

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

Chavdar Papazov and Darius Burschka

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

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

## 1 Introduction

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

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

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

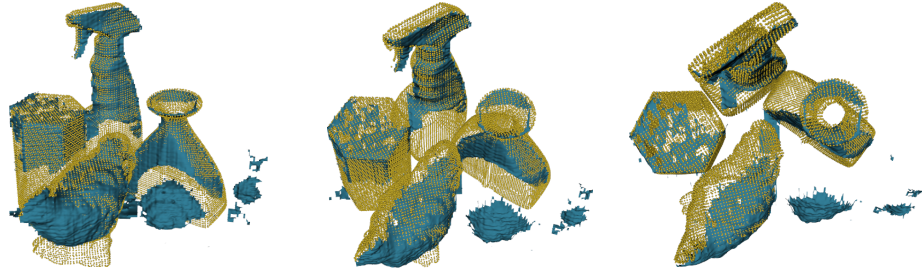
input:  
M: models  
S: Scene  
  
output:  
(M, T)  
Model and transform.

### ASSUMPTIONS

I need YCB models that have NORMAL.  
Range Image of the scene.  
Transformation must be rigid.

### Challenges

- Objects in the scene are not known beforehand.
- We don't know their pose in the scene.
- Scene has noise and outliers.
- Object is partially visible.



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

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

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

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

## 2 Related Work

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

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

### Classes of Object Recognition

1- **Voting Methods** (ex: Generalized Hough Transform, Geometric Hashing, Tensor Matching Algorithm)

The [Generalized Hough Transform](#) (GHT) is a powerful image processing and computer vision technique used for object recognition and pattern matching.

2- **Correspondence based methods** (Correspondences between the models and the scene are established usually using local geometric [descriptors](#) like [spin images](#), [local feature histograms](#), [3D shape context](#), [harmonic shape context](#) and [integral invariants](#)).

All correspondence based algorithms rely heavily on the assumption that the models to be recognized have distinctive feature points, i.e., points with rare descriptors.

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

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

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

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

if I know two doublets, I can determine the aligning transformation

Robust descriptor  
hashing technique (Doublets)  
efficient RANSAC

Descriptors:

#### - Shape Descriptor

- + Centroid
- + Bounding Box
- + Bounding Sphere
- + PCA

#### - Local Feature Descriptors

- + 3D Key Points
- + 3D Shape Context
- + Spin Image
- + SHOT (Signature of Histograms of Orientations)
- + PFH (Point Feature Histogram)

#### - Color and Texture Descriptors

- Statistical Descriptors

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

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

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

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

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

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

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

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

<sup>1</sup> A minimal point set is the smallest set of points required to uniquely determine a given type of transform.



**Well-grounded:**  
means that the criteria or methods  
being discussed are not based on  
solid theoretical principles  
(makeshift)

**Ad hoc**  
solutions are often temporary and  
may lack the robustness and  
generality of more principled  
methods.

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

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

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

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

### 3 Method Description

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

#### 3.1 Model Preprocessing Phase

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


Step1:

Given the Models,

- Sample pairs providing that the distance between them is (  $d - \delta_d, d + \delta_d$ )    We can tune  $d, \delta_d$     Large ( $d$ ) is preferred. (check reasons up)
- Keep only subset  $K$  of this Hash Table (  $K = 0.1$ )

-

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

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

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

### 3.2 Online Recognition Phase

The online recognition phase can be outlined as follows:

1. Initialization
  - (a) Compute an octree for the scene  $\mathbf{S}$  to produce a modified scene  $\mathbf{S}^*$ .
  - (b)  $\mathcal{T} \leftarrow \emptyset$  (an empty solution list).
2. Compute a number of iterations  $N$  needed to achieve a probability for successful recognition higher than a predefined value  $P_S$ .  
**[repeat  $N$  times]**
3. Sampling
  - (a) Sample a point  $\mathbf{p}_u$  uniformly from  $\mathbf{S}^*$ .
  - (b) Sample  $\mathbf{p}_v \in \mathbf{S}^*$  uniformly from all points at a distance  $d \pm \delta_d$  from  $\mathbf{p}_u$ .
4. Estimate normals  $\mathbf{n}_u$  and  $\mathbf{n}_v$  at  $\mathbf{p}_u$  and  $\mathbf{p}_v$ , respectively, to get an oriented scene point pair  $(\mathbf{u}, \mathbf{v}) = ((\mathbf{p}_u, \mathbf{n}_u), (\mathbf{p}_v, \mathbf{n}_v))$ .
5. Compute the descriptor  $f_{\mathbf{uv}} = (f_2(\mathbf{u}, \mathbf{v}), f_3(\mathbf{u}, \mathbf{v}), f_4(\mathbf{u}, \mathbf{v}))$  (see (1)).
6. Use  $f_{\mathbf{uv}}$  as a key to the model hash table to retrieve the oriented model point pairs  $(\mathbf{u}_j, \mathbf{v}_j)$  similar to  $(\mathbf{u}, \mathbf{v})$ .  
**[repeat for each  $(\mathbf{u}_j, \mathbf{v}_j)$ ]**
  - (a) Get the model  $\mathbf{M}$  of  $(\mathbf{u}_j, \mathbf{v}_j)$ .
  - (b) Compute the rigid transform  $T$  that best aligns  $(\mathbf{u}_j, \mathbf{v}_j)$  to  $(\mathbf{u}, \mathbf{v})$ .
  - (c) Set  $\mathcal{T} \leftarrow \mathcal{T} \cup (\mathbf{M}, T)$  if  $(\mathbf{M}, T)$  is accepted by an acceptance function  $\mu$ .**[end repeat]**
- [end repeat]**
7. Filter conflicting hypotheses from  $\mathcal{T}$ .

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

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

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

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

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

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

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

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

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

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

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

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

### 3.3 Time Complexity

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

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

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

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



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

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

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

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

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

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

$$P_M(n) = (m/n)KC. \quad (7)$$

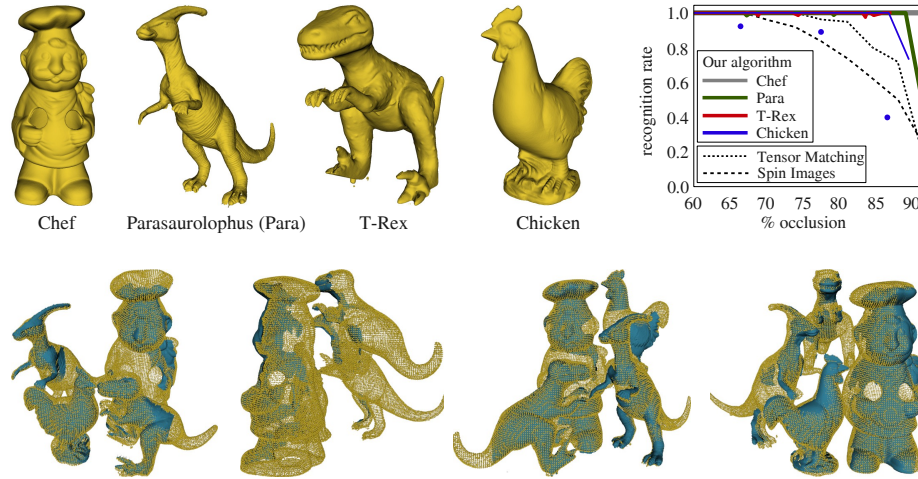
Again, approximating the denominator in (4) by its Taylor series  $\ln(1 - P_M(n)) = -P_M(n) + O(P_M(n)^2)$  we get for the number of iterations

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

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

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

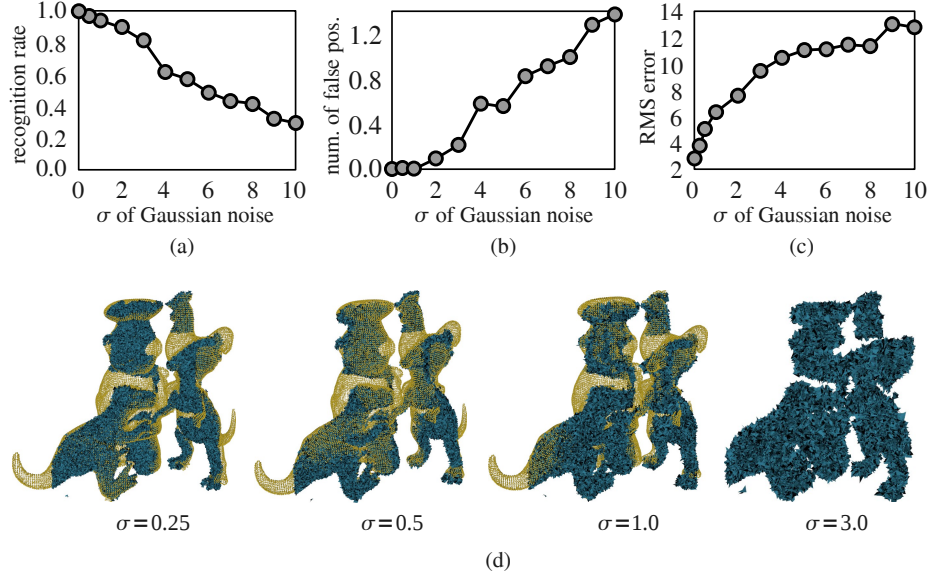
(iii) The acceptance function  $\mu$  runs in  $O(l)$  time, where  $l$  is the number of model points. Note that  $\mu$  does not depend on the number of scene points since back projecting a model point in the range image is performed in constant time.



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

## 4 Experimental Results

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

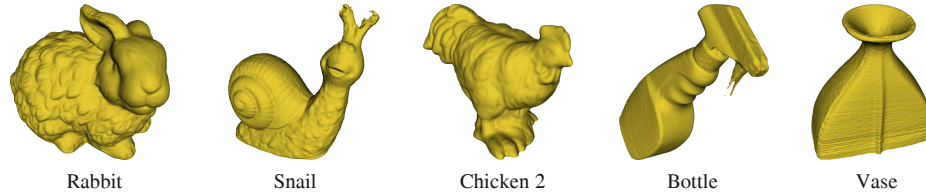


**Fig. 3.** (a) - (c) Recognition rate, mean number of false positives and mean RMS error as functions of the  $\sigma$  of Gaussian noise. One  $\sigma$  unit equals 1% of the bounding box diagonal length of the scene. The RMS units are in millimeters. (d) Typical recognition results for noise degraded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

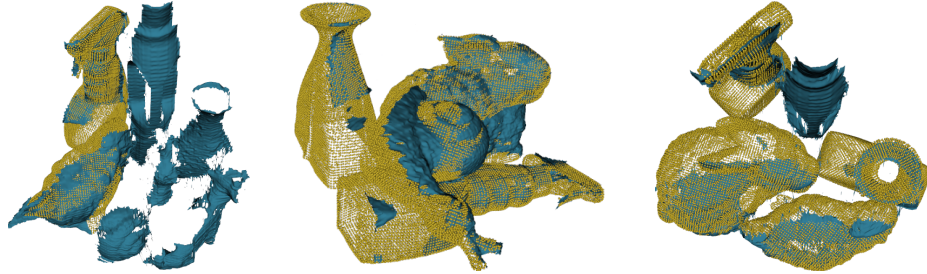
and compute the recognition rate for each object represented in the scene in the following way. We visually inspect the result of each of the 100 trials. If object **A** was recognized  $n$  times ( $0 \leq n \leq 100$ ) then the recognition rate for **A** is  $n/100$ . Since the occlusion of every object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

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

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



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



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

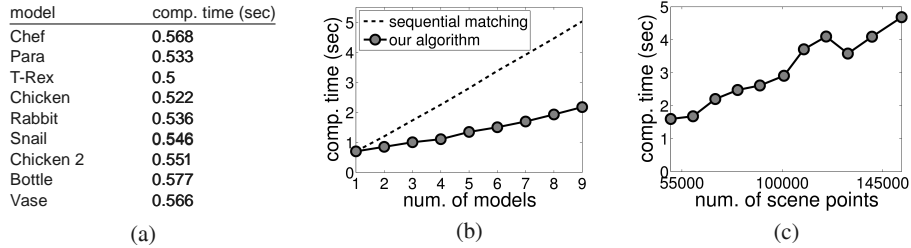
where  $N$  is the number of points in  $\mathbf{P}$ . Since we know the ground truth location of each model in the test scene the RMS error of the rigid transform computed by our method can be easily calculated.<sup>2</sup> The results of all noise tests are summarized in Fig. 3(a) – (c). Typical recognition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

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

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

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

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



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

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

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

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

## 5 Conclusions

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

## References

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