Model Globally, Match Locally: Efficient and Robust 3D Object Recognition

Bertram Drost[1], Markus Ulrich[1], Nassir Navab[2], Slobodan Ilic[2]

[1]MVTec Software GmbH
Neherstraße 1, 81675 Munich, Germany
[2]Department of Computer Science, Technical University of Munich (TUM)
Boltzmannstraße 3, 85748 Garching, Germany
{drost,ulrich}@mvtec.com, navab@cs.tum.edu, Slobodan.Ilic@in.tum.de

Abstract

This paper addresses the problem of recognizing free-form 3D objects in point clouds. Compared to traditional approaches based on point descriptors, which depend on local information around points, we propose a novel method that creates a global model description based on oriented point pair features and matches that model locally using a fast voting scheme. The global model description consists of all model point pair features and represents a mapping from the point pair feature space to the model, where similar features on the model are grouped together. Such representation allows using much sparser object and scene point clouds, resulting in very fast performance. Recognition is done locally using an efficient voting scheme on a reduced two-dimensional search space.

We demonstrate the efficiency of our approach and show its high recognition performance in the case of noise, clutter and partial occlusions. Compared to state of the art approaches we achieve better recognition rates, and demonstrate that with a slight or even no sacrifice of the recognition performance our method is much faster then the current state of the art approaches.

1. Introduction

The recognition of free-form objects in 3D data obtained by different sensors, such as laser scans, TOF cameras and stereo systems, has been widely studied in computer vision [2, 9, 12]. Global approaches [8, 13, 14, 18, 23, 25] are typically neither very precise nor fast, and are limited mainly to the classification and recognition of objects of certain type. By contrast, local approaches that are based on local invariant features [1, 4, 5, 6, 7, 10, 15, 17, 19, 20, 21, 24] became extremely popular and proved to be quite efficient. However, defining local invariant features heavily depends on local surface information that is directly related to the quality and resolution of the acquired and model data.

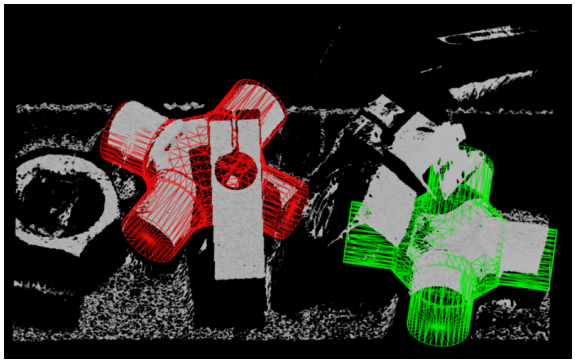


Figure 1. Example of two partly occluded instances of an object found in a noisy, cluttered scene. The matched objects are shown as red and green wireframe and might not be recognizeable in B/W copies.

In contrast to the approaches outlined above we propose a method that creates a global model description using an oriented point pair feature and matches it by using a fast voting scheme. The point pair feature describes the relative position and orientation of two oriented points as described in Sec. 3.1. The global model description consists of all model point pair features and represents a mapping from the feature space to the model, where similar features on the model are grouped together. Such a representation provides a global distribution of all point pair features on the model surface. Compared to the local methods, which require dense local information, our approach allows the model and the scene data to be represented only by a sparse set of oriented points that can easily be computed from the input data. Using sparse data also allows for an important increase in the recognition speed, without significant decrease in the recognition rate. A fast voting scheme, similar

to the Generalized Hough Transform, is used to optimize the model pose in a locally reduced search space which is parametrized in terms of points on the model and rotation around the surface normals.

We test our approach on a number of synthetic and real sequences and compare it to state-of-the-art approaches. We demonstrate that in the presence of noise, clutter, and occlusions we obtain better results than Spin Images [7] and Tensors [10] in terms of recognition rates and efficiency.

2. Related Work

The problem of detecting and recognizing free-form objects in three-dimensional point clouds is well studied. Methods for refining a coarse registration, such as ICP [26], are able to optimize a coarse registration. However, we are only interested in methods for object registration that do not need a coarse pose as input.

Several global methods for 3D object detection have been proposed. However, they either detect only shapes such as planes, cylinders and spheres, or require a segmentation of the scene. Wahl *et al*. [23] introduce an object identification scheme that identifies segmented free-form objects by computing the histogram of oriented point relations, called surflet pairs. The two-point feature used in our method is based on the idea of surflet pairs. Several approaches detect objects using a variant of the Generalized Hough Transform [8, 14, 25] but are limited to primitive objects as the recovery of a full 3D pose with 6 degrees of freedom is computationally too expensive. Schnabel *et al*. [18] detect primitives in point clouds by using an efficient variant of RANSAC. Park *et al*. [13] detect objects in range images by searching for patterns of the object created from multiple directions. They parallelize their algorithm on the GPU in order to obtain matching times of around 1 second. By contrast, our approach works with general 3D point clouds and is equally efficient without parallelization.

A second class of methods, local methods, usually use a pipeline that first identifies possible point to point correspondences between the model and the scene. Multiple correspondences are then grouped to recover the pose of the model. A typical way of finding the correspondences is the use of point descriptors that describe the surface around a certain point using a low-dimensional representation. The descriptors need to be as discriminating as possible while being invariant against a rigid movement of the surface, robust against clutter and noise and embedded in a framework that can deal with occlusion. Point correspondences are built by comparing the descriptors of the model to those of the scene. Extensive surveys over different descriptors are given in [2, 9, 12], which is why only a few selected ones are covered here.

Point descriptors can be categorized by the radius of influence that affects them. Local descriptors exploit the geometric properties around a surface point, most notably by using different representations of the surface curvature [1, 5] or by fitting polynomials [15] or splines [24]. Regional descriptors try to capture the surface shape around the reference point. Splashs [20] describe the distribution of normal orientations around a point. Point Signatures [4] use the distance of neighbouring points to the normal plane of the reference point. Point Fingerprints [21] use geodesic circles around the reference point as description. Gelfand *et al*. [6] introduced an integral descriptor for point cloud registration that describes the intersecting volume of a sphere around the reference point with the object. Rusu *et al*. [17] build a Point Feature Histogram of two-point descriptors for all neighbouring points of the reference point. Chen and Bhanu [3] introduced a local surface patch representation that is a histogram of shape index values vs. the angle between normals. Johnson and Hebert [7] introduced spin images, which are histograms of the surface created by rotating a half-plane around the normal of the reference point and summing the intersecting surface. In [19] the spin images were used as an index into a database of objects with subsequent pose detection using a batch RANSAC. Ruiz-Correa *et al*. [16] defined a symbolic surface signature to detect classes of similar objects.

Mian *et al*. [10] use two reference points to define a coordinate system where a three-dimensional Tensor is built by sampling the space and storing the amount of surface intersecting each sample. The Tensors are stored using a hash table that allows an efficient lookup during the matching phase. The Tensors can be used not only for matching, but also for general point cloud registration.

The problem with point descriptors is that they are often sensitive to occlusion, noise and local clutter. Since they describe the surface locally, it's hard for them to discriminate self-similar surface parts, such as planar patches, spheres and cylinders. Increasing the radius of influence increases the discrimination capabilities, but makes the descriptors more sensitive to missing surface parts and clutter. We will show that our approach has no problem with such self-similar objects. Also, we do not require a post-processing step such as RANSAC for grouping the correspondences. Finally, descriptor-based approaches require a dense surface representation for calculating the features, while our method efficiently matches objects in sparse point clouds. This allows our method to be much more efficient in terms of detection time than descriptor-based methods.

3. Model Description and Voting Scheme

We assume that both the scene and the model are represented as a finite set of oriented points, where a normal is associated with each point. Such representations can be easily computed from meshes or point clouds. We denote $\mathbf{s}_i \in S$ for points in the scene and $\mathbf{m}_i \in M$ for points in the

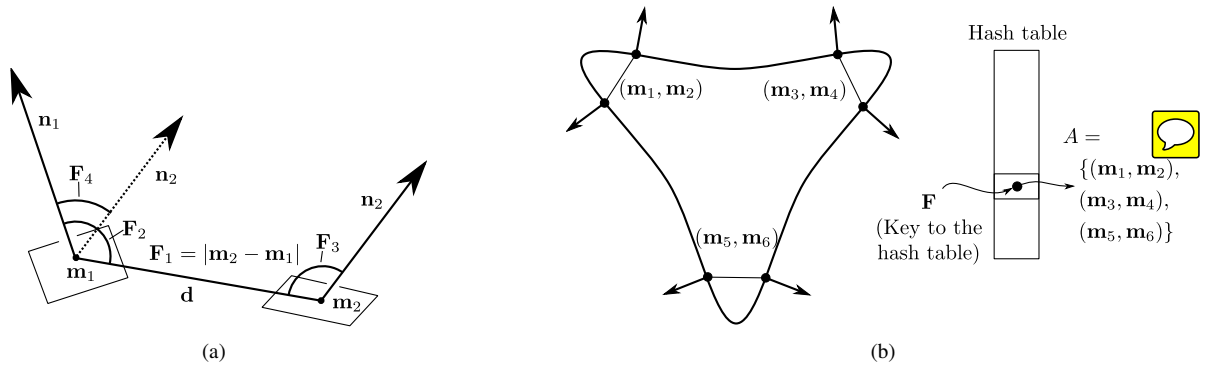


Figure 2. (a) Point pair feature $\mathbf{F}$ of two oriented points. The component $\mathbf{F}_1$ is set to the distance of the points, $\mathbf{F}_2$ and $\mathbf{F}_3$ to the angle between the normals and the vector defined by the two points, and $\mathbf{F}_4$ to the angle between the two normals. (b) The global model description. Left: Point pairs on the model surface with similar feature vector $\mathbf{F}$. Right: Those point pairs are stored in the same slot in the hash table.

model.

In the off-line phase the global model description is created. In the on-line phase, a set of reference points in the scene is selected. All other points in the scene are paired with the reference points to create point pair features. These features are matched to the model features contained in the global model description and a set of potential matches is retrieved. Every potential match votes for an object pose by using an efficient voting scheme where the pose is parametrized relative to the reference point. This voting scheme finally returns the optimal object pose.

We first introduce the point pair feature and then describe how the global model description is built using those point pair features. We then describe the voting scheme in detail.

3.1. Point Pair Feature

Our method uses a *point pair feature* that describes the relative position and orientation of two oriented points which is similar to the surflet-pair feature from [17, 23] and depicted in Fig. 2(a). For two points $\mathbf{m}_1$ and $\mathbf{m}_2$ with normals $\mathbf{n}_1$ and $\mathbf{n}_2$, we set $\mathbf{d} = \mathbf{m}_2 - \mathbf{m}_1$ and define the feature $\mathbf{F}$ as:

$$\mathbf{F}(\mathbf{m}_1, \mathbf{m}_2) = (\|\mathbf{d}\|_2, \angle(\mathbf{n}_1, \mathbf{d}), \angle(\mathbf{n}_2, \mathbf{d}), \angle(\mathbf{n}_1, \mathbf{n}_2)), \tag{1}$$

where $\angle(\mathbf{a}, \mathbf{b}) \in [0; \pi]$ denotes the angle between two vectors. Contrary to [23], our feature $\mathbf{F}$ is asymmetric. We use this feature to create the global model description and later for finding the object of interest in the scene.

3.2. Global Model Description

To build a global model description in the off-line phase, we use the point pair feature described above. The model is represented by a set of point pair features with similar feature vectors being grouped together. To do that, we calculate the feature vector $\mathbf{F}$ of Eq. (1) for all point pairs $\mathbf{m}_i, \mathbf{m}_j \in M$ on the model surface. The distances and the angles are sampled in steps of d_{dist} and $d_{angle} = 2\pi / n_{angle}$ respectively. Feature vectors whose discrete versions are equal are then grouped together.

The global model description is a mapping from the sampled point pair feature space to the model. Formally, we can write this mapping as $L : \mathbb{Z}^4 \rightarrow A \subset M^2$, where the four dimensional point pair features defined at Eq. 1 are mapped to set A of all pairs $(\mathbf{m}_i, \mathbf{m}_j) \in M^2$ that define an equal feature vector. To illustrate this, Fig 2(b) shows an example of point pairs with similar features on a single object. They are collected in set A. In practice, this model description is represented as a hash table indexed by the sampled feature $\mathbf{F}$. All model features $\mathbf{F}_m(\mathbf{m}_i, \mathbf{m}_j)$ that are similar to a given scene feature $\mathbf{F}_s(\mathbf{s}_i, \mathbf{s}_j)$ can then be searched in constant time by using $\mathbf{F}_s$ as a key to access the hash table.

3.3. Voting Scheme

Local Coordinates We consider an arbitrary reference point $\mathbf{s}_r \in S$ from the scene and assume that it lies on the object that we try to detect. If the assumption is correct then there is a point $\mathbf{m}_r \in M$ that corresponds to $\mathbf{s}_r$. After aligning those two points and their normals, the object can be rotated around the normal of $\mathbf{s}_r$ to align the model to the scene. This removes another degree of freedom for the pose of the model in the scene. The rigid motion from the model space into the scene space can thus be described by a point on the model and a rotation angle α. We call such a pair $(\mathbf{m}_r, \alpha)$ the *local coordinates* of the model with respect to reference point $\mathbf{s}_r$.

In our method, a point pair $(\mathbf{m}_r, \mathbf{m}_i) \in M^2$ is aligned to a scene pair $(\mathbf{s}_r, \mathbf{s}_i) \in S^2$ where both pairs have a similar feature vector $\mathbf{F}$. The transformation from the local model coordinates to the scene coordinates is defined by

$$\mathbf{s}_i = T_{s \rightarrow g}^{-1} R_{\mathbf{x}}(\alpha) T_{m \rightarrow g} \mathbf{m}_i \tag{2}$$

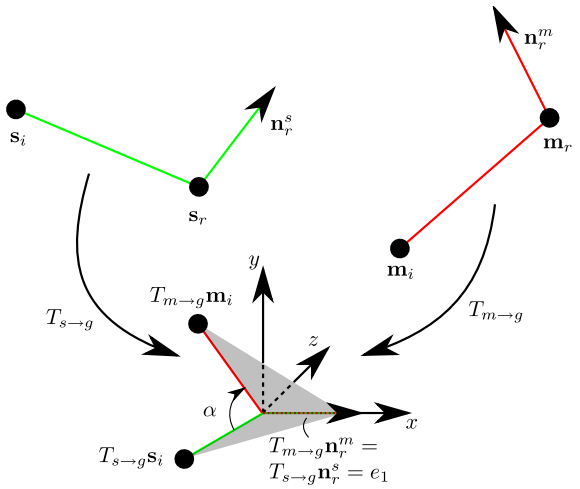


Figure 3. Transformation between model and scene coordinates. The two point pairs have a similar feature $\mathbf{F}$ and, thus, the same distance and relative orientation. The transformation $T_{m\to g}$ translates $\mathbf{m}_r$ into the origin and rotates its normal $\mathbf{n}_r^m$ onto the $\mathbf{x}$-axis. $T_{s\to g}$ does the same for the scene point pair. Since the images of $\mathbf{s}_i$ and $\mathbf{m}_i$ will be misaligned, the rotation $R_{\mathbf{x}}(\alpha)$ around the x-axis with angle α is required to match them.

and explained in Fig. 3. Note that local coordinates have three degrees of freedom (one for the rotation angle α and two for a point on the model surface), while a general rigid movement in 3D has six.

Voting Scheme Given a fixed reference point $\mathbf{s}_r$, we want to find the "optimal" local coordinates such that the number of points in the scene that lie on the model is maximized. This is done using a voting scheme, which is similar to the Generalized Hough Transform and very efficient since the local coordinates only have three degrees of freedom. Once the optimal local coordinates are found, the global pose of the object can be recovered.

For the voting scheme, a two-dimensional accumulator array is created. The number of rows, N_m, is equal to the number of model sample points $|M|$. The number of columns N_{angle} corresponds to the number of sample steps n_{angle} of the rotation angle α. This accumulator array represents the discrete space of local coordinates for a fixed reference point.

For the actual voting, reference point $\mathbf{s}_r$ is paired with every other point $\mathbf{s}_i \in S$ from the scene, and the model surface is searched for point pairs $(\mathbf{m}_r, \mathbf{m}_i)$ that have a similar distance and normal orientation as $(\mathbf{s}_r, \mathbf{s}_i)$. This search answers the question of where on the model the pair of scene points $(\mathbf{s}_r, \mathbf{s}_i)$ could be, and is performed using the precomputed model description: Feature $\mathbf{F}_s(\mathbf{s}_r, \mathbf{s}_i)$ is calculated and used as key to the hash table of the global model description, which returns the set of similar features on the model. For each match $(\mathbf{m}_r, \mathbf{m}_i)$, i.e. for every possible position of $(\mathbf{s}_r, \mathbf{s}_i)$ on the model surface, the rotation angle α is calculated using Eq. 2 that corresponds to the local coordinate that maps $(\mathbf{m}_r, \mathbf{m}_i)$ to $(\mathbf{s}_r, \mathbf{s}_i)$, as shown in Fig. 3. A vote is cast for the local coordinates $(\mathbf{m}_r, \alpha)$. Fig. 4 outlines the voting process.

After all points $\mathbf{s}_i$ are processed, the peak in the accumulator array corresponds to the optimal local coordinate, from which a global rigid movement can be calculated. For stability reasons, all peaks that received a certain amount of votes relative to the maximum peak are used.

Efficient Voting Loop To obtain a highly efficient object detection algorithm, we will now show how the above voting scheme can be implemented efficiently.

To speed up solving Eq. 2 for every point pair in the list, we split α into two parts, $\alpha = \alpha_m - \alpha_s$, such that α_m and α_s depend only on the point pair on the model and scene respectively. We split $R_{\mathbf{x}}(\alpha) = R_{\mathbf{x}}(-\alpha_s)R_{\mathbf{x}}(\alpha_m)$ and use $R_{\mathbf{x}}^{-1}(-\alpha_s) = R_{\mathbf{x}}(\alpha_s)$ to obtain

$$\begin{aligned}\mathbf{t} &= R_{\mathbf{x}}(\alpha_s)T_{s\to g}\mathbf{s}_i = \\ &= R_{\mathbf{x}}(\alpha_m)T_{m\to g}\mathbf{m}_i \in \mathbb{R}\mathbf{x} + \mathbb{R}_0^+\mathbf{y},\end{aligned} \tag{3}$$

i.e. $\mathbf{t}$ lies on the half-plane defined by the x-axis and the non-negative part of the y-axis. For each point pair in the model or in the scene, $\mathbf{t}$ is unique. α_m can thus be precalculated for every model point pair in the off-line phase and is stored in the model descriptor. α_s needs to be calculated only once for every scene point pair $(\mathbf{s}_r, \mathbf{s}_i)$, and the final angle α is a simple difference of the two values.

3.4. Pose Clustering

The above voting scheme identifies the object pose if the reference point lies on the surface of the object. Therefore, multiple reference points are necessary to ensure that one of them lies on the searched object. As shown in the previous section, each reference point returns a set of possible object poses that correspond to peaks in its accumulator array. The retrieved poses will only approximate the ground truth due to different sampling rates of the scene and the model and the sampling of the rotation in the local coordinates. We now introduce an additional step that both filters out incorrect poses and increases the accuracy of the final result.

To this end, the retrieved poses are clustered such that all poses in one cluster do not differ in translation and rotation for more than a predefined threshold. The score of a cluster is the sum of the scores of the contained poses, and the score of a pose is the number of votes it recived in the voting scheme. After finding the cluster with the maximum score, the resulting pose is calculated by averaging the poses contained in the cluster. Since multiple instances of the object can be in the scene, several clusters can be returned by the

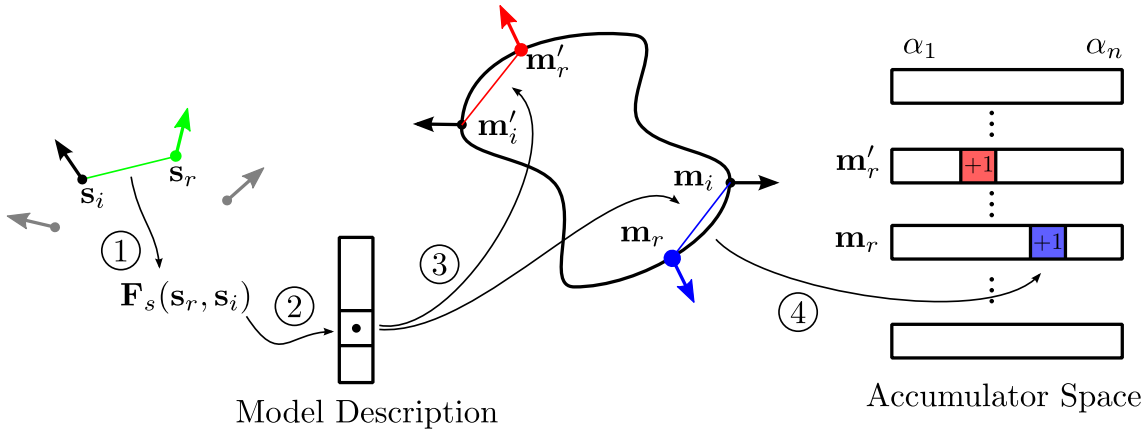


Figure 4. Visualisation of different steps in the voting scheme: (1) Reference point $\mathbf{s}_r$ is paired with every other point $\mathbf{s}_i$, and their point pair feature $\mathbf{F}$ is calculated. (2) Feature $\mathbf{F}$ is matched to the global model description, which returns a set of point pairs on the model that have similar distance and orientation (3). For each point pair on the model matched to the point pair in the scene, the local coordinate α is calculated by solving $\mathbf{s}_i = T_{s\to g}^{-1} R_{\mathbf{x}}(\alpha) T_{s\to g} \mathbf{m}_i$. (4) After α is calculated, a vote is cast for the local coordinate $(\mathbf{m}_i, \alpha)$.

method. Pose clustering increases the stability of the algorithm by removing isolated poses with low scores, and the averaging step increases the accuracy of the final pose.

4. Results

We evaluated our method against a large number of synthetic and real datasets and tested the performance, efficiency and parameter dependence of the algorithm.

For all experiments the feature space was sampled by setting d_{dist} to be relative to the model diameter $d_{dist} = \tau_d \; diam(M)$. By default, the sampling rate τ_d was set to 0.05. This makes the parameter independent from the model size. Normal orientation was sampled for $n_{angle} = 30$. This allows a variation of the normal orientation in respect to the correct normal orientation of up to 12°. Both model and scene cloud were subsampled such that all points have a minimum distance of d_{dist}. 1/5th of the points in the subsampled scene were used as reference points. After resampling the point cloud, the normals were recalculated by fitting a plane into the neighbourhood of each point. This step ensured that the normals correspond to the sampling level and avoids problems with fine details, such as wrinkles on the surface. The scene and the model are sampled in the same way and the same parameters were used for all experiments except where noted otherwise.

We will show that our algorithm has a superiour performance and allows an easy trade-off between speed and recognition rate. The algorithm was implemented in Java, and all benchmarks were run on a 2.00GHz Intel Core Duo T7300 with 1GB RAM. Even though it is straightforward to parallelize our method on various levels, we ran a single-threaded version for all timings. We assume that an implementation in C++ and parallelization would significantly speed up the matching times. All given timings measure the whole matching process including the scene subsampling, normal calculation, voting and pose clustering. Note that the pose was not refined by way of, for example, ICP [26], which would further increase the detection rate and accuracy. For the construction of the global model description, all point pairs in the subsampled model cloud were used. The construction took several minutes for each model.

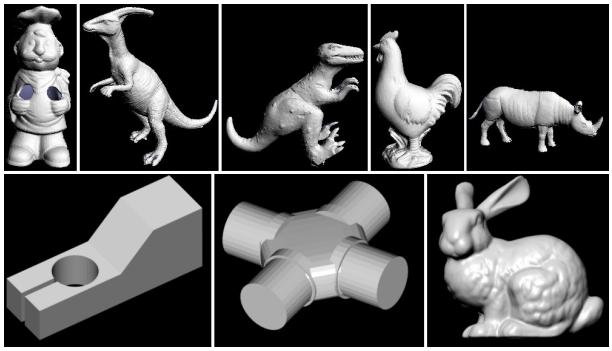
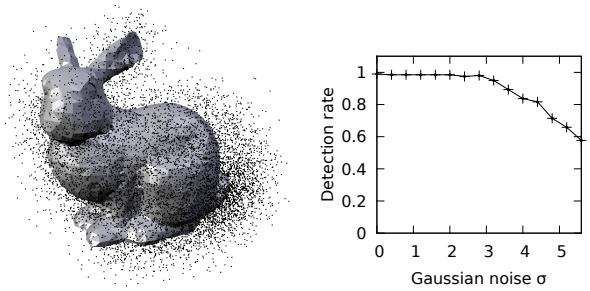


Figure 5. Models used in the experiments. Top row: The five objects from the scenes of Mian *et al.* [11, 10]. Note that the rhino was excluded from the comparisons, as in the original paper. Bottom row from left to right: The Clamp and the Cross Shaft from us, and the Stanford Bunny from [22].

4.1. Synthetic Data

We first evaluated the algorithm against a large set of synthetically generated three-dimensional scenes. We choosed four models of Fig. 5, the T-Rex, the Chef, the Bunny and the Clamp, to generate the synthetic scenes. The chosen models demonstrate different geometries.



(a) (b)

Figure 6. Results for the artificial scenes with a single object. (a) A point cloud with additive Gaussian noise added to every point ($\sigma = 5\%$ of the model diameter). The pose of the Bunny was recovered in 0.96 seconds. (b) Detection rate against gaussian noise over all 200 test scenes.

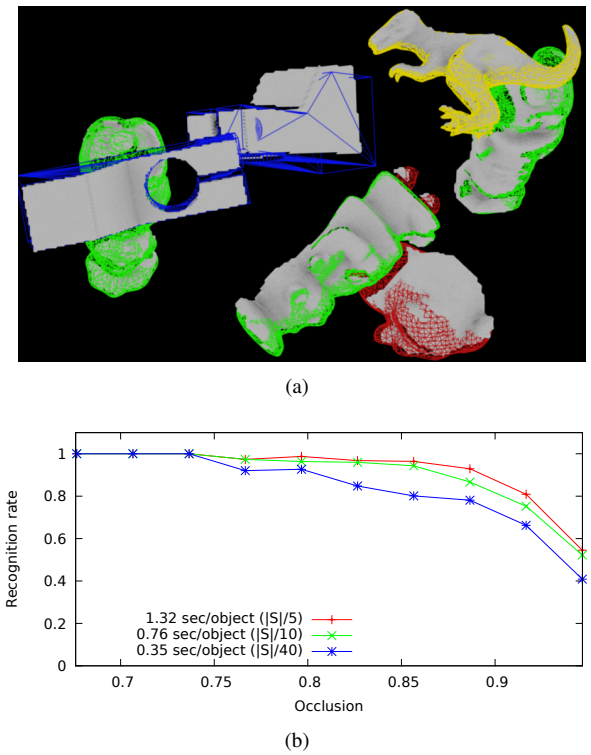
In the first set of experiments, scenes containing only a single object were used, and the performance with respect to noise was measured. Each of the four afore-mentioned objects was rendered from 50 different directions. This results in 200 point clouds, which were then corrupted by additive Gaussian noise with a standard derivation given relative to the object's diameter. This was done prior to the subsampling step. We were interested in the recognition rate, *i.e*. the number of scenes where the object was succesfully found, as well as in the accuracy of the recovered pose. An object is defined to be detected if the error of the resulting pose relative to the ground truth is smaller then some predefined threshold. In our experiments the threshold was set to $diam(M)/10$ for the translation and 12° for the rotation. Fig. 6 shows an example scene and the recoginition rates.

In a second set of experiments, 50 artificial scenes were rendered, each containing from four to nine randomly placed objects from the four objects used above. This means that multiple instances of one object can appear in one scene. In total, 347 objects were placed in 50 scenes. We measured the performance of our algorithm with respect to occlusions and in case of real data also with respect to clutter. The definition of [7] for occlusion and [10] for clutter, both defined per object instance, were used:

$$\text{occlusion} = 1 - \frac{\text{model surface area in the scene}}{\text{total model surface area}}, \quad (4)$$

$$\text{clutter} = 1 - \frac{\text{model surface area in the scene}}{\text{total surface area of scene}}. \quad (5)$$

The average number of points in the subsampled scenes is $|S| \approx 1690$. We ran out algorithm three times, using $1/5$th, $1/10$th and $1/40$th of the scene points as reference points. Fig. 7 shows an example scene and the recognition rates. Both the recognition rate and the execution time depend on the number of used reference points. For $|S|/5$ reference points, 89.3% of all objects were correctly detected.



(a)

(b)

Figure 7. (a) One of the 50 synthetic scenes with the detected objects overlayed as colored wireframe. The poses are the result of our method, without any pose refinement such as ICP. (b) Recognition rate against occlusion for the synthetic scenes with multiple objects. The number of reference points was $|S|/5$, $|S|/10$ and $|S|/40$ respectivly.

The missed objects are mostly the highly occluded ones: 98% of all objects with more than 15% of visible surface, *i.e*. less than 85% occlusion, were found. For $|S|/40$ reference points, 77.2% of all objects and 89.1% of objects more than 15% visible were found. However, the latter setup was roughly 4 times faster. From the experiments it is obvious that there is a tradeoff between speed and performance. A significantly faster matching can be archieved at the price of not detecting ill-conditioned objects with high occlusion.

4.2. Real Data

We now present the results on real data. We first present quantitative evaluations and comparisons with previous works and then show qualitative results on the scenes we acquired using a laser scanner.

Quantitative evaluation Our method was evaluated against the dataset provided by Mian *et al*. [10, 11], which consists of 50 scenes taken with a Minolta range scanner. Each scene contains four or five of the objects shown in Fig. 5 with known ground truth. In the original compari-

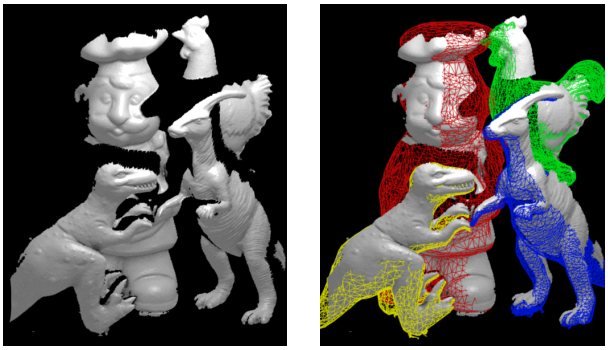
son, the rhino was excluded, because the spin images failed to detect it in any scene. We did the same to allow direct comparison with this prior work. Each object was searched in every scene using our method, and the pose with the best score from the pose clustering was then compared with the ground truth. We did two detection runs with varied sampling rate τ_d to test its influence on the detection rate and the runtime. Fig. 8(a) and 8(b) show an example scene with results. Fig. 8(c) shows the results for our two runs compared with the data of the spin images and the tensor matching from [10]. The parameters of spin images were set to yield maximum performance, resulting in a very large runtime.

For $\tau_d = 0.025$, our method runs approximatly as fast as the Tensor Matching of Mian *et al*. The recognition rate increases slightly to 97.0% of all objects with less than 84% occlusion compared to 96.6% for the tensor matching and 87.8% for spin images. The values relative to the 84%-boundary was taken from [10] and is given for comparability. The main advantage of our method is the possible trade-off between speed and recognition rate: For $\tau_d = 0.04$, our recognition rate drops to 89.2% for objects with less than 84% occlusion. However, the matching was more than 40 times faster and took less than 2 seconds per object instance. The recognition rate still exceeds that of spin images. The recognition rate in respect to clutter is similar. Note that for one object we try to recognize in the scene all the other objects are considered as clutter.

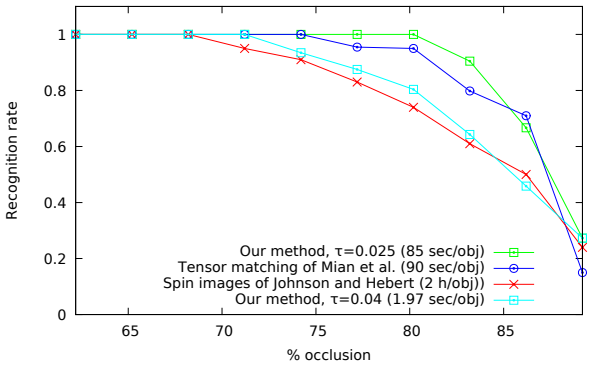
Qualitative results To show the performance of our algorithm concerning real data, we took a number of scenes using a self-build laser scanning setup. Two of the scenes and detections are shown in Fig. 1 and 9. We deliberately did not do any post-processing of acquired point clouds, such as outlier removal and smoothing. The objects were matched despite a lot of clutter, occlusion and noise in the scenes. The resulting poses seem accurate enough for object manipulation, such as pick and place applications.

5. Conclusion

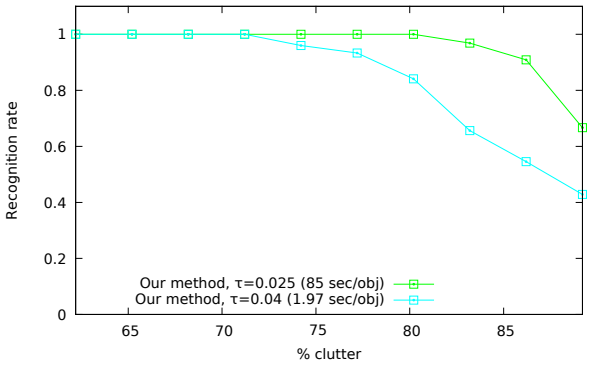
We introduced an efficient, stable and accurate method to find free-form 3D objects in point clouds. We built a global representation of the object that leads to independence from local surface information and showed that a locally reduced search space results in very fast matching. We tested our algorithm on a large set of synthetic and real data sets. The comparison with traditional approaches shows improvements in terms of recognition rate. We demonstrated that with slight or even no sacrifice of the recognition performance we can achieve dramatic speed improvement.



(a) (b)



(c)



(d)

Figure 8. (a) Example scene from the dataset of Mian *et al*. [10] (b) Recognition results with our method. All objects in the scene were found. The results were not refined. (c) Recognition rate against occlusion of our method compared to the results described in [10] for the 50 scenes. The sample rate is $\tau_d = 0.025$ with $|S|/5$ reference points for the green curve, and $\tau_d = 0.04$ with $|S|/10$ reference points for the light blue curve. (d) Recognition rate against clutter for our method.

References

[1] P. J. Besl and R. C. Jain. Three-dimensional object recognition. *ACM Computing Surveys (CSUR)*, 17(1):75–145, 1985.

[2] R. J. Campbell and P. J. Flynn. A survey of Free-Form object

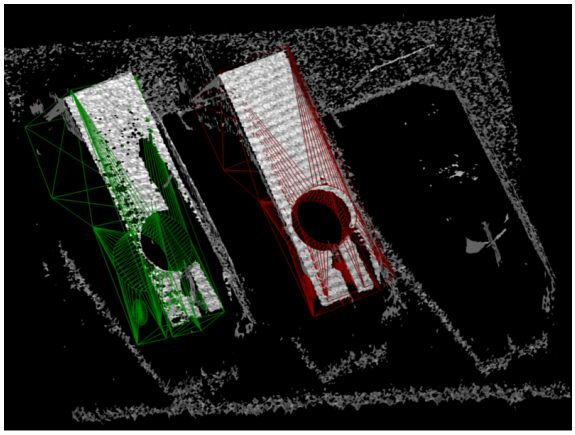


Figure 9. A cluttered, noisy scene taken with a laser scanner. The detected clamps are shown as colored wireframe.

representation and recognition techniques. *Computer Vision and Image Understanding*, 81(2):166–210, 2001.

[3] H. Chen and B. Bhanu. 3D free-form object recognition in range images using local surface patches. *Pattern Recognition Letters*, 28(10):1252–1262, 2007.

[4] C. S. Chua and R. Jarvis. Point signatures: A new representation for 3d object recognition. *International Journal of Computer Vision*, 25(1):63–85, 1997.

[5] C. Dorai and A. K. Jain. COSMOS - a representation scheme for 3D Free-Form objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(10):1115–1130, 1997.

[6] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann. Robust global registration. In *Proceedings of the third Eurographics symposium on Geometry processing*, page 197. Eurographics Association, 2005.

[7] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3 d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999.

[8] D. Katsoulas. Robust extraction of vertices in range images by constraining the hough transform. *Lecture Notes in Computer Science*, pages 360–369, 2003.

[9] G. Mamic and M. Bennamoun. Representation and recognition of 3D free-form objects. *Digital Signal Processing*, 12(1):47–76, 2002.

[10] A. S. Mian, M. Bennamoun, and R. Owens. Three-dimensional model-based object recognition and segmentation in cluttered scenes. *IEEE transactions on pattern analysis and machine intelligence*, 28(10):1584–1601, 2006.

[11] A. S. Mian, M. Bennamoun, and R. Owens. On the repeatability and quality of keypoints for local feature-based 3D object retrieval from cluttered scenes. *International Journal of Computer Vision*, pages 1–14, 2009.

[12] A. S. Mian, M. Bennamoun, and R. A. Owens. Automatic correspondence for 3D modeling: An extensive review. *International Journal of Shape Modeling*, 11(2):253, 2005.

[13] I. K. Park, M. Germann, M. D. Breitenstein, and H. Pfister. Fast and automatic object pose estimation for range images on the GPU. *Machine Vision and Applications*, pages 1–18, 2009.

[14] T. Rabbani and F. V. D. Heuvel. Efficient hough transform for automatic detection of cylinders in point clouds. In *Proceedings of the 11th Annual Conference of the Advanced School for Computing and Imaging (ASCI05)*, volume 3, pages 60–65, 2005.

[15] N. S. Raja and A. K. Jain. Recognizing geons from superquadrics fitted to range data. *Image and Vision Computing*, 10(3):179–190, 1992.

[16] S. Ruiz-Correa, L. G. Shapiro, and M. Meila. A new paradigm for recognizing 3-d object shapes from range data. In *Proc. Int. Conf. Computer Vision*, pages 1126–1133. Citeseer, 2003.

[17] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (FPFH) for 3D registration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Kobe, Japan*, 2009.

[18] R. Schnabel, R. Wahl, and R. Klein. Efficient RANSAC for point-cloud shape detection. In *Computer Graphics Forum*, volume 26, pages 214–226. Citeseer, 2007.

[19] Y. Shan, B. Matei, H. Sawhney, R. Kumar, D. Huber, and M. Hebert. Linear model hashing and batch ransac for rapid and accurate object recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2. IEEE Computer Society; 1999, 2004.

[20] F. Stein and G. Medioni. Structural indexing: efficient 3-D object recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):125–145, 1992.

[21] Y. Sun, J. Paik, A. Koschan, D. L. Page, and M. A. Abidi. Point fingerprint: a new 3-D object representation scheme. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 33(4):712–717, 2003.

[22] G. Turk and M. Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, page 318. ACM, 1994.

[23] E. Wahl, G. Hillenbrand, and G. Hirzinger. Surflet-pair-relation histograms: A statistical 3d-shape representation for rapid classification. In *3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings. Fourth International Conference on*, pages 474–481, 2003.

[24] J. Y. Wang and F. S. Cohen. Part II: 3-D object recognition and shape estimation from image contours using b-splines, shape invariant matching, and neural network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):13–23, 1994.

[25] T. Zaharia and F. Prêteux. Hough transform-based 3D mesh retrieval. In *Proc. of the SPIE Conf. 4476 on Vision Geometry X*, pages 175–185, 2001.

[26] Z. Zhang. Iterative point matching for registration of free-form curves. *Int. J. Comp. Vis*, 7(3):119–152, 1994.