Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2014) 1–15

Procedia Computer Science

Efficient pose estimation of rotationally symmetric objects

Rui Pimentel de Figueiredo, Plinio Moreno, Alexandre Bernardino

Abstract

In this paper we propose algorithms for 3D object recognition from 3D point clouds of rotationally symmetric objects. We base our work in a recent method that represents objects using a hash table of shape features, which allows to match efficiently features that vote for object pose hypotheses. In the case of symmetric objects, the rotation angle about the axis of symmetry does not provide any information, so the hash table contains redundant information. We propose a way to remove redundant features by adding a weight factor for each set of symmetric features. The removal procedure leads to significant computational savings both in storage and time while keeping the recognition performance. We analyze the theoretical storage gains and compare them against the practical ones. We also compare the execution time gains in feature matching and pose clustering. The experiments show storage gains up to 100x and execution time savings up to 3500x with respect to state-of-the-art methods.

© 2011 Published by Elsevier Ltd.

Keywords: 3D pose estimation, 3D object recognition, symmetry, voting, real-time

1. Introduction

3-D object recognition plays a role of major importance in the robotics field. Many applications, such as object grasping and manipulation, critically depend on visual perception algorithms. These must be robust to cluttered environments and to sensor noise, as well as fast enough for real-time operation, in order for the robot to correctly interact with the surrounding environment. During the last decades several methods have been proposed to solve the object recognition problem, but it is still a very challenging task and many research efforts continue to be made. Due to recent technological advances in the field of 3-D sensing, range sensors provide 3-D points with reasonable quality and high sampling rates, sufficient for efficient shape-based object recognition. In recent work, Drost *et al.* [1] proposed an approach which extracts description from a given object model, using point pair features, encoding the geometric relation between oriented point pairs. The matching process is done locally using an efficient voting scheme (see Fig. 3) similar to the Generalized Hough Transform (GHT) [2]. Their method is robust to sensor noise and outperforms other feature-based state-of-the-art methods like Spin Images [3] and Tensors [4], both in terms of computational speed in terms, robustness to occlusion and clutter.

In this paper we introduce an important extension to [1] for dealing efficiently with rotationally symmetric objects [5], which are common in many daily tasks (e.g. kitchenware objects like cups, glasses, cans, plates), showing improvements both in memory storage and in processing speed with respect to [1].

We also extend the previous approach with a set of rules for re-sampling the point clouds that prevent aliasing effects related to feature space discretization, and an extensive analytical analysis of the proposed contributions.

Next section addresses the related work, followed by the description of Drost *et al.* object recognition and pose estimation algorithm. Then, in section 4 we propose a methodology to efficiently deal with rotational symmetries. In

section 5 we propose an automatic sampling selection criteria to avoid loss of information caused by sampling aliasing in the feature space. Lastly, in section 6 we show results that validate our approach.

2. Related work

Symmetry in objects has been extensively studied due to its application to areas such as: object recognition [6], shape matching [7], object model compression [8], geometric hashing [9] and pose detection [10]. The common idea across all these areas is that symmetry allows to define invariant models/features to geometric transformations, which are gathered in the symmetry group [11].

Regarding shape matching, symmetry descriptors (i.e. invariant features) are able to match parts of 3D objects that hold the symmetry constraints (i.e. partial symmetry). Kazhdan et. al. [7] introduced a collection of spherical functions that detect symmetry and retrieve objects. The main constraint of this work is the availability of full models in order to retrieve robustly the objects.

Regarding model compression, detection of symmetry in 3D models provides the parts of the models that can be compressed. Martinet et. al. [8] demonstrate experimentally the storage gains of the models when symmetries are detected in the objects.

Regarding geometric hashing, the parameters of symmetric transformations work as some of the indexes of the hash table. Geometric hashing was initially formulated on a 2D problem [9], and then extended to 3D object matching [12] and Bayesian hashing [13]. Wolfson and Rigoustsos [14] propose to reduce the size of the hash table by collapsing the redundant features, which in the case of geometric hashing could lead to storage savings by a factor of two.

In a recent work Thomas [10] reduces the size of the hash table for pose detection, exploiting symmetries. Each object is represented on a hash table that indexes features computed on point pairs and triplet pairs. The storage size of the hash table is reduced by considering rotational symmetries.

In this work we follow the idea of Wolfson and Rigoustsos [14], which is also applied by Thomas [10]. Similarly to Thomas [10], we build a hash table indexed by point pairs. We apply the hash table reduction of [14] to the method of [1] and, in contrast to previous works, we analyze the theoretical computational storage savings and confirm these results with experiments on real data.

3. Method Overview

The basic units to describe surface shape are surflets [15] $\mathbf{s} = (\mathbf{p}, \mathbf{n})$, where $\mathbf{p}$ represents sample points in the surface and $\mathbf{n}$ are the associated surface normals. Let M be the set of all model surflets, $M = \{\mathbf{s}_i^m, i = 1..N\}$ and let S be the set of all scene surflets, $S = \{\mathbf{s}_i^s, i = 1...N\}$.

The recognition process consists in matching scene surflet pairs $(\mathbf{s}_r^s, \mathbf{s}_i^s)$ to model surflet pairs $(\mathbf{s}_r^m, \mathbf{s}_i^m)$. Being $\mathbf{s}_r$ and $\mathbf{s}_i$ two surflets, the Point Pair Feature (PPF) $\mathbf{F} \in F \subset \mathbb{R}^4$ is defined as a 4-tuple composed by: the distance between the reference, $\mathbf{p}_r$, and secondary, $\mathbf{p}_i$, points; the angle between the normal of the reference point $\mathbf{n}_r$ and the vector $\mathbf{d} = |\mathbf{p}_i - \mathbf{p}_r|$; the angle between the normal of the secondary point $\mathbf{n}_i$ and $\mathbf{d}$; and the angle between $\mathbf{n}_r$ and $\mathbf{n}_i$ as illustrated in Fig. 1. This could be formally described by

$$\mathbf{F} = \mathrm{PPF}(\mathbf{s}_r, \mathbf{s}_i) = (f_1, f_2, f_3, f_4) = (\|\mathbf{d}\|, \angle(\mathbf{n}_r, \mathbf{d}), \angle(\mathbf{n}_i, \mathbf{d}), \angle(\mathbf{n}_r, \mathbf{n}_i)) \tag{1}$$

The data structure chosen to represent the model description was a hash table, for fast lookup during the matching phase, in which the key value is given by the discrete PPF while the mapped value is the respective surflet pair. In a more rigorous sense, the hash table is a multi-valued hash table since one key could be associated with several surflet pairs. Thus, each slot of the hash table contains a list of surflet pairs with similar discrete feature. The hash function samples the 4-tuple PPF and converts the resulting discrete feature to a single integer I which serves as an index of the hash table. This conversion is given by the multi-dimensional array addressing formula:

$$\begin{aligned} I &= \mathrm{hash}(\mathbf{F}, n_\alpha) = \\ &= f_1^{\mathrm{bin}} + f_3^{\mathrm{bin}} \cdot n_\mathrm{d} \cdot n_\alpha + f_4^{\mathrm{bin}} \cdot n_\mathrm{d} \cdot n_\alpha \cdot n_\alpha \end{aligned} \tag{2}$$

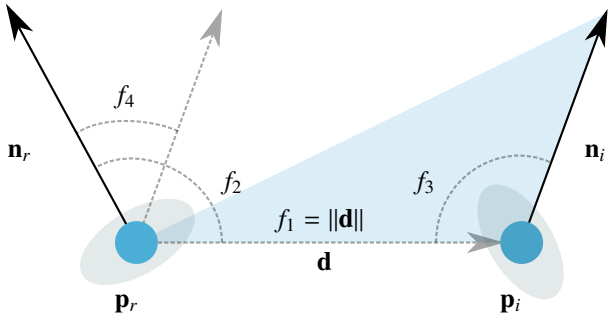


Figure 1. Point Pair Feature

The hash function is thus a mapping from the PPF space F to the model description A, which can be formally expressed by the following:

$$\text{hash} : F \subset \mathbb{R}^4 \rightarrow A \subset M^2 \tag{3}$$

3.1. Pose Estimation

A set of reference surflets on the scene $R_s \subset S$ is uniformly sampled from S and each of them is paired with all the other surflets on the scene. The number of reference points is given by $|R_s| = \xi |S|$ where $\xi \in [0; 1]$ is the reference points sampling ratio control parameter. For each scene surflet pair $(\mathbf{s}_r^s, \mathbf{s}_i^s) \in S^2$, $\text{PPF}(\mathbf{s}_r^s, \mathbf{s}_i^s)$ is computed and set of similar model surflet pairs is retrieved from the hash table. From every match between a scene surflet pair $(\mathbf{s}_r^s, \mathbf{s}_i^s) \in S^2$ and a model surflet pair $(\mathbf{s}_r^m, \mathbf{s}_i^m) \in M^2$, one is able to compute the rigid transformation that aligns the matched model with the scene. This is done first by computing the transformations $\mathbf{T}_{m\rightarrow g}$ and $\mathbf{T}_{s\rightarrow g}$ that align $\mathbf{s}_r^m$ and $\mathbf{s}_r^s$, respectively, to the object reference coordinate frame x axis, and secondly by computing the rotation α around the x axis that aligns $\mathbf{p}_i^m$ with $\mathbf{p}_i^s$. The transformation that aligns the model with the scene is then computed considering the ensuing expression:

$$\mathbf{T}_{m\rightarrow s} = \mathbf{T}_{s\rightarrow g}^{-1}\mathbf{R}(\alpha)\mathbf{T}_{m\rightarrow g} \tag{4}$$

In detail, the transformations $\mathbf{T}_{m\rightarrow g}$ and $\mathbf{T}_{s\rightarrow g}$ translate $\mathbf{p}_r^m$ and $\mathbf{p}_r^s$, respectively, to the reference coordinate frame origin and rotates their normals $\mathbf{n}_r^m$ and $\mathbf{n}_r^s$ onto the x axis. After applying these two transformations, $\mathbf{p}_i^m$ and $\mathbf{p}_i^s$ are still misaligned. The transformation $\mathbf{R}(\alpha)$ applies the final rotation needed to align these two points. The previous reasoning is depicted in Fig. 2.

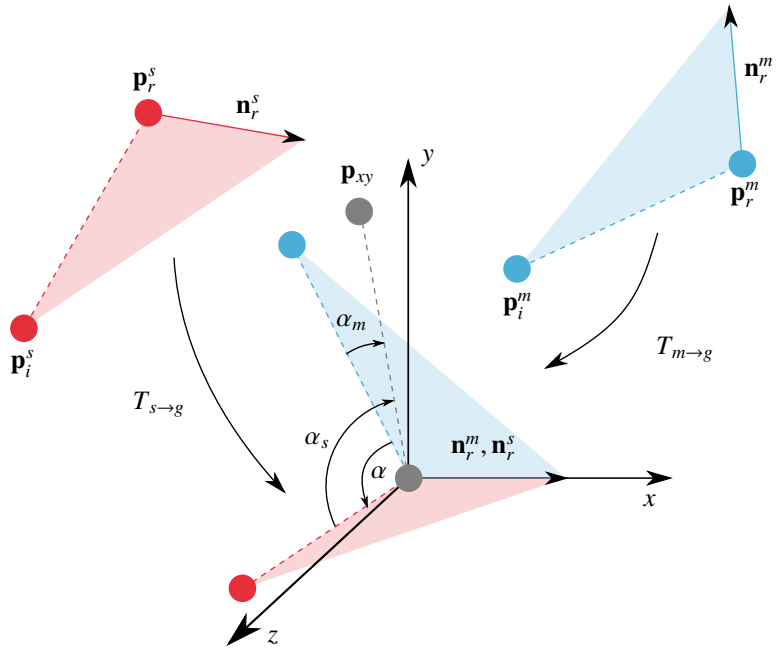


Figure 2. Pose acquisition by surflet pair aligment

The transformation expressed in eq. (4) can be parametrized by a surflet on the model and a rotation angle α. In [1], this pair $(\mathbf{s}_r^m, \alpha)$ is mentioned as the *local coordinates* of the model with respect to reference point $\mathbf{s}_r^s$.

3.1.1. Voting Scheme

This method uses a voting scheme similar to the Generalized Hough Transform (GHT) for pose estimation. For each scene reference surflet, a two-dimensional accumulator array that represents the discrete space of local coordinates is created. The number of rows, N_m, is the same as the number of model sample surflets $|M|$, and the number of columns N_α is equal to the number of sample steps of the rotation angle α. A vote is placed in the accumulator array

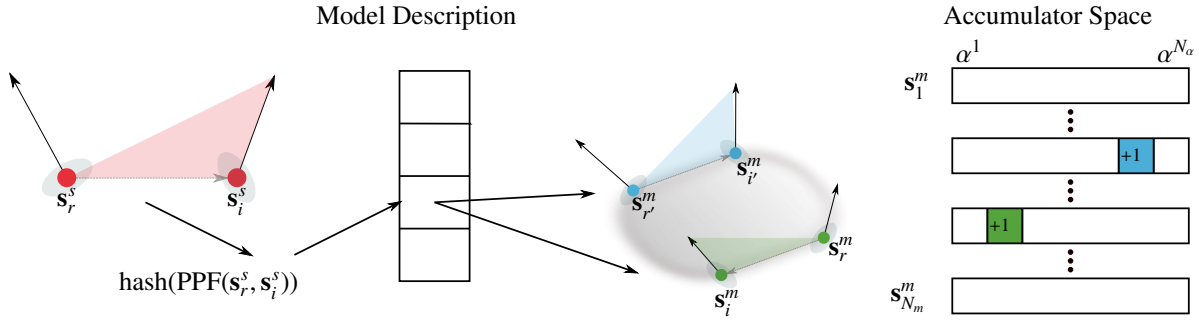


Figure 3. Feature Matching and Voting Scheme

by incrementing the position associated to the local coordinates $(\mathbf{s}_r^m, \alpha)$, by 1 (see Fig. 3). After pairing $\mathbf{s}_r^s$ with all $\mathbf{s}_i^s$, the highest peak – i.e. the position with most votes – in the accumulator, corresponds to the optimal local coordinate.

In the end, all retrieved pose hypotheses whose position and orientation do not differ more than a predefined threshold are clustered together.

3.1.2. Pose Clustering

As shown in the pseudo-code of Algorithm 1, the voting process is done for several reference surflets on the scene to ensure that at least one of them lies on the object we want to detect. For each reference surflet, a set of pose hypotheses – each corresponding to a peak in the accumulator array and weighted by its associated number of votes – is generated. After the voting procedure, we consider an additional pose clustering step that brings stability and increases the accuracy of the final result since the retrieved poses only approximate the ground truth due to sensor noise and sampling artifacts. The pose clustering procedure (see Algorithm 1) outlined in this subsection is a clustering algorithm that aggregates all pose hypotheses that do not differ in position and orientation for more than predefined thresholds $\frac{\text{diam}(M)}{\sigma_d}$ and $\frac{2\pi}{\sigma_\alpha}$ where σ_d and σ_α are two clustering control parameters. The procedure starts by sorting all poses in decreasing order according to their number of votes. A cluster is created for the first pose, *i.e.*, the pose with more votes. The algorithm iterates over the remaining hypotheses and similar hypotheses are grouped together in the same cluster. If a hypothesis differs in position or orientation, more than specified thresholds, from the existing clusters, then a new cluster is initialized. In the end, each cluster has a score equal to the sum of the scores of the pose hypotheses contained in it and its resulting pose is the average of the poses contained in that cluster. The resulting clusters are sorted again in decreasing order and the top ranked clusters are returned. The number of returned clusters, ζ, is equal to the object instances that one expects to be in the scene.

begin
1: **for all** $\mathbf{s}_r^s \in R_s$ **do**
2: reset the Hough accumulator
3: **for all** $\mathbf{s}_i^s \in S \setminus s_r^s$ **do**
4: compute PPF(s_r^s, s_i^s) and the corresponding hash table index
5: **for all** feature matches **do**
6: vote on the Hough accumulator
7: **end for**
8: **end for**
9: get highest peaks on the accumulator and compute the corresponding pose transforms
10: **end for**
11: pose clustering
end

Algorithm 1: Pose estimation

4. Dealing with Rotational Symmetry

We consider an object to be rotationally symmetric if its shape appearance is invariant to rotations around a given axis of symmetry (see Fig. 4). In order to efficiently deal with this kind of objects, we incorporate a strategy that reduces the size of the model description A, by discarding redundant surflet pairs, thus obtaining significant gains in storage and speed. To accomplish this, a Euler angle representation [16], is used to describe orientation. In our work we chose the X-Y-Z Euler representation since we assume that the object axis of symmetry Ψ is aligned with the z axis of the object reference coordinate frame:

$$\Psi = (0, 0, z) \tag{5}$$

During the creation of the model description, for each surflet pair, we compute the transformation with respect to the object model reference frame (see section 3.1) that aligns it with each similar pair already stored in the hash table. If the aligning transformation has a very low translation $\mathbf{t}$ and if the axis of rotation $\mathbf{r}_\text{axis}$ is aligned with the $\mathbf{z}$ axis, then this surflet pair corresponds to a rotation around the symmetry axis. Thus, the surflet pair is redundant because can be represented jointly with the other similar pairs (homologous):

$$2\text{acos}\left(\mathbf{r}_\text{axis} \cdot \mathbf{z}\right) < \phi_\text{th} \quad \text{and} \quad \|\mathbf{t}\| < t_\text{th} \tag{6}$$

The weight w of the homologous surflet pair, stored in the hash table, is then incremented by 1. This process is clearly illustrated in Fig. 5.

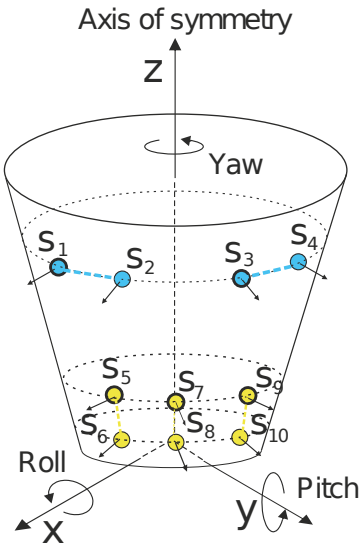


Figure 4. An example of a rotationally symmetric object model. All illustrated surflet pairs have similar discrete feature. In the figure, pairs represented with similar color are redundant.

Due to the fact that the sampled model point clouds are not perfect, *i.e.*, only approximate the true shape of the object, the α_{th} and d_{th} thresholds must take into account these sampling imperfections. Higher thresholds increase the number of jointly represented surflet pairs but reduce the stringency with which we consider two given model surflet pairs redundant.

By representing redundant features jointly we decrease the number of feature matches thus decreasing the computations during the voting process. Each feature match contributes with a weight equal to the model feature weight, w, instead of 1. The peaks in the accumulator – originated by redundant surflet pairs – which were previously scattered throughout the local coordinates are now concentrated at single local coordinates. This is the result of keeping only one surflet pair, *i.e.*, one local coordinate, representing all the redundant ones which correspond to different local coordinates.

Before clustering, we collapse all poses that only differ on the yaw component, *i.e.*, redundant hypotheses, to a single pose. This is achieved by means of an additional step – after knowing the final transformation $\mathbf{T}_{m\to s}$ (see equation (4)) – which removes the rotational component around the object axis of symmetry, *i.e.*, $\phi_{yaw} = 0$, ensuring that all redundant poses are gathered in the same cluster, therefore allocating less resources and reducing the number of computations during the pose clustering step. In the following section we provide a mathematical analysis of the computational gains of our method.

4.1. Computational effiency

To analytically model the computational savings of our approach let us consider a generic rotationally symmetric object surface, represented by a set of surflets. Given the axis of symmetry, we assume the surflets are sampled on a set of planar sections orthogonal to the axis. The intersection of the sections with the object are circles of several radii, which represent a one-dimensional signal in polar coordinates. On these circular paths the surflets are sampled uniformly according to the perimeter of each circle. Let us denote by $L = \{l_1, \ldots, l_i, \ldots, l_n\}$ the set of circular paths, where the surflets are sampled at each circle (i.e. section) with frequency f_0^i (deg/surflet) so the number of surflets at section i are $360/f_0^i$.

We consider two sets of surflet pairs : (i) The pairs computed from all surflet combinations on a circular path (i.e. intra-section) and (ii) the pairs computed from all surflet combinations between two circular paths (i.e. inter-section). Let $M^{l_j} = \{\mathbf{s}_1^{m,l_j}, \ldots, \mathbf{s}_i^{m,l_j}, \ldots, \mathbf{s}_{360/f_0^j}^{m,l_j}\}$ be the set of surflets at level l_j, having the sampling frequency f_0^j and radius r_j. The frequency f_0^j express the periodicity of the intra-section surflet pairs, because a rotation of a surflet pair by kf_0^j

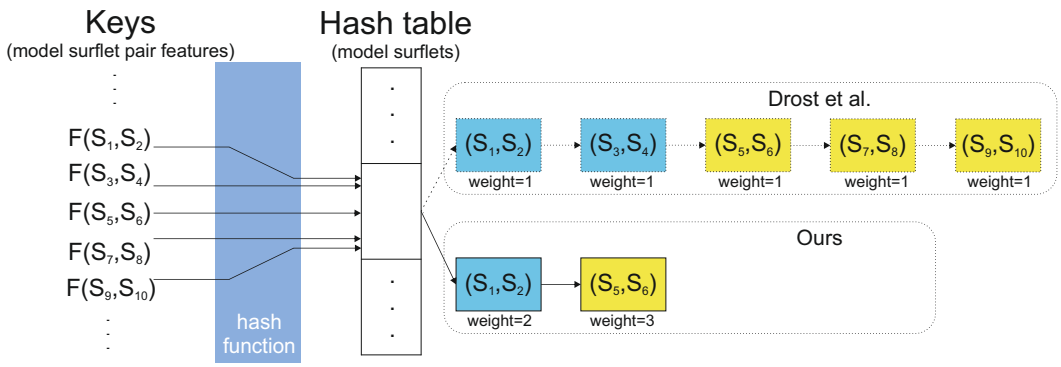


Figure 5. Example of surflet pairs with similar feature stored in the same slot of the hash table, during the creation of the object model description.

is equivalent to the surflet computed at kf_0^j. Thus, the frequency f_0^j yields the storage gain of the intra-section surflet pairs.

Let $A^{l_i} = \{(\mathbf{s}_r^{m,l_i}, \mathbf{s}_j^{m,l_i}), r \neq j\}$ be the set of intra-section surflet pairs at level l_i, with cardinality given by all possible surflet pairs:

$$\left|A^{l_i}\right| = \left|M^{l_i}\right| \times \left(\left|M^{l_i}\right| - 1\right). \tag{7}$$

Since f_0^j provides the storage gain of the intra-section surflet pairs, the number of representative features of this set is as follows:

$$\left|A^{l_i,r}\right| = \frac{\left|A^{l_i}\right|}{f_0^i}. \tag{8}$$

The inter-section set of the surflet pairs belonging to two distinct circular paths l_i and l_j is denoted by $A^{l_i,l_j} = \{(\mathbf{s}_r^{m,l_i}, \mathbf{s}_k^{m,l_j})\} \cup \{(\mathbf{s}_r^{m,l_j}, \mathbf{s}_k^{m,l_i})\}$, and its corresponding cardinality given by:

$$\left|A^{l_i,l_j}\right| = \left|M^{l_i}\right|\left|M^{l_j}\right| \tag{9}$$

In the case of inter-section surflet pairs, given the periodicity of each level one has to find the periodicity of the new rigid object. The periodicity of the attached section is another periodic signal with frequency:

$$f_0^{i,j} = GCD\left(f_0^i, f_0^j\right) = GCD\left(\left|M^{l_i}\right|, \left|M^{l_j}\right|\right) \tag{10}$$

where $GCD : \mathbb{N}^2 \longmapsto \mathbb{N}$ is the Greatest Common Divisor function. Similarly to the intra-section frequency, the inter-section frequency $f_0^{i,j}$ express the periodicity of the inter-section surflets, providing the storage gain for each inter-section. Thus, the set of representative features (*i.e.*, features that are sufficient to describe the object) for a combination of two sections is given by:

$$\left|A^{l_i,l_j,r}\right| = \frac{\left|A^{l_i,l_j}\right|}{f_0^{i,j}}. \tag{11}$$

The set of representative features A^r is obtained by the union of all the intra-section and inter-section sets. The size of A^r is simply the summation of all intra and inter-section representative features as follows:

$$|A^r| = \underbrace{\sum_{i=1}^{|L|} \frac{\left|A^{l_i}\right|}{f_0^i}}_{intra-plane} + \underbrace{\sum_{i=1}^{|L|} \left(\sum_{j=1}^{i-1} \frac{\left|A^{l_i,l_j}\right|}{f_0^{i,j}} + \sum_{j=i+1}^{|L|} \frac{\left|A^{l_i,l_j}\right|}{f_0^{i,j}} \right)}_{inter-plane}. \tag{12}$$

The total number of surflet pairs initially computed on the object is as follows:

$$|A| = \underbrace{\sum_{i=1}^{|L|} \left|A^{l_i}\right|}_{intra-plane} + \underbrace{\sum_{i=1}^{|L|} \left(\sum_{j=1}^{i-1} \left|A^{l_i,l_j}\right| + \sum_{j=i+1}^{|L|} \left|A^{l_i,l_j}\right| \right)}_{inter-plane}. \tag{13}$$

The storage saving provided by our method is $G = |A^r| \,/\, |A|$. In the following we compute the storage gains for two extreme cases of rotationally symmetric geometries: Cylinder and cone.

4.1.1. Case studies

We proceed with the analysis of the storage gains of two basic parametric shape primitives.

Cyllinder. We first consider the cyllinder since it is the simplest parametric rotationally symmetric shape. Taking into account that the cylinder radius is constant for all cutting-planes, we have $\left|M^{l_i}\right| = f_0, \forall_{l_i \in L}$. Therefore the number of representative features and the correspondent storage gain are as follows:

$$|A^r| = \underbrace{\sum_{i=1}^{|L|} (f_0 - 1)}_{|M|-|L|} + \underbrace{\sum_{i=1}^{|L|} \left(\sum_{j=1}^{i-1} f_0 + \sum_{j=i+1}^{|L|} f_0 \right)}_{|M|(|L|-1)} = |L|\,(|M| - 1) \tag{14}$$

$$G_{\text{cyl}} = \frac{|A|}{|A^r|} = \frac{|M|\,(|M|-1)}{|L|\,(|M|-1)} = \frac{|M|}{|L|} = f_0. \tag{15}$$

The total gain of Eq. (15) is a simple expression, which is independent of the cyllinder height, depending only on the number of samples, *i.e.* sampling frequency, per cutting-plane, which is constant. Figure 6(a) illustrates this fact, by varying the number of levels $|L|$ and the frequency of the cylinder's base, f_0^{base}.

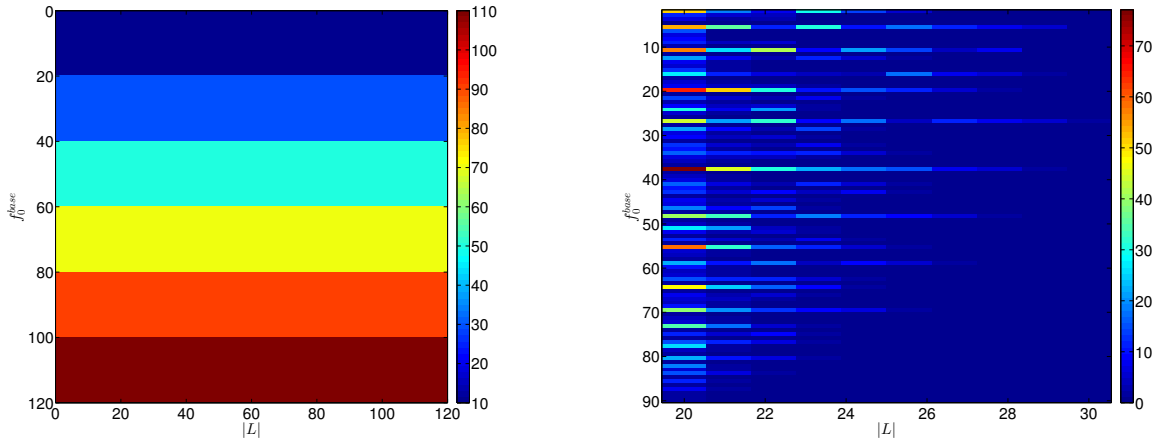
Cone. The inter-section storage gains of this primitive cannot be simplified like in the cylinder, because of the solution of Eq. (10) is in several cases 1. Given the frequency of the base section $f_0^{l_1} = f_0^{\text{base}}$; and the frequency of the top section $f_0^{l_{|L|}} = f_0^{\text{top}}$; we constrain the frequency of a given level as follows:

$$f_0^{l_i} = f_0^{l_{i-1}} + \frac{f_0^{\text{base}}}{|L| - 1} \quad \text{with} \quad f_0^{l_i} \in \mathbb{N} \tag{16}$$

In Figure 6(b) we plot the storage gains on a cone in function of the frequency at the base section $f_0^{\text{base}} \subset \mathbb{N}$ and the number of levels $|L|$. On one hand, for a given cone base radius $f_0^{l_i}$, the storage gains decrease exponentially as we increase the number of levels $|L|$ of the cone. On the other hand, for a constant $|L|$ the gains are highly dependent on the solution of *GCD* which is a discontinuous function. Having the curves of Figure 6 in mind, one expect that other objects have storage gains between the cylinder and the cone, as rotationally symmetric objects can be decomposed in a set of cylinders and clipped cones.

5. Automatic Sampling Selection

In order to ensure that the surface points displacement correspond to the feature distance space sampling level, and also to normalize the number of points in the scene with respect to the model, both models and scene point clouds are primarily downsampled using a space quantization step $s_{\text{step}} = \frac{\text{diam}(M)}{n_s}$, where n_s is the number of quantization steps. This is done by partitioning the 3-D space using a *voxel* (*i.e.*, a $s_{\text{step}} \times s_{\text{step}} \times s_{\text{step}}$ 3-D square box, analogous to the *pixel*, that represents a discrete value in 3-D space) grid and approximating (*i.e.*, downsampling) each voxel by the centroid of the points contained in it. Bigger sampling step sizes (*i.e.*, *bigger voxels* or *at lower resolution*) reduces the number of considered sample points and consequently the algorithm's computational effort. Bigger sampling step sizes also decreases the noise levels but at the cost of removing the surfaces fine details thus losing discriminative power which



(a) Cyllinder (b) Cone

Figure 6. Theoretical storage gains

is essential if we want to be able to distinguish similar surfaces (*e.g.* at low resolutions a wrinkled surface becomes smooth). Since the noise level depends mainly on the sensor characteristics, the selection of the sampling step size should be chosen according to the sensor noise model and not according to the scene which is unpredictable. After the downsampling step, the surface normals are calculated – or re-calculated in the case of the off-line construction of the model description phase. This step also ensures that the surface normals correspond to the sampling level. To avoid the loss of information in the feature space caused by sampling aliasing, the space quantization step must be chosen such that the Nyquist criterion is verified:

$$n_{\mathrm{d}} > 2n_{\mathrm{s}} \tag{17}$$

Taking this into account we select n_{d} according to the following rule:

$$n_{\mathrm{d}} = 2n_{\mathrm{s}} + 1 \tag{18}$$

The feature angle quantization step α_{step} is further obtained by constraining it to be equal to the angle between two neighbour points on the model section with bigger radius, at the feature sampling level:

$$d_{\mathrm{step}}^2 = \Big(r\sin(\alpha_{step})\Big)^2 + \Big(r - r\cos(\alpha_{step})\Big)^2 \iff \alpha_{\mathrm{step}} = 2\mathrm{acos}\left(1 - 0.5\left(\frac{d_{\mathrm{step}}}{r}\right)^2\right) \tag{19}$$

where r is the radius of the biggest model section and is obtained by searching for the furthest model surface point to the axis of symmetry:

$$r = \max\left(\left\|\mathbf{p}_{z=0}^{i} - \Psi\right\|\right) \tag{20}$$

6. Results

Several experiments were done in simulation in order to validate and evaluate the storage and performance gains of the proposed strategies to deal efficiently with symmetry. Not only the method performance and storage gains were evaluated but also its robustness when dealing with different levels of noise and clutter. The performance dependence on the methods parameters were also evaluated. All algorithms were implemented in C++ and the experiments were run on a single core of a dual-core 2.6 GHz computer with 4GB of RAM. In all our experiments we set the pose thresholds to $\phi_{\mathrm{th}} = \alpha_{\mathrm{step}}$ and $t_{\mathrm{th}} = d_{\mathrm{step}}$. During recognition, all peaks in the accumulator having an amount of votes at least higher than 75% relatively to the highest peak were retrieved. To ensure this, the parameter ρ was set to 0.75.

The clustering control parameters were set to $\sigma_{\mathrm{d}} = n_{\mathrm{d}}$ and $\sigma_{\alpha} = n_{\alpha}$ meaning that all aggregated poses inside a cluster, did not differ in orientation and position more than α_{step} and d_{step}, respectively.

In all our experimental scenarios the models library comprised only one model at a time, since we were interested in evaluating the quality of the poses recovered by the algorithms. With this purpose, for all experimental scenarios we generated 200 synthetic scenes containing a single instance of a given object model, on a random pose. By using synthetically generated scenes, we were able to compare the algorithm pose results with a known ground truth. A recovered pose was considered to be correct if the error relative to the ground truth pose (*i.e.* position and orientation) was smaller than $\frac{\mathrm{diam(M)}}{20}$ for the position and 12° for the orientation.

In the next subsection we provide results that support the proposed algorithms and the computational gains mathematical analysis described in 4.1. We also evaluate its computational gains and robustness in the presence of noisy data. Then, in subsection 6.1.2 we evaluate our method with common objects which were acquired with noisy range scanners.

6.1. Performance evaluation

The aim of our first experiment was to validate our method gains by numerically compare our implementation against the theoretical gain obtained in section 4.1, and to quantify the algorithm robustness to noise. For this purpose, we synthetically generated a set of cyllinders and cones with varying parameters.

Cyllinder. The cyllinder surface was uniformly sampled by constraining the distance between connected neighbour surflets (*i.e.* surface sampling s_{step}) to be constant, *i.e.* $s_{\mathrm{step}} = 1$. The cyllinder radius r was then obtained according to:

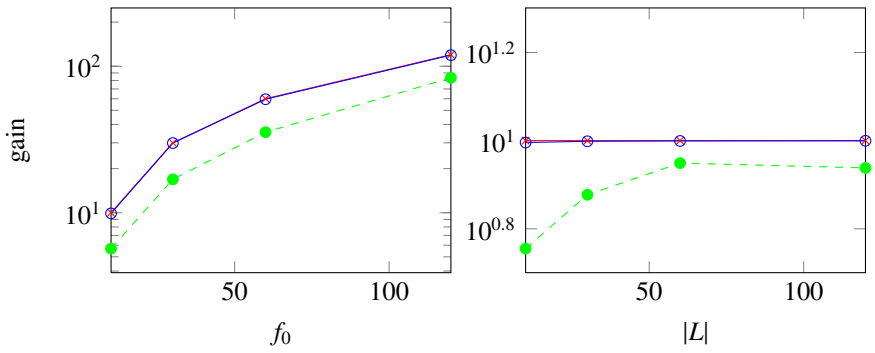
$$r = \sqrt{\frac{s_{\mathrm{step}}^2}{2 - 2\cos\left(\frac{2\pi}{f_0}\right)}} \quad \text{with} \quad s_{\mathrm{step}} = 1 \tag{21}$$

where $f_0 \subset \mathbb{N}$ is de point sampling frequency at the cyllinder radius, *i.e.* the number of points per level section. The cyllinder height h was set to be given by $h = |L|\, s_{\mathrm{step}}$, such that $h \subset \mathbb{N}$.

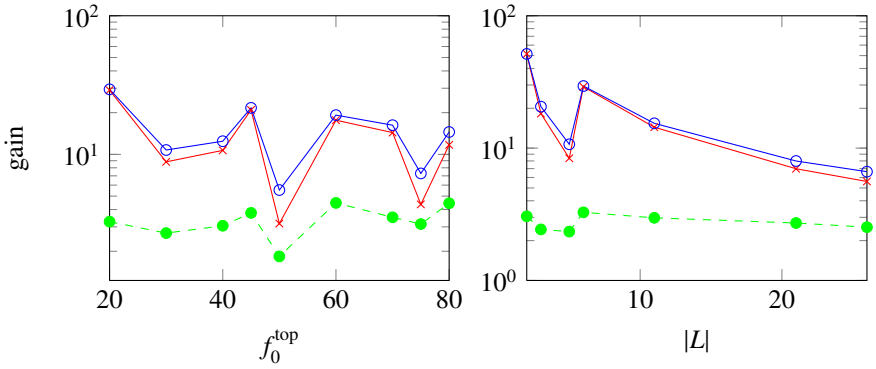
Cone. The cone shapes generation was similar to the cyllinder with $h = |L|\, s_{\mathrm{step}}$ and constrained to the possible solutions of eq.(16) with fixed $f^{\mathrm{base}} = 120$. In our experiments we first fixed $f_0^{\mathrm{top}} = 20$ and varied $|L|$ in $[2, 60]$ and then fixed $|L|$ and varied in f_0^{top} in $[20, 80]$.

6.1.1. Computational gains

In our first experimental scenario we were interested in measuring the feature storage and time savings of our method, in the absence of noise. Thus, only one scene point was selected as reference point and only the highest peak in the accumulator was retrieved. To evaluate the time performance we measured only the feature matching gains since the other steps – except the clustering step – of the online pose estimation (algorimth 1) are constant. It is important to remark that the clustering step is not performed when only a single scene reference point is selected and the highest peak is retrieved from the accumulator. We first set f_0, r, and vary the number of sections $|L|$, *i.e.*, h. As exhibited in figure 7 the obtained storage gains are equal to the ones determined analitically while the feature matching time savings are upper bounded by the storage gains.



(a) Cyllinder.

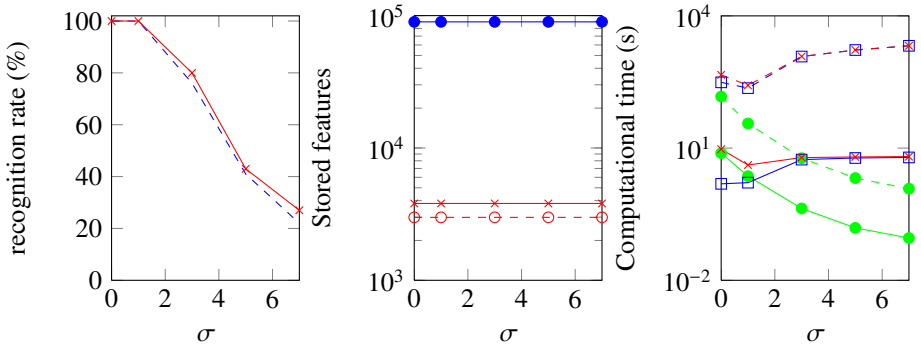


(b) Cone.

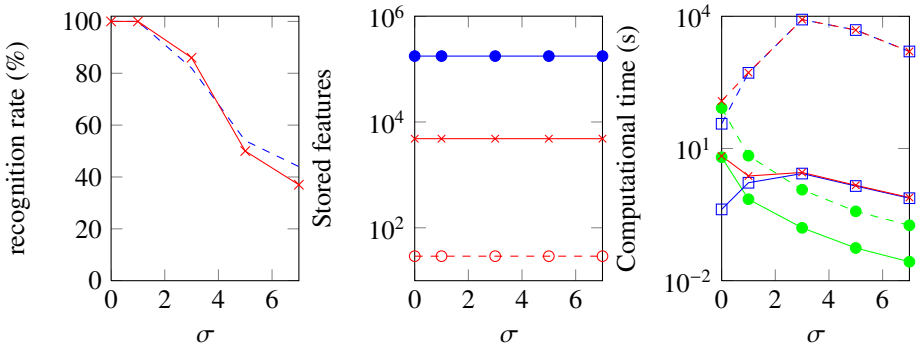
Figure 7. Performance evaluation results of our approach in terms of storage (continuous lines) and speed-up (dashed lines) gains. Gains: model storage space compression (theoretical —×—, real —○—), time performance (matching - -●- -)

6.1.2. Robustness to noise

In order to evaluate the quality of the poses recovered by the algorithm, in the presence of noise, we created an experimental scenario in simulation similar to the one referred in [1]. Each of the 200 synthetically generated scenes containing a single instance of a cyllinder or a cone was corrupted by different levels of additive gaussian noise, with standard deviation σ proportional to the model diameter diam(M). Both for the cyllinder and the cone, the shape parameters were constant. During recognition we chose 5% of the scene points as reference points by setting ξ to 0.05. A higher percentage would increase the robustness to noise but also the recognition runtime. In these tests we also measured the clustering time gains, given that more than one pose was retrieved from the Hough accumulator. As depicted in figure 8 we achieved speed-up gains up to 327x (6.41 against 2095.6 seconds) for the cylinder and 3336x (1.46 against 4871.2 seconds) for the cone, at the cost of residual losses in recognition rate. The main computational gains are in the clustering step, which are proportional to the levels of noise. This is due to the decreasing consensus for the ground truth pose in the Hough Accumulator which leads to pose hypotheses more scattered around the axis of symmetry, which are collapsed together with our method.



(a) Cyllinder.



(b) Cone.

Figure 8. Performance evaluation results of our approach in terms of robustness to sensory noise and performance. Left: Recognition rate (%) of our method (—×—) vs Drost *et al.* (- - -), Middle: model storage space of our method (theoretical - ⊖ -, real —×—) vs Drost *et al.* (—●—), Right: time performance of our method (matching —●—, clustering —□—, total —×—) vs Drost *et al.* (dashed lines)

6.2. Dealing with household objects

To evaluate the performance gains of our methodologies with non-parametric objects we selected a set of household objects which are common on everyday scenarios and repeated the aforementioned experiments. However, this time, both models and scene were subsampled before training and recognition with d_{step}.We assume that our database has only one object, as well as the generated scene. The objects are part of the ROS household objects library (see Figure 9(a)), on a random pose.

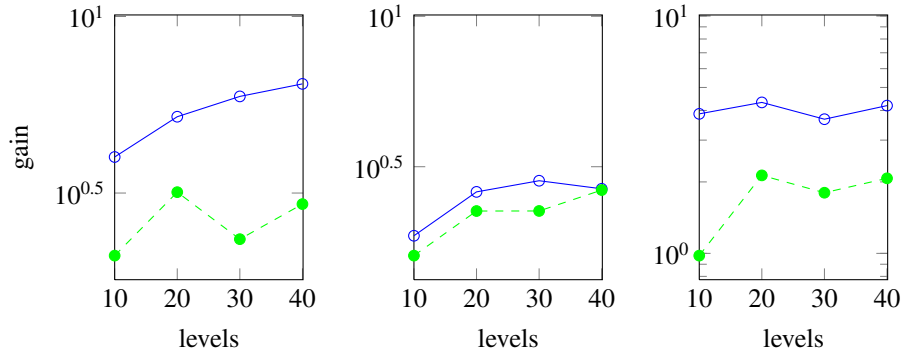
(a) (b)

Figure 9. (a) ROS Household object models. From left to right: coke can, champagne glass and cup. (b) Our method correctly detecting a Coca-Cola can. Figure best seen in color.

Our main goal was to measure if our algorithm was suitable for object models acquired by noisy 3D range scanners and applicable in noisy and cluttered scenarios.

6.2.1. Computational gains

In the following experiment we varied the number of points in the objects surface by chosing different sampling steps, s_{step}. Analogous to the experiment 6.1.2, this corresponded to varying the object radius, f_0 and height, $|L|$, at the same time. As shown in Figure 10 there is a strong correlation between the gain values and object shape. Constant radius shape such as the Coke can exhibit linear gains similar to the cyllinder, while less structured objects such as the champagne glass tend to have non-linear gains. The cup is in between the other two objects, being similar to a cyllinder. The shown computational gains of our methodologies allow its application in multi-object recognition and pose estimation scenarios [17].



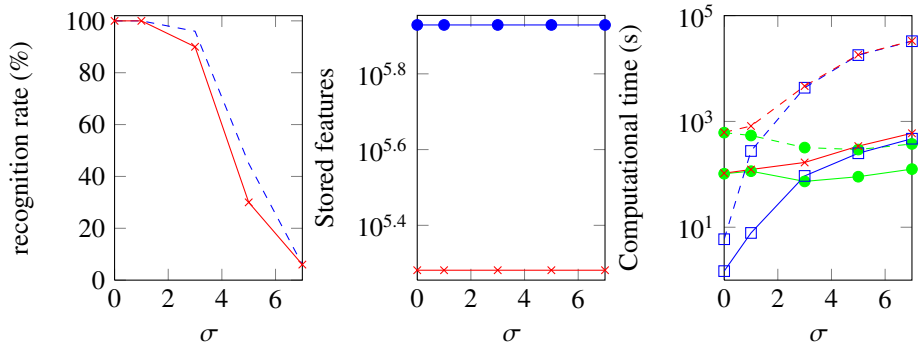
(a) Cup. (b) Coke can. (c) Champagne glass.

Figure 10. Performance evaluation results for real objects. Gains: model storage space compression (—○—), matching time performance gains (- ● -)

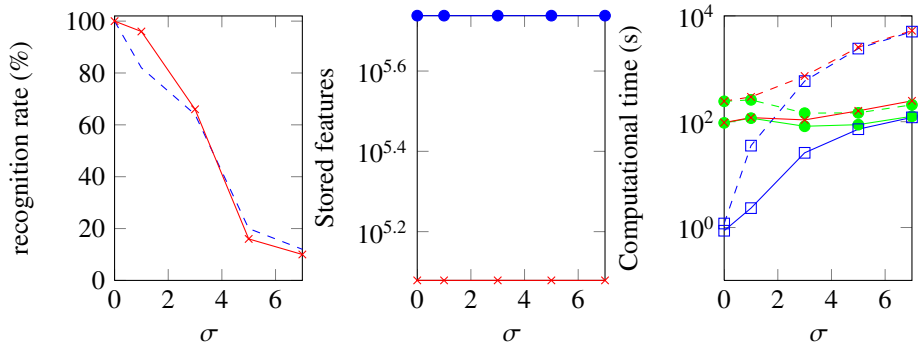
6.2.2. Robustness to noise and clutter

In order to test the algorithm's robustness to clutter we repeated the experimental scenario described in 6.1.2 but in this case each scene was corrupted by noise before the downsampling step. This way, the number of points in the scene increased directly proportionally with the noise which were considered as clutter.

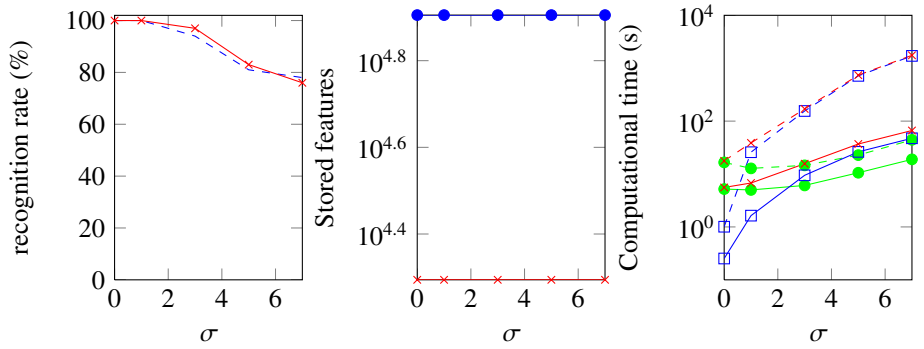
For the cup model we were able to discard 84.5% surflet pairs during the creation of the model description, and reduce the number of computations during pose recognition. As shown in Fig. 11, the recognition rate drops slightly for high levels of noise due to sampling effects, but the recognition time performance increases significantly. For $|S| \approx 5000$, our method achieves a recognition time 55.6 times faster than [1] (597.6 against 33205 seconds). However, the number of jointly represented surflet pairs depends heavily on the object geometric configuration. For objects whose shape has a smaller radius relative to the axis of symmetry, and also lower surflet density on the surface, less performance gains can be achieved. For the tests comprising the coke can model we were only able to discard 62.5% surflet pairs during the creation of the model description. Fig. 9(b) shows qualitative results of our method with real data, in a cluttered scenario. Overall, we were able to obtain major improvements on recognition speed without significant cost on recognition performance. For all objects, the time savings were mainly on the clustering step so the computational savings are mostly due to collapsing of pose hypotheses around the axis of rotational symmetry, during the pose clustering step.



(a) Cup.



(b) Coke can.



(c) Champagne glass.

Figure 11. Performance evaluation results of our approach in terms of robustness to sensory noise and clutter and performance. Left: Recognition rate (%) of our method (—×—) vs Drost *et al.* (- - -), Middle: model storage space of our method (—×—) vs Drost *et al.* (—●—), Right: time performance of our method (matching —●—, clustering —□—, total —×—) vs Drost *et al.* (dashed lines)

7. Conclusions

Symmetry can be exploited in order to reduce the storage of object models, which has an association with the gains in execution time during pose recognition. Given the axis of symmetry of an object, we show how to collapse surflet pair features, keeping the representative features. The theoretical analysis of storage gains shows the upper bounds of our method, which were verified experimentally on scans of actual objects. This reduced representation induces storage gains up to 100x and execution time savings up to 3000x the object recognition approach of [1]. On one hand, the storage gains depend on the surface properties of the object, so more complex geometries lead to low storage gains. On the other hand, the execution time savings are directly related to the level of noise and clutter in the

environment. It is important to remark that our method allows to have symmetric and non-symmetric objects in the database and the feature collapsing brings a more robust response of the classifier in the pose of symmetric objects.

References

[1] B. Drost, M. Ulrich, N. Navab, S. Ilic, Model globally, match locally: Efficient and robust 3d object recognition, IEEE Transactions on Computer Vision and Pattern Recognition (CVPR) (2010) 998 – 1005.

[2] D. H. Ballard, Generalizing the Hough transform to detect arbitrary shapes, Pattern Recognition 13 (2) (1981) 111–122.

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

[4] A. S. Mian, M. Bennamoun, R. Owens, Three-dimensional model-based object recognition and segmentation in cluttered scenes, IEEE Transactions on Pattern Anal. Mach. Intell 28 (2006) 1584–1601.

[5] R. P. de Figueiredo, P. Moreno, A. Bernardino, Fast 3D object recognition of rotationally symmetric objects, in: Pattern Recognition and Image Analysis - 6th Iberian Conference, IbPRIA 2013, Funchal, Madeira, Portugal, June 5-7, 2013. Proceedings, 2013, pp. 125–132.

[6] A. Andreopoulos, J. K. Tsotsos, 50 years of object recognition: Directions forward, Computer Vision and Image Understanding 117 (8) (2013) 827 – 891. doi:http://dx.doi.org/10.1016/j.cviu.2013.04.005.
URL http://www.sciencedirect.com/science/article/pii/S107731421300091X

[7] M. Kazhdan, T. Funkhouser, S. Rusinkiewicz, Symmetry descriptors and 3d shape matching, in: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, SGP '04, ACM, New York, NY, USA, 2004, pp. 115–123. doi:10.1145/1057432.1057448.
URL http://doi.acm.org/10.1145/1057432.1057448

[8] A. Martinet, C. Soler, N. Holzschuch, F. Sillion, Accurate detection of symmetries in 3d shapes, ACM Transactions on Graphics 25 (2) (2006) 439 – 464.
URL http://maverick.inria.fr/Publications/2006/MSHS06

[9] J. T. Schwartz, M. Sharir, Identification of partially obscured objects in two and three dimensions by matching noisy characteristic, Int. J. Rob. Res. 6 (2) (1987) 29–44. doi:10.1177/027836498700600203.
URL http://dx.doi.org/10.1177/027836498700600203

[10] U. Thomas, Stable pose estimation using ransac with triple point feature hash maps and symmetry exploration, in: Proc. of MVA2013 IAPR International Conference on Machine Vision Applications, 2013.

[11] N. J. Mitra, M. Pauly, M. Wand, D. Ceylan, Symmetry in 3d geometry: Extraction and applications, in: EUROGRAPHICS State-of-the-art Report, 2012. doi:10.1111/cgf.12010.
URL http://dx.doi.org/10.1111/cgf.12010

[12] A. Sehgal, U. Desai, 3d object recognition using bayesian geometric hashing and pose clustering, Pattern Recognition 36 (3) (2003) 765 – 780. doi:http://dx.doi.org/10.1016/S0031-3203(02)00102-4.
URL http://www.sciencedirect.com/science/article/pii/S0031320302001024

[13] I. Rigoutsos, R. Hummel, A bayesian approach to model matching with geometric hashing, Computer Vision and Image Understanding 62 (1) (1995) 11 – 26. doi:http://dx.doi.org/10.1006/cviu.1995.1038.
URL http://www.sciencedirect.com/science/article/pii/S1077314285710387

[14] H. J. Wolfson, I. Rigoutsos, Geometric hashing: An overview, IEEE Comput. Sci. Eng. 4 (4) (1997) 10–21. doi:10.1109/99.641604.
URL http://dx.doi.org/10.1109/99.641604

[15] E. Wahl, U. Hillenbrand, G. Hirzinger, Surflet-pair-relation histograms: A statistical 3D-shape representation for rapid classification, 3D Digital Imaging and Modeling, International Conference on (2003) 474doi:http://doi.ieeecomputersociety.org/10.1109/IM.2003.1240284.

[16] J. J. Craig, Introduction to Robotics: Mechanics and Control, 2nd Edition, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[17] R. P. de Figueiredo, P. Moreno, A. Bernardino, J. Santos-Victor, Multi-object detection and pose estimation in 3D point clouds: A fast grid-based bayesian filter, in: 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10, 2013, 2013, pp. 4250–4255.