put it in the further red bin. The Kinect sensor can be seen in the upper right corner. In the lower left corner, the range image and the recognized models are shown (before the bottle has been taken away) from a viewpoint close to the one of the sensor.

governing the grasping process. The object involved in this test scenario was the soda club bottle (Fig. 8(b)). The object pose error was simulated by translating the bottle by 1.5 cm in a random direction parallel to the table in front of the robot. We performed ten grasping trials for each of the following stiffness configurations: "soft", "moderately stiff" and "rigid". The success rate is listed in Table 1. The optimal values (line 3) correspond to a soft (very compliant) translation and a rigid rotation behavior. A soft translation and a moderately stiff rotation (line 2) as well as a moderate stiffness in both translation and rotation (line 5) led to good success rates too.

6.3 Vision-Based Impedance Controlled Grasping

In this Section, we experimentally validate the overall vision-based impedance controlled grasping system. The models involved in these tests are shown in Fig. 8(b). We started with grasping single standing objects, moved on to object grasping from an unsorted pile and finished with a more complex task of cleaning up a table. We used the 7-degrees-of-freedom Cartesian impedance controlled DLR Lightweight robot III developed at the German Aerospace Center (DLR). It was mounted on a table

	object		
test scenario	Soda Club	Amicelli	Rusk
single standing objects	100%	95%	95%
object pile	95%	95%	90%

TABLE 2
Success rates in the grasping experiments.

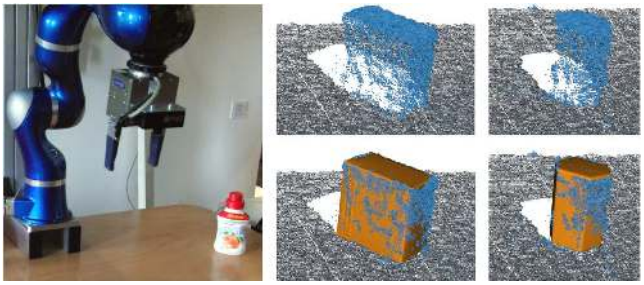


Fig. 15. (Left) The single standing object grasping scenario. (Middle, Right) two input range images (top) and the recognition results (bottom) for the rusk and the Amicelli box, respectively. The points off the plane (used for matching) are shown in light blue.

and covered an area of approximately 2.5 square meters. The scene was digitized with a Kinect sensor [Kinect for Xbox 360, 2011]. Since all objects were standing on or above the table, its plane was detected in each range image (using a simple RANSAC procedure) and all points belonging to the plane or lying below were removed. The setup is shown in Fig. 14.

6.3.1 Grasping Single Standing Objects

In the first scenario, multiple grasps were performed on single standing objects (see Fig. 15). We varied the pose of the objects such that all pre-saved grasp poses were executed. A grasp trial was considered successful if the object was correctly recognized, grasped and carried to the right place (table corner for the rusk box or one of the red bins for the Amicelli box/soda club bottle). We ran ten trials for each object pose and recorded the number of successful trials. The results are summarized in the first row of Table 2. One grasp failed for the Amicelli and the rusk box, respectively. This was due to the fact that the alignment computed by the matching algorithm was too imprecise.

6.3.2 Grasping from an Object Pile

Next, the robot performed multiple grasps on a pile consisting of seven objects placed next and on top of each other. Again, we changed the positions of the items such that the robot tried all pre-saved grasp poses for each object. A grasp was considered successful if the robot picked a correctly recognized object and carried it to the right place. After an object had been taken away, we built up a new pile, i.e., the robot had to deal every time with a full pile. This experiment added some additional difficulties to both the geometry matching and the robot control algorithms. Obviously, the risk of recognition failures increased since there were more objects in the scene. Besides that, objects in a pile are in a more unstable configuration (from a statics point of view) compared to single standing ones. This made it more difficult for the impedance-based control to compensate for matching imprecision. We performed ten trials for each grasp pose and recorded the number of successful trials. The results are compiled in the second row of Table 2. As to be expected, the failure rate increased compared to the first grasping experiment.

6.3.3 Cleaning up the Table

In the last test scenario, we let the robot repeatedly perform a more complex task, namely, cleaning up the table in front of it. Seven objects were randomly placed on a pile which resulted in highly cluttered and occluded scenes. The task to the robot was to pick each object, put it away and halt when it "believes" that the table is empty. The recognition process was restarted each time an object was carried away. Thus, the robot had to deal with the changing scene and with unforeseen situations which happened during the cleanup like, e.g., an object falling off the pile. The task was accomplished if at the end each object was at the place designated for it. This time we did not consider it a failure when an object slipped out of the gripper as long as it was picked up later on and left in the right place. After each cleanup trial we built up a new pile and let the robot perform the task again. We repeated this 15 times and counted the number of successful trials. The robot achieved a success rate of 80%. Fig. 16 exemplary shows one cleanup process. More examples can be seen in Extension 2.

7 Conclusions and Future Work

In this paper, we introduced a vision-based approach for grasping of known objects. Our approach relies on a robust 3D geometry matching algorithm for the recognition and localization of multiple objects in noisy, partially reconstructed and unsegmented scenes. The matching algorithm is based on a robust geometric descriptor, a hashing technique and an efficient, localized RANSAC-like sampling strategy. We provided a theoretical complexity analysis and derived a formula for computing the number of iterations required to recognize the objects with a predefined success probability. The result of the complexity analysis, namely a linear time dependency on the number of scene points, was experimentally validated. Tests on real range data confirmed that our method performs well on complex scenes in which only small parts of the objects are visible. In a direct comparison with the spin images [Johnson and Hebert, 1999] and the tensor matching [Mian et al., 2006] approaches, our method performed better in terms of recognition rate. A further

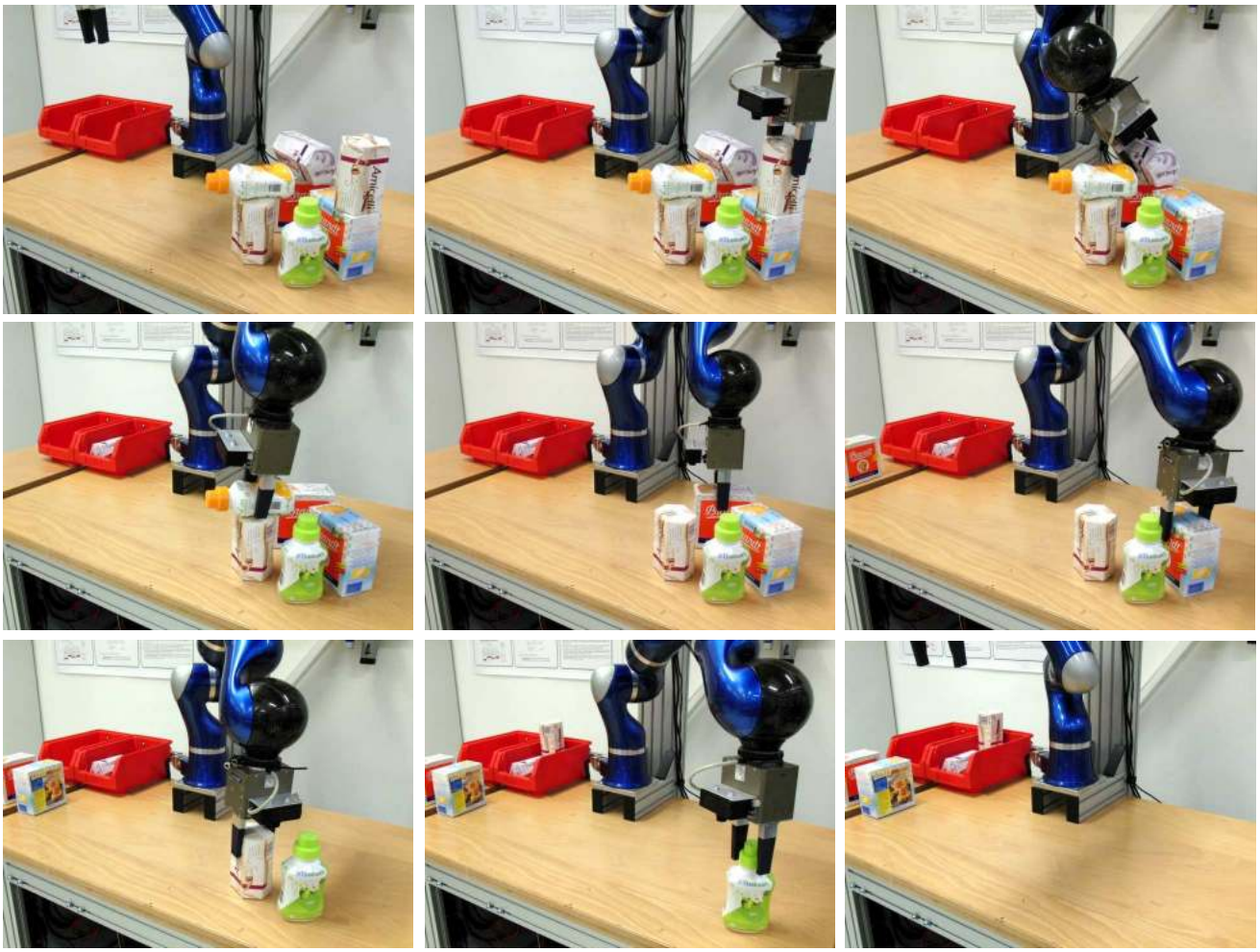


Fig. 16. (Left to right, top to bottom) A sequence of images showing the robot cleaning up the table. The first and the last image show the beginning and the end of the task, respectively. Each in-between shot shows the robot in the moment of grasping an object. Note that the first Amicelli box and all bottles are placed in a lying orientation in the red bins and are not visible from this point of view.

experimental validation with the DLR Lightweight-Robot III showed how well this new method can be exploited for grasping in unstructured and cluttered environments. The presented solution is capable of quickly recognizing and robustly grasping known objects from an unsorted pile of different everyday items. This is extremely useful for typical service robotics or industrial co-worker tasks.

One possible extension of the recognition algorithm would be to incorporate higher dimensional geometric descriptors. We expect this to lead to more uniformly spread model point pairs in the feature space and to further reduce the overpopulation of hash table cells mentioned in Section 4.1. Furthermore, in the current implementation of the recognition method, the pair width d is selected manually based on the extent of the object models stored in the hash table and on the expected degree of object occlusion in the scene. An elaborate way of automatically selecting the right value for d is the current topic of our research.

Acknowledgments

We would like to thank Tim Rokahr for his help. This work has been partially funded by the European Commission's Seventh Framework Programme as part of the projects GRASP and SAPHARI.

Appendix A: Index to Multimedia Extensions

Extension	Media Type	Description
1	Video	Impedance control and collision detection for sensitive grasping and placing.
2	Video	Cleaning up a table full of grocery items.

References

[Aiger et al., 2008] Aiger, D., Mitra, N. J., and Cohen-Or, D. (2008). 4-points Congruent Sets for Robust Pairwise Surface Registration. *ACM Trans. Graph.*, 27(3).

[Albu-Schäffer et al., 2007] Albu-Schäffer, A., Haddadin, S., Ott, C., Stemmer, A., Wimböck, T., and Hirzinger, G. (2007). The DLR lightweight robot - lightweight design and soft robotics control concepts for robots in human environments. *Industrial Robot Journal*, 34(5):376–385.

[Asanovic et al., 2006] Asanovic, K., Bodik, R., Catanzaro, B. C., Gebis, J. J., Husbands, P., Keutzer, K., Patterson, D. A., Plishker, W. L., Shalf, J., Williams, S. W., and Yelick, K. A. (2006). The Landscape of Parallel Computing Research: A View from Berkeley. Technical report, EECS Department, University of California, Berkeley.

[Ballard, 1981] Ballard, D. H. (1981). Generalizing the Hough Transform to Detect Arbitrary Shapes. *Pattern Recognition*, 13(2):111–122.

[Barr, 1981] Barr, A. (1981). Superquadrics and Angle-Preserving Transformations. *Computer Graphics and Applications*, 1(1):11 –23.

[Biegelbauer et al., 2010] Biegelbauer, G., Vincze, M., and Wohlkinger, W. (2010). Model-based 3D object detection. *Mach. Vis. Appl.*, 21(4):497–516.

[Binford, 1971] Binford, T. (1971). Visual Perception by a Computer. In *IEEE Conf. on Systems and Control*.

[de Berg et al., 2000] de Berg, M., van Kreveld, M., Overmars, M., and Schwarzkopf, O. (2000). *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2 edition.

[De Luca et al., 2006] De Luca, A., Albu-Schäffer, A., Haddadin, S., and Hirzinger, G. (2006). Collision detection and safe reaction with the DLR-III lightweight manipulator arm. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS2006), Beijing, China*, pages 1623–1630.

[Dickinson et al., 1997] Dickinson, S. J., Metaxas, D. N., and Pentland, A. (1997). The Role of Model-Based Segmentation in the Recovery of Volumetric Parts From Range Data. *IEEE TPAMI*, 19(3):259–267.

[Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395.

[Frome et al., 2004] Frome, A., Huber, D., Kolluri, R., Bülow, T., and Malik, J. (2004). Recognizing Objects in Range Data Using Regional Point Descriptors. In *ECCV*, pages 224–237.

[Fuchs et al., 2010] Fuchs, S., Haddadin, S., Parusel, S., Keller, M., and Kolb, A. (2010). Cooperative bin-picking with time-of-flight camera and impedance controlled DLR Lightweight robot III. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS2008)*, pages 4862–4867.

[Gelfand et al., 2005] Gelfand, N., Mitra, N., Guibas, L., and Pottmann, H. (2005). Robust Global Registration. In *Eurographics Symposium on Geometry Processing*, pages 197–206.

[Grewe and Kak, 1995] Grewe, L. and Kak, A. C. (1995). Interactive Learning of a Multiple-Attribute Hash Table Classifier for Fast Object Recognition. *Computer Vision and Image Understanding*, 61(3):387–416.

[Haddadin et al., 2008] Haddadin, S., Albu-Schäffer, A., Luca, A. D., and Hirzinger, G. (2008). Collision detection & reaction: A contribution to safe physical human-robot interaction. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS2008), Nice, France*, pages 3356–3363.

[Haddadin et al., 2009] Haddadin, S., Suppa, M., Fuchs, S., Bodenmüller, T., Albu-Schäffer, A., and Hirzinger, G. (2009). Towards the robotic co-worker. In *International Symposium on Robotics Research (ISRR2009), Lucerne, Switzerland*.

[Hetzel et al., 2001] Hetzel, G., Leibe, B., Levi, P., and Schiele, B. (2001). 3D Object Recognition from Range Images Using Local Feature Histograms. In *CVPR*, pages 394–399.

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

[Keren et al., 1994] Keren, D., Cooper, D. B., and Subrahmonia, J. (1994). Describing Complicated Objects by Implicit Polynomials. *IEEE TPAMI*, 16(1):38–53.

[Kinect for Xbox 360, 2011] Kinect for Xbox 360 (2011). http://www.xbox.com/en-US/kinect. Accessed: 20/04/2011.

[Konica Minolta, 2011] Konica Minolta (2011). Minolta VIVID 910. http://www.konicaminolta.com/instruments/products/3d/non-contact/vivid910/index.html. Accessed: 18/09/2011.

[Lamdan and Wolfson, 1988] Lamdan, Y. and Wolfson, H. (1988). Geometric Hashing: A General and Efficient Model-Based Recognition Scheme. In *ICCV*, pages 238–249.

[Matei et al., 2006] Matei, B., Shan, Y., Sawhney, H. S., Tan, Y., Kumar, R., Huber, D. F., and Hebert, M. (2006). Rapid Object Indexing Using Locality Sensitive Hashing and Joint 3D-Signature Space Estimation. *IEEE TPAMI*, 28(7):1111–1126.

[Mian et al., 2006] Mian, A. S., Bennamoun, M., and Owens, R. A. (2006). Three-Dimensional Model-Based Object Recognition and Segmentation in Cluttered Scenes. *IEEE TPAMI*, 28(10):1584–1601.

[Papazov and Burschka, 2010] Papazov, C. and Burschka, D. (2010). An Efficient RANSAC for 3D Object Recognition in Noisy and Occluded Scenes. In *Asian Conference on Computer Vision (ACCV'10)*, pages 135–148.

[Parusel et al., 2011] Parusel, S., Haddadin, S., and Albu-Schäffer, A. (2011). Modular state-based behavior control for safe human-robot interaction: A lightweight control architecture for a lightweight robot. In *IEEE Int. Conf. on Robotics and Automation (IROS2011), Shanghai, China*.

[Schnabel et al., 2007] Schnabel, R., Wahl, R., and Klein, R. (2007). Efficient RANSAC for Point-Cloud Shape Detection. *Comput. Graph. Forum*, 26(2):214–226.

[Solina and Bajcsy, 1990] Solina, F. and Bajcsy, R. (1990). Recovery of Parametric Models from Range Images: The Case for Superquadrics with Global Deformations. *IEEE TPAMI*, 12(2):131–147.

[Stockman, 1987] Stockman, G. (1987). Object Recognition and Localization via Pose Clustering. *Computer Vision, Graphics, and Image Processing*, 40(3):361–387.

[Sun et al., 2009] Sun, J., Ovsjanikov, M., and Guibas, L. J. (2009). A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion. *Comput. Graph. Forum*, 28(5):1383–1392.

[Taylor and Kleeman, 2003] Taylor, G. and Kleeman, L. (2003). Robust Range Data Segmentation using Geometric Primitives for Robotic Applications. In *SIP*, pages 467–472.

[Winkelbach et al., 2006] Winkelbach, S., Molkenstruck, S., and Wahl, F. M. (2006). Low-Cost Laser Range Scanner and Fast Surface Registration Approach. In *Proceedings of the 28th DAGM Symposium on Pattern Recognition*, pages 718–728.

[Wu et al., 2010] Wu, H.-Y., Zha, H., Luo, T., Wang, X., and Ma, S. (2010). Global and Local Isometry-Invariant Descriptor for 3D Shape Comparison and Partial Matching. In *CVPR*, pages 438–445.