Point Pair Features Based Object Detection and Pose Estimation Revisited

Tolga Birdal
Department of Computer Science, CAMP
Technische Universität München
tolga.birdal@tum.de

Slobodan Ilic
Siemens AG
Munich, Germany
slobodan.ilic@siemens.de

Abstract

We present a revised pipe-line of the existing 3D object detection and pose estimation framework [10] based on point pair feature matching. This framework proposed to represent 3D target object using self-similar point pairs, and then matching such model to 3D scene using efficient Hough-like voting scheme operating on the reduced pose parameter space. Even though this work produces great results and motivated a large number of extensions, it had some general shortcoming like relatively high dimensionality of the search space, sensitivity in establishing 3D correspondences, having performance drops in presence of many outliers and low density surfaces.

In this paper, we explain and address these drawbacks and propose new solutions within the existing framework. In particular, we propose to couple the object detection with a coarse-to-fine segmentation, where each segment is subject to disjoint pose estimation. During matching, we apply a weighted Hough voting and an interpolated recovery of pose parameters. Finally, all the generated hypothesis are tested via an occlusion-aware ranking and sorted. We argue that such a combined pipeline simultaneously boosts the detection rate and reduces the complexity, while improving the accuracy of the resulting pose. Thanks to such enhanced pose retrieval, our verification doesn't necessitate ICP and thus achieves better compromise of speed vs accuracy. We demonstrate our method on existing datasets as well as on our scenes. We conclude that via the new pipe-line, point pair features can now be used in more challenging scenarios.

1. Introduction

Many computer vision applications require finding the object of interest in either 2D or 3D scenes. The objects are usually represented with the CAD model or object's 3D reconstruction and typical task is detection of this particular object instance in the scenes captured with RGB/RGBD or a depth camera. Detection considers determining location of the object in the input image, usually denoted by the bounding box. However, in many scenarios, this information is not sufficient and complimentary 6DOF pose (3 degrees of rotation and 3 degrees of translation) is also required. This is typical in robotics and machine vision applications. Consequently, the joint problem of localization and pose estimation is much more challenging due to the high dimensionality of the search space. In addition, objects are often sought in cluttered scenes under occlusion and illumination changes and also close to real-time performance is usually required. In this paper, we rely only on depth data, which alleviates the problem of illumination changes. One of the most promising algorithms for matching 3D models to 3D scenes was proposed by Drost et al. [10]. In that paper, authors couple the existing idea of point-pair features (PPF), with an efficient voting scheme to solve for the object pose and location simultaneously. Given the object's 3D model, the method begins by extracting 3D features relating pairs of 3D points and their normals. These features are then quantized and

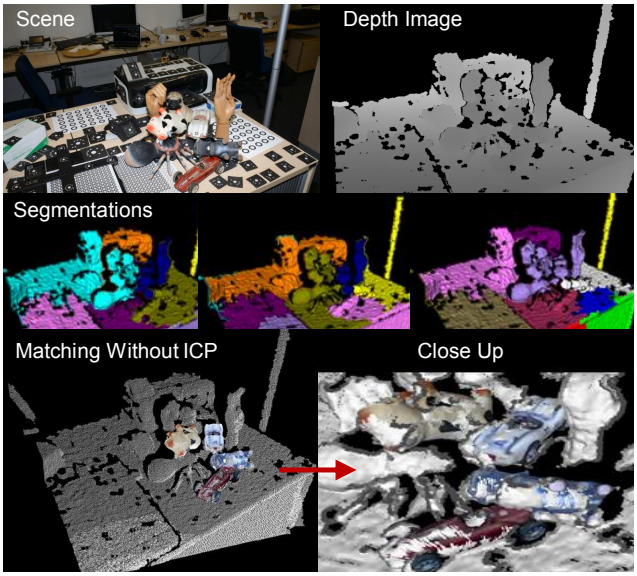

Figure 1. Outputs from our approach. Our segmentation aided matching has improved detection rates along with accurate poses.

stored in a hash table and used for representing the 3D model for detection. During run-time stage, the same features are extracted from a down-sampled version of a given scene. The hash-table is then queried per extracted/quantized feature and a Hough-like voting is performed to accumulate the estimated pose and location, jointly. In order to overcome complexity of the full 6DOF parametrization, assumption is made that at least one reference point in the scene belongs to the object. In that case if the correspondence is established between that reference point in the scene and one model point there, and if their normals are aligned, then there is only one degree of freedom, rotation around the normal, to be computed in order to determine the object's pose. Based on this fact, a very efficient voting scheme has been proposed. The great advantage of this technique lies in its robustness in presence of clutter and occlusion. Moreover, it is possible to find multiple instances of the same object, simply by selecting multiple peaks in the Hough space. While operating purely on 3D point clouds, this approach is fast and easy to implement.

Due to its pros, aforementioned matching method immediately attracted attention of scholars and was plugged into many existing frameworks. Moreno et al. used it to constrain a SLAM system by detecting multiple repetitive object models [21]. They also devise a strategy towards an efficient GPU implementation. Another immediate industrial application is bin picking, where multiple instances of the CAD model is sought in a pile of objects [13]. Besides, there is a vast number of robotic applications [3, 20] where this method has been applied.

The original method also enjoyed a series of add-ons developed. A majority of these works concentrated on augmenting the feature description to incorporate color [5] or visibility context [14]. Choi et al. proposed using points or boundaries to exploit the same framework in order to match planar industrial objects [6]. Drost et al. modified the pair description to include image gradient information [9]. There are also attempts to boost the accuracy and performance of the matching, without touching the features. Figueiredo et al. made use of the special symmetric object properties to speed up the detection by reducing the hash-table size [8]. Tuzel et al. proposed a scene specific weighted voting method by learning the distinctiveness of the features as well as the model points using a structured SVM [22].

Unfortunately, despite being well-studied, method of Drost et al. [10] is often criticized by high dimensionality of the search space [4], being sensitive to 3D correspondences [16], having performance drops in presence of many outliers, and low density surfaces [19]. Furthermore, the succeeding works report to significantly outperform the technique in many datasets [12, 4]. Yet, these methods work with RGB-D data, cannot handle occlusions and heavily depend on the post-processing and pose refinement.

In defense of the point pair features, we propose a revised pipeline, in which we address the crucial components of the framework. Instead of targeting the specific part of the original method as others, we revise the whole algorithm and draw a more elaborate picture of an improved object detection and pose estimation method.

Our approach starts by generating more accurate model representation relaying on PPFs. Since the normals are integral part of the PPFs, we compute them accurately by a second order approximation of the local surface patches. Giving different importance to the PPF is also important in building more reliable model representation. Unlike [22] where scene dependent PPF weighting has been performed, we rely on ambient occlusion maps [18] and associate weights to each model point, obtained via visibility queries over a set of rendered views. This is scene independent and causes a cleaner Hough space, eventually increasing the pose accuracy. During the online operation, the scene (depth map) is first segmented into multiple clusters, in a hierarchical fashion. In our context, coarse-to-fine/hierarchical segmentation refers to a set of partitioning varying from under- to over-segmentation. We detect objects in all segments, separately. Note that, while a variety of methods also segment the 3D model and use the parts [15], we deliberately avoid this, because the proposed matching is already robust to clutter and occlusion, which would be present in distinct clusters. By introducing a hierarchy of depth segment clusters with varying sizes, we deal with the segmentation errors. Processing disjoint segments inherently reduces the clutter and thus, the voting space gets much cleaner. We can then have a better detection rate, with a more accurate pose. Thanks to the same reason, we can detect small objects as well as large ones. This also improves the ability to find multiple objects without cluttering the Hough space. These benefits come with no additional computational cost. In fact, choosing reasonable segment sizes often reduce the run-time.

Our voting scheme makes effective use of the computed model weights and an enhanced Hough voting to achieve further accuracy of poses with more correct detections. Finally, all the estimated hypotheses in all segments are gathered and checked through an occlusion and clutter aware hypothesis verification. Moreover, thanks to entire procedure, the necessity of ICP pose refinement is minimized, further speeding up the real life applications. To accomplish all this, neither the feature representation nor the matching scheme is altered. This way, all the other methods, benefiting the similar framework can enjoy the contributions. Fig. 1 shows visual results from our ameliorated pipeline.

We evaluate our approach quantitatively and qualitatively on both synthetic and real datasets. We demonstrate the boosted pose accuracy along with the improvements in detection results and compare it to the state of the art. We show that the proposed pipeline yields more accurate poses, an

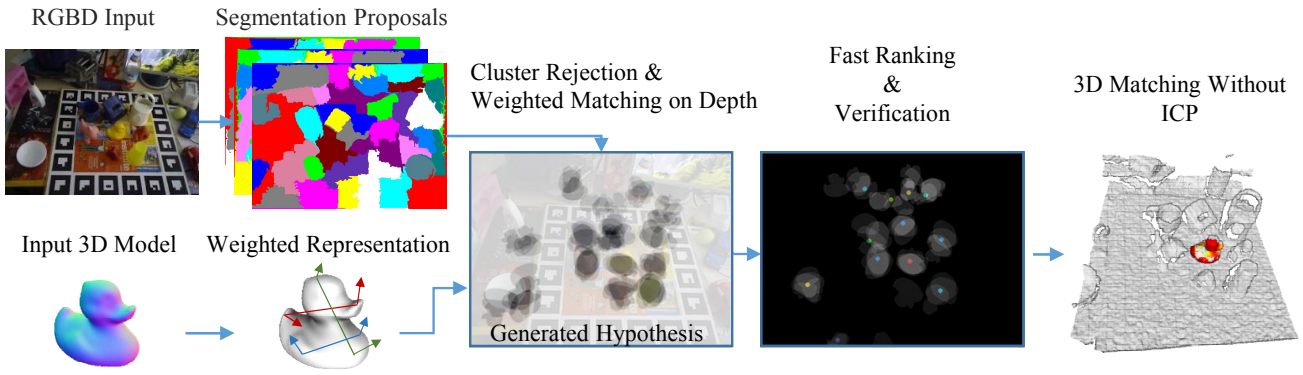

Figure 2. Illustration of the proposed pipeline. First input CAD models are trained using ambient occlusion maps as weighting. Each captured scene is first segmented into smaller regions and each region is matched to trained model. Per each segment, we retain many hypothesis and verify using the rendered CAD model. We rank the hypothesis according to the scores and reject the ones with low confidence.

increased detection rate and reduced complexity. The following sections are devoted to the related work, description of the proposed method and experiments.

2. Method

Our modeling and matching framework follows the one of Drost. et al. [10]. Our contributions lie in an enhanced model representation, along with the introduction of segmentation into the voting and a fast hypothesis verification. We will now describe this object detection pipeline, visualized in Fig. 2.

2.1. Model Representation

Given a mesh or a 3D point cloud, we represent the model by first computing the surface normals and the weights. Subsequently, the points are downsampled and a hash-table is created, storing the quantized pair features as well as the weights and the rotation angles to the ground plane. In this section, we give a detailed description of these steps.

2.1.1 Surface Features

Our features to describe the surface is called the point pair features (PPF). PPFs are antisymmetric 4D descriptors of a pair of oriented 3D points $\mathbf{m_1}$ and $\mathbf{m_2}$, constructed 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 $\mathbf{d}$ is the difference vector, $\mathbf{n_1}$ and $\mathbf{n_2}$ are the normals at $\mathbf{m_1}$ and $\mathbf{m_2}$. $\|\cdot\|$ is the Euclidean distance and in this paper, we always compute the angle between two vectors as follows:

$$\angle(\mathbf{v_1}, \mathbf{v_2}) = \tan^{-1}\left(\frac{\|\mathbf{v_1} \times \mathbf{v_2}\|}{\mathbf{v_1} \cdot \mathbf{v_2}}\right) \tag{2}$$

This doesn't suffer from numerical accuracy with small angles, and it is guaranteed to provide results in range $[0, \pi)$.

During the training stage, the vector $\mathbf{F}(\mathbf{m_1}, \mathbf{m_2})$ is quantized and indexed. In the test stage, same features are extracted from the scene and compared to the database.

2.1.2 Computing Model Normals

Since our features make heavy use of the surface normals, the method doesn't tolerate inaccurate estimations of those. Yet, for efficiency reasons, many algorithms resort to linear approaches, where the eigen-decomposition of the covariance matrix is utilized. However, the neighborhoods of local structures are not well represented by planar patches and thus first order approaches do not suffice in terms of accurately representing 3D models. A better approach is to use 2^{nd} order terms, where the convexity and concavity are also modeled. Even though computing 2^{nd} order approximations are costly for online phase, it is safe to use them in the offline stage. Hence, our objective is, to find the parameters of a second order polynomial, approximating the height field of the neighboring points, given a local reference frame [1]. Formally, given a point p_i on the set $P \in R^3$, MLS operates by fitting a surface of order m in a local K-neighborhood $\{p_k\}$ and projecting the point on this surface. Fitting is essentially a standard weighted least squares estimation of the polynomial surface parameters. The closer the neighbors are, the higher the contribution is. This is controlled by the weighting function $w(p_i) = exp(-\|p_i - p_k\|/2\sigma^2_{mls})$. The point p_i is then projected on the second order surface. This process is repeated for all points resulting in a smoothed point set with well defined normals. σ_{mls} can also be selected adaptively. The details are omitted, and we refer the reader to [1]. We show the effect of this scheme in Fig 3(c).

2.1.3 Weighting Model Points

The original technique treats all sampled points equally. Similar to [22], we argue that not all points carry the equal im-

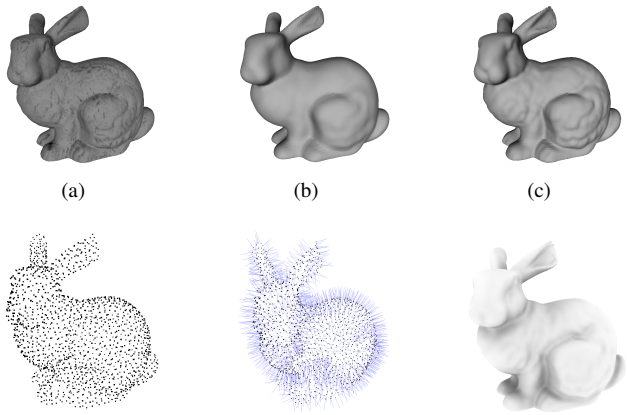

Figure 3. Model preparation. a) The original mesh. b) 1^{st} order MLS smoothing. c) 2^{nd} order approximation. d) Poisson Disk Sampling. e) Normals of sampled cloud. f) Occlusion map.

portance for matching. However, while authors in [22] are performing a scene dependent weighting and learning for a given task, we emphasize the necessity of a scene independent one. Unlike [22] however, our goal is not only to improve the detection rate, but to get better pose accuracy as well. For that, we are trying to focus on the visible surfaces of the object, where the normals are accurate and repeatability is better. Consequently, we base our weighting strategy on ambient occlusion maps [18]. Given a hemisphere Ω, the occlusion $A_{\mathbf{p}}$ at point $\mathbf{p}$ on a surface with normal $\mathbf{n}$ can be obtained by computing the integral of the visibility function V:

$$A_{\mathbf{p}} = \frac{1}{\pi} \int_{\Omega} V(\mathbf{p} \cdot \mathbf{w}) d\mathbf{w} \tag{3}$$

V is a dirac delta function, defined to be 1 if $\mathbf{p}$ is occluded in the direction of $\mathbf{w}$ and 0 otherwise. This integral is approximated via rendering the model from several angles and accumulating the visibility per each vertex. The cosine weighted average is then reported as the vertex-wise occlusion value. Based on $A_{\mathbf{p}}$, we propose to weigh the entries of the hashtable. Thus, given the hashtable bins, our weights are nothing but a normalized, geometric mean of $A_{\mathbf{m_r}}$ and $A_{\mathbf{m_i}}$. This way, the likelihood of using a potentially hidden point is reduced. In the experiments section, we show that even though this weighting doesn't necessarily increase the detection rate, it improves the accuracy of the resulting pose.

2.1.4 Global Model Description

Given the extracted PPF, the global description is implemented as a hashtable mapping the feature space to the space of point pairs. To do this, the distances and the angles are sampled in steps of d_{dist} and $d_{angle} = 2\pi / n_{angle}$ respectively. These quantized features are then used as keys to the hashtable. The pair features, which map to the same bucket are grouped together in the same bin, along with the weights. To reduce the computational complexity, a careful downsampling is required at this stage, which would respect the quantization properties. This requires all the points to have at least d_{dist} distances. We found out that using a Poisson Disk Sampling algorithm [7], this is ensured to an acceptable extent. This algorithm consists of generating samples from a uniform random distribution where the minimum distance between each sample is $2r$. This suggests that, a disk of radius r centered on each sample does not overlap any other disk, satisfying our quantization constraint.

2.2. Online Matching

Our input in runtime is only a depth image, typically acquired by a range sensor. First, the required normals are computed using SRI method proposed by Badino et al. [2]. This choice is motivated by the grid structure of the range image and the availability of the camera matrix. While not being identical to model normals, they are both accurate and computed quickly. The scene is then downsampled in a similar fashion to model creation. The triangulated depth points are then subject to a voting procedure, over the local coordinates. This section is devoted to the description of a coupled segmentation and voting approach, together with pose clustering and a hypothesis verification.

2.2.1 Hough Voting

Having a fixed scene point pair $(\mathbf{s_r}, \mathbf{s_i})$, we seek the optimal model correspondence $(\mathbf{m_r}, \mathbf{m_i})$ to compute the matching and 6DOF pose. Unfortunately, due to quantization, ambiguities and the noise in data, such assignment cannot be found by a simple scan. Instead, a voting mechanism, resembling Generalized Hough Transform is conducted. While votes can be cast directly on 6DOF pose space, Drost et al. [10] proposed an efficient scheme, reducing the voting space to 2D, using local coordinates. Whenever a model pair, corresponding to a scene pair is found, an intermediate coordinate system is established, where $\mathbf{m_i}$ and $\mathbf{s_i}$ are aligned by rotating the object around the normal. The planar rotation angle α_m for the model is precomputed, while the analogous for the scene point α_s is computed online. The resulting planar rotation angle around x-axis is found by a simple subtraction, $\alpha = \alpha_m - \alpha_s$.

An accumulator Acc is 2D voting space composed of $\mathbf{m_r}$ (the model index) and α. It collects votes for each scene reference point. $\mathbf{m_r}$ is already a discrete entity, while α is a continuous one, subject to discretization over the voting space. Unlike original method, we also maintain another accumulator Acc_{α} retaining the weighted averages of the corresponding α values, for each bin in Acc. This is done for the sake of not sacrificing further accuracy.

Notice that, because the pose parameters and the model correspondence are recovered simultaneously, an incorrect estimation of one, directly corrupts the other. This makes the algorithm sensitive to noisy correspondences. To compensate for these artifacts, we propose to vote with the computed weights in Section 2.1.3. Moreover, in presence of significant noise, correct correspondences can fall into neighboring bins, decreasing the evidence. Thus, when voting, the value of each bin is added also to the closest bins. Subsequently, we perform a subpixel maximization over the continuous variable α by fitting a second order polynomial to the k-nn of the discrete maximum and use α_k, obtained from the weighted averaging of the corresponding bins.

2.2.2 Matching Disjoint Segments

Our method employs a pre-segmentation to partition the scene into different meaningful clusters. Each cluster is then processed separately, having distinct Hough domains. This is different than previous works like [15], in which the model is also segmented.

We treat the depth image as an undirected graph $G = \{V, E\}$, with vertices $v_i \in V$ and edges $(v_i, v_j) \in E$. As a dissimilarity measure, each edge has a non-negative weight $w(v_i, v_j)$. We then seek to find a set of components $C \in S$, where S is the segmentation. The component-wise similarity is achieved via the weights of the graph. Felzenszwalb and Huttenlocher propose a graph theoretic segmentation algorithm, addressing a similar problem [11]. Their approach is designed for RGB images, whereas we adapt it to depth images. The algorithm uses a pair-wise comparison predicate (P), which is defined as:

$$P(C_1, C_2) = \begin{cases} 1, & \text{if } D(C_1, C_2) > M_{int}(C_1, C_2) \leq 0 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Here, $D(C_1, C_2)$ is the difference between components and defined as the minimum weight edge:

$$D(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E} w((v_i, v_j)) \tag{5}$$

where the minimum internal difference M_{int} equals:

$$M_{int}(C1, C2) = min(Int(C_1) + \tau(C_1), \\ Int(C_2) + \tau(C_2)) \tag{6}$$

with $Int(C) = max_{e \in MST(C,E)}\, w(e)$ and MST being the minimum spanning tree of the graph. The threshold function $\tau(C) = k/|C|$ exists to compensate for small components with k being a constant and $|C|$, the cardinality of C. Note that, smaller components are allowed when there is a sufficiently large difference between neighboring components. The segmentation $S = \{C_1 \dots C_s\}$ can be efficiently found by union-find algorithm. The adaptation to depth images is done by designing the weights. We use the local smoothness of the surface normals along with the proximity of the neighboring points. Segmentation weights are defined as:

$$w(v_i, v_j) = \|v_i^n - v_j^n\| \angle(n_i, n_j) \tag{7}$$

where (v_i^n, v_j^n) is the edge in the normalized coordinates.

While this approach generates a descent segmentation, we do not need to process every segment. In fact, many of these segments might lack sufficient geometry, or can be very small / large, or be coplanar. For that, we apply a filtering. We first remove segments which have a lot of undefined depth values. Then, the segments not obeying the size constraints are filtered out. Finally, we evaluate the linearity of the segments. Because, we have the set of normals $\{\mathbf{N}_j^i\}$ defined for each point $\mathbf{n}_j$ of cluster i, this procedure is simply applying a threshold τ_c over the deviation from the mean normal, computed as:

$$\sigma(\mathbf{N}^i) = \frac{1}{|\mathbf{N}^i|} \sum_{j=1}^{|\mathbf{N}^i|} (\angle(\mathbf{n}_{ij} \bar{\mathbf{n}}_i))^2 \tag{8}$$

where $\{\bar{\mathbf{n}}^i\}$ is the set of mean cluster normals (see Section 2.2.3). The clusters which satisfy the condition $\sigma(\mathbf{N}^i) < \tau_c$ are early-rejected. In our experiments, we use a coarse-to-fine (under to over) set of segmentations. Worst case resorts to using the whole scene, while difficult cases, such as small objects are found in coarser levels. Each segment is processed disjointly. We then verify the detected poses as in Section 2.2.4. This reduces the clutter and decreases the number of scene points sought. Because less clutter implies more relevant votes, it *demystifies* the Hough space and eases the maximization. Then, the accuracy of the resulting pose, as well as the detection rate increases. Besides, reduction in computational cost comes as a by-product. We will discuss this more in Section 3.3.

2.2.3 Pose Clustering and Averaging

As a result of Hough voting on disjoint clusters of segments, we obtain, for each scene reference, a pose candidate. These candidates are clustered for each segment separately. An agglomerative clustering coupled with a good pose averaging scheme is found to be reasonably accurate. Our clustering is similar to [6]. Initially, the candidate poses are sorted by the number of votes. The highest vote creates the first cluster. We only create a new pose cluster, if the candidate pose deviates significantly from the existing clusters. Each time a pose is added to a cluster, the cluster mean is updated and the cluster score is incremented.

The described clustering requires a pose averaging step, which visits each pose candidate, once. For the sake of accuracy, it is prohibitive to use rotation matrices as they cannot

be directly averaged. On the other hand, techniques involving Lie algebra are generally found to be computationally expensive [6]. Therefore, we employ a quaternion based fast averaging technique as proposed in [17]. Given $\{\mathbf{q}_i\}$, a set of quaternions, we form the weighted dot product matrix:

$$\mathbf{A} = \frac{1}{n_q} \sum_{i=1}^{n_q} w_i^{\mathbf{q}} (\mathbf{q}_i^T \cdot \mathbf{q}_i) \tag{9}$$

where n_q is the number of poses, and w_i^q is the number of scene points found on the model, given the pose $\mathbf{q}_i$. The mean quaternion $\mathbf{q}_{avg}$ is given by the eigenvector $\mathbf{e}_{max}$ corresponding to the maximum eigenvalue of $\mathbf{A}$, λ_{max}.

In our trials, we found out that when the weights are chosen appropriately, as explained, this averaging is twice as better as the naive mean. For this reason, in all of the experiments, we will be using this method to obtain the pose clusters.

2.2.4 Hypotheses Verification

Our method generates a set of hypotheses per each object, with reasonable pose accuracy. Yet, such a huge set of hypotheses demands an efficient verification scheme. Typical strategies, such as Hinterstoister et al. [12], either put ICP in the loop, whereas, for our method, the pose accuracy is sufficient for ICP-less evaluations.

To verify and rank the collected hypothesis, we categorize the visible space into 3: Clutter (outlier) S_c, occluders S_o and points on the model S_m according to the following projection error function:

$$E_h(p, m) = D_p - \Phi(p|\mathbf{M}, \Theta_h, \mathbf{K}) \tag{10}$$

Φ selects the projection of the model points M corresponding to pixel p, given a camera matrix K and the pose parameters Θ_h for hypothesis h. The classification for a given valid point p is then conducted as:

$$p \in \begin{cases} S_m, & \text{if } |E_h(p,m)| \leq \tau_m \\ S_o, & \text{if } E_h(p,m) \geq \tau_o \\ S_c, & \text{otherwise} \end{cases}$$

subsequently, the score for a given hypothesis is:

$$S_h = \left(1 - \frac{|p \in S_o|}{N_m}\right) \cdot \frac{|p \in S_m|}{N_m - |S_o|} \tag{11}$$

where N_m is the number of model points on valid region of the projection $\Phi(p|\mathbf{M}, \Theta_h, \mathbf{K})$. The thresholds τ_m and τ_o depend on the sensor and are relaxed, due to the missing points not acquired by the sensor. Similarly, we include the check for coinciding normals using Eq. 2. Luckily, these scores can be computed very efficiently using vertex buffers and Z-buffering on the GPU. Instead of transforming the model with the given pose, we use the Θ^{-1} to update the current camera view. Thanks to the accuracy in pose estimation, the ICP is not a strict requirement of this stage. In fact, frequently, the verification is ICP-free. Retrieving the top N_{best} poses finalizes our object detection pipeline.

This metric favors less occluded and less cluttered matches, having more model points with consistent normals. Yet, in our experiments, we found that use of filtered clusters increases the chances of hypothesizing a descent pose, which is only at seldom missed by the verification.

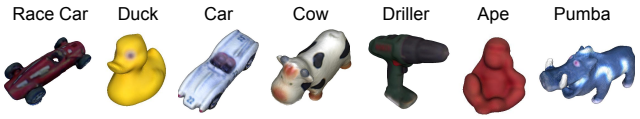

Figure 4. 3D models of some of the objects used in our experiments.

3. Results

We evaluate our method quantitatively and qualitatively on synthetic and real datasets. For all experiments, the points are downsampled with a distance of 3% times the diameter. Normal orientation is sampled for $n_{angle} = 45$. Some models used in our experiments are shown in 4.

3.1. Synthetic Experiments

We synthetically evaluate the accuracy of our pose estimation. To do that, we virtually render multiple CAD models in 3D scenes along with artificial clutter, also generated by other CAD models. To match the reality, our models are the reconstructions of real objects taken from ACCV3D dataset [12]. We synthesize 162 camera poses over the full sphere for each object. Specifically, our objects are PHONE, APE, DUCK, IRON, DRILLER, CAR and BENCHWISE. The chosen models cover a variety of geometrical structures. This corresponds to 1134 point cloud scenes, all of which had apriori additive Gaussian noise. Because at this point we are concerned for the pose accuracy, no segmentation is applied and we record the rotational and translational errors for correct detections. At this stage, an object is marked detected if the resulting pose is close to the ground truth pose. We set the threshold to 10% of the object diameter for detection and 10° for rotation. Fig. 5(a) and 5(b) depict the results obtained from pose estimation. It is seen that, for many of the objects, our rotational component is twice as more accurate as Drost et al. [10]. Since the translation is also computed from rotations and the matching model component, there is also similar refinement in translational accuracy.

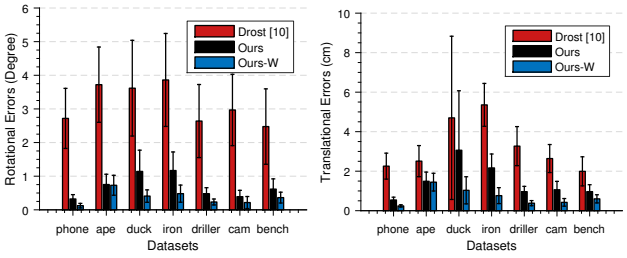

Figure 5. Comparison of the voting strategies (Ours-W is the weighted variant). **a)** Rotational errors. **b)** Translational errors.

Table 1. Detection results on ACCV3D for different objects.

	LineMod	Drost et al.	Ours
ape	42.88%	65.54%	**81.95%**
cam	68.78%	84.92%	**91.00%**
cat	35.62%	87.30%	**95.76%**
driller	51.52%	81.06%	**81.22%**
iron	35.22%	87.06%	**93.92%**
Average	46.80%	81.18%	**88.77%**
Avg. Runtime	119ms	6.3s	2.9s

3.2. Real Scenes

We evaluated our approach in terms of pose accuracy and detection rate on real datasets. For quantitative comparisons, ACCV3D dataset from [12] is used. This dataset is now standard in object detection and many early works evaluated their methods on it. The package includes 15 non-symmetric objects appearing in $\sim$ 1100 scenes per object. Each scene of an object is cluttered with the other objects. In none of the scenes, the objects are subject to heavy occlusion.

Given a depth image, our algorithm not only detects the object, but also estimates the pose. Thus, we compare our method against LineMod [12], and Drost et al. [10], which are the state of the art in pose estimation using only depth images. Yet, it is worth paying attention to the following:

LineMod [12] is designed for multi-modal features and incorporates color information. Yet, we favor a fair comparison to our method by only using depth cues. Naturally, this has a negative impact on LineMod's performance, as it relies significantly on the color information. We, nevertheless, include LineMod as a baseline. Unlike the experiments in [12], we do not tune the parameters for each object or scene when using our method. While carefully tuning parameters sacrifices computational time for detection accuracy, it is cumbersome and unfair. LineMod without ICP has very low detection rates, as it is integrated in its matching pipeline. Thus, we use it in the detection stage. Yet, the reported poses are solely obtained by template matching and not ICP. In the experiments with LineMod original implementation of the authors was used, but only with depth information and without post-processing. Unlike LineMod [12], Drost et al. [10] don't necessarily require the refinement. For this reason, neither our poses nor the poses for Drost. et al. [10] are refined and re-scored. We find this strategy inevitable to reason about our pose results and not the results of ICP. For all these reasons, our results will differ from the original ones, but will be consistent along the experimentation.

The detection rates are shown in Table 1, for a subset of objects in ACCV3D. We select a subset due to either lack of accurate CAD models or large performance drops for LineMod [12] (which would be unfair to show). For our datasets, only 2-3 segmentation levels per scene were sufficient. In harder cases, one might use more maps to cover for larger variation. It is clearly seen that our method outperforms both methods. Our detection rates never fall below Drost. et al. [10], as the worst case converges to full matching. Thanks to meaningful segments, using all the points as a single cluster is highly unlikely. On the average, we get 7% more, although this dataset is not cut for our method. It is noteworthy that our improvements in the detection are more significant for small objects (which are hard to spot in clutter and occlusion), while the pose accuracy is more significant in large objects with varying surface characteristics. Nevertheless, we realize increased accuracy in both pose and detection rate regardless of the object size. Also note that, LineMod [12] uses only a hemisphere, whereas we recover the full pose (see Fig. 1).

Next, we evaluate the pose accuracy, on the same dataset. To do that, we first define a new error function, which is free of the point correspondences but rather depends directly on the pose parameters:

$$\mathbf{err}_i = \mathbf{d}_\theta(\mathbf{M}^0_{marker}(\mathbf{M}^i_{marker})^{-1}\mathbf{M}^i_{obj}, \mathbf{M}^0_{obj}) \quad (12)$$

$\mathbf{d}_\theta$ is a function returning an error vector $\mathbf{err}$ with angular and translational components. $\mathbf{M}^i_{obj}$ is the object pose at frame i, where as $\mathbf{M}^i_{marker}$ is the pose of the marker board for the same frame. The overall error per object is simply reported as the average pose error in all test frames, i.e average of the set $\{\mathbf{err}_i\}$. This metric transfers each estimated pose to the first frame and computes a pose error between detected object pose transformed to the first frame and ground truth pose in the first frame. We evaluate the error on CAT, DUCK and CAM objects. After transferring each to the initial frame, we perform an ICP and report in Fig. 7 $\{\mathbf{err}_i\}$ convergence from our detected pose. After the same number of ICP iterations we have lower rotation/translation error and also because of the better detected pose we need less iterations to achieve better accuracy after ICP refinement.

Finally, Fig. 6 visualizes the results of our method both on a self built setup (Fig. 6(a)) and on ACCV3D dataset using CAT object (Fig. 6(b)). We show that both the detections and the pose accuracy is visually better than the antecedents.

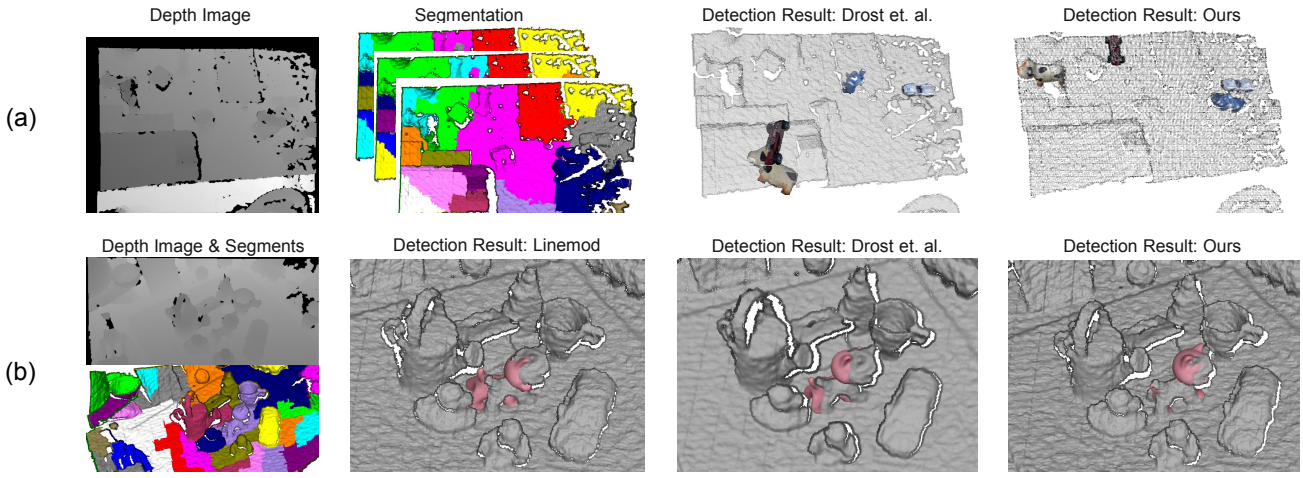

Figure 6. Qualitative results. **a)** Detection results in our data, with presence of small objects in long-range Kinect scans. **b)** Pose estimation results on ACCV3D dataset [12]. The accuracy in our poses is even visually distinguishable.

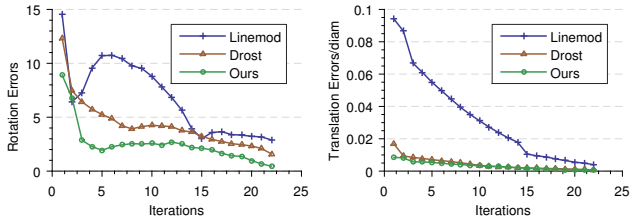

Figure 7. Pose errors on ACCV3D dataset as ICP iterates. Rotations are in degrees, while for translation, y-axes are normalized with model diameter.

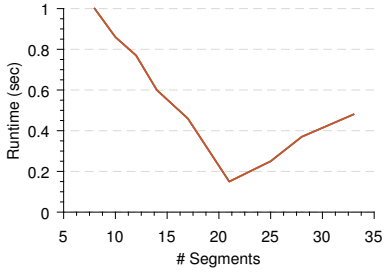

Figure 8. Effects of number of segments on runtime.

3.3. Runtime

Complexity Analysis One drawback of PPF matching is the combinatorial pairing approach. One way to overcome this problem is by pairing the scene points very sparsely, which is suboptimal. Let M be the number of scene reference points $\mathbf{S_r}$, and N denote all the paired points in the scene. The pairing $(\mathbf{S_r}, \mathbf{S_i})$ then creates a complexity of $O(MN) = O(N_x^2 N_y^2)$, where (N_x, N_y) denotes the dimensions (invisible points and hash-table search are excluded).

If we are to segment the image into K clusters, the average number of $\mathbf{S_r}$ per cluster as well as the number of paired points are reduced to $\frac{M}{K}$ and $\frac{N}{K}$, respectively, resulting in an overall complexity of $O(\frac{MN}{K^2})$ per cluster. The overall average time complexity is reduced to $O(\frac{MN}{K})$. If we agree to keep the same complexity, we can now vote for more points. Instead, we prefer to use a set of segmentation with different segment sizes, resulting in more clusters.

Performance We first report the runtime of our algorithm on ACCV3D dataset in Table 1. Note that even though we get 2x speed-up over Drost et al. [10], this is less than the theoretical possibility. In fact, this is due to the trade-off of obtaining superior detection rate and accuracy by using a sequence of segmentation and more scene points w.r.t. the original algorithm.

As explained, the performance is largely affected by the size and the number of the segments. Next experiment targets this effect. We take 3 arbitrary models present in 1000 images, where the objects of interest were CAR, APE and DUCK. We sample ~ 900 model points. In each scene, we seek for the minimal number of scene points to pair for a correct match and use that to record the timings. This is in order to make sure that every trial actually results in a correct pose. We plot the segment size vs speed relation in Fig. 8. These timings exclude the data acquisition. Note that there is an optimum point (correct segmentation), which generally depends on the scene. For this experiment, we could reduce the matching time to 170 ms by just using $\frac{1}{50^{th}}$ of the scene points. However, typically, suboptimal choices already allow a descent reduction of computational time, as we do not rely on the precision of the segmentation. This means that, being able to use more clusters, decreases the demand on the sampling and one could use much less scene

points to obtain a successful match. Naturally, increasing the segment sizes, reduces the number of clusters and thus the performance converges to that of the original algorithm.

4. Conclusion & Future Work

We revised a complete pipeline for robust 3D object detection based on point pair features and Hough-like voting of Drost et al. [10], which enjoys segmentation proposals together with strengthened representation, voting mechanisms and hypothesis verification. We compared our approach to the state of the art methods and showed that our pipeline produced better detection rate and pose accuracy than the state of the art with reduced computational complexity. Our technique is especially suited for detection of objects with varying sizes in clutter and occlusion.

In the future work, we plan to investigate scalable joint detection and pose estimation using segments and adaptive sampling strategies.

References

[1] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Computing and rendering point set surfaces. *Visualization and Computer Graphics, IEEE Transactions on*, 9(1):3–15, 2003.

[2] H. Badino, D. Huber, Y. Park, and T. Kanade. Fast and accurate computation of surface normals from range images. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3084–3091. IEEE, 2011.

[3] M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mosenlechner, D. Pangercic, T. Ruhr, and M. Tenorth. Robotic roommates making pancakes. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 529–536. IEEE, 2011.

[4] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6d object pose estimation using 3d object coordinates. In *Computer Vision–ECCV 2014*, pages 536–551. Springer, 2014.

[5] C. Choi and H. I. Christensen. 3d pose estimation of daily objects using an rgb-d camera. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 3342–3349. IEEE, 2012.

[6] C. Choi, Y. Taguchi, O. Tuzel, M.-Y. Liu, and S. Ramalingam. Voting-based pose estimation for robotic assembly using a 3d sensor. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1724–1731. IEEE, 2012.

[7] M. Corsini, P. Cignoni, and R. Scopigno. Efficient and flexible sampling with blue noise properties of triangular meshes. *Visualization and Computer Graphics, IEEE Transactions on*, 18(6):914–924, 2012.

[8] R. P. de Figueiredo, P. Moreno, and A. Bernardino. Fast 3d object recognition of rotationally symmetric objects. In *Pattern Recognition and Image Analysis*, pages 125–132. Springer, 2013.

[9] B. Drost and S. Ilic. 3d object detection and localization using multimodal point pair features. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, pages 9–16. IEEE, 2012.

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

[11] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[12] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Computer Vision–ACCV 2012*, pages 548–562. Springer, 2013.

[13] D. Holz, M. Nieuwenhuisen, D. Droeschel, J. Stückler, A. Berner, J. Li, R. Klein, and S. Behnke. Active recognition and manipulation for mobile robot bin picking. In *Gearing Up and Accelerating Cross-fertilization between Academic and Industrial Robotics Research in Europe:*, pages 133–153. Springer, 2014.

[14] E. Kim and G. Medioni. 3d object recognition in range images using visibility context. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3800–3807. IEEE, 2011.

[15] J. Lam and M. Greenspan. 3d object recognition by surface registration of interest segments. In *3D Vision-3DV 2013, 2013 International Conference on*, pages 199–206. IEEE, 2013.

[16] M.-Y. Liu, O. Tuzel, A. Veeraraghavan, Y. Taguchi, T. K. Marks, and R. Chellappa. Fast object localization and pose estimation in heavy clutter for robotic bin picking. *The International Journal of Robotics Research*, 31(8):951–973, 2012.

[17] F. L. Markley, Y. Cheng, J. L. Crassidis, and Y. Oshman. Averaging quaternions. *Journal of Guidance, Control, and Dynamics*, 30(4):1193–1197, 2007.

[18] G. Miller. Efficient algorithms for local and global accessibility shading. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 319–326. ACM, 1994.

[19] M. Mohamad, D. Rappaport, and M. Greenspan. Generalized 4-points congruent sets for 3d registration. In *3D Vision (3DV), 2014 2nd International Conference on*, volume 1, pages 83–90, Dec 2014.

[20] M. Nieuwenhuisen, D. Droeschel, D. Holz, J. Stuckler, A. Berner, J. Li, R. Klein, and S. Behnke. Mobile bin picking with an anthropomorphic service robot. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2327–2334. IEEE, 2013.

[21] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1352–1359. IEEE, 2013.

[22] O. Tuzel, M.-Y. Liu, Y. Taguchi, and A. Raghunathan. Learning to rank 3d features. In *Computer Vision–ECCV 2014*, pages 520–535. Springer, 2014.