Learning to Rank 3D Features[⋆]

Oncel Tuzel, Ming-Yu Liu, Yuichi Taguchi, and Arvind Raghunathan

Mitsubishi Electric Research Labs (MERL), Cambridge, MA, USA

Abstract. Representation of three dimensional objects using a set of oriented point pair features has been shown to be effective for object recognition and pose estimation. Combined with an efficient voting scheme on a generalized Hough space, existing approaches achieve good recognition accuracy and fast operation. However, the performance of these approaches degrades when the objects are (self-)similar or exhibit degeneracies, such as large planar surfaces which are very common in both man made and natural shapes, or due to heavy object and background clutter. We propose a max-margin learning framework to identify discriminative features on the surface of three dimensional objects. Our algorithm selects and ranks features according to their importance for the specified task, which leads to improved accuracy and reduced computational cost. In addition, we analyze various grouping and optimization strategies to learn the discriminative pair features. We present extensive synthetic and real experiments demonstrating the improved results.

Keywords: 3D pose estimation, feature selection, max-margin learning.

1 Introduction

Three dimensional object recognition and pose estimation tasks require identification and matching of scene measurements to the known object model. Various methods have been proposed. Many of them are based on selecting salient points in the 3D point cloud and using feature representations that can invariantly describe regions around the points [29,15,27,8,2,1,26]. These methods are known to produce successful results when the 3D object shape is rich and detailed, and the scene measurements are of high resolution and less noisy. However, under less ideal conditions their accuracy degrades rapidly. More importantly, partial measurements due to self occlusion and contaminating background clutter make a detailed region representation unavailable. Recently, using a set of simple pair features [33] has been shown to be useful for detection and pose estimation tasks. Drost *et al.* [7] used pairs of oriented points on the surface of the object in a voting framework. Even though the descriptor associated with the pair feature is not very discriminative, by accumulating evidence from a large number of pairs, it produces accurate results and becomes robust against moderate occlusion and background clutter. This

[⋆] Electronic supplementary material -Supplementary material is available in the online version of this chapter at `http://dx.doi.org/10.1007/978-3-319-10590-34`. Videos can also be accessed at `http://www.springerimages.com/videos/978-3-319-10589-5`

D. Fleet et al. (Eds.): ECCV 2014, Part I, LNCS 8689, pp. 520–535, 2014.

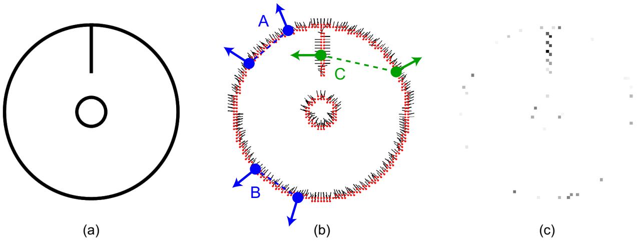

Fig. 1. Learning to rank features. (a) Input shape. Only the line at the center encodes orientation information. (b) Feature representation of the points using 2D location and orientation. Two points on the circle do not encode orientation information (the pairs A and B are indistinguishable). The orientation can be determined if one of the points is selected on the line as in the pair C. (c) Points selected by our learning algorithm are highlighted according to their weights. Primarily the points on the line are selected.

framework is also known to benefit from a carefully designed hashing and voting scheme, similar to geometric hashing [18], making the algorithm efficient.

Our observation is that not all the pairs of points have similar discriminative power or repeatability. In fact, certain features do not carry relevant information to the task in hand. For example, let us use the simple two dimensional shape shown in Figure 1(a), and define the task to be estimating the 2D pose (rotation and translation) of this shape in a scene. The feature is defined to be 2D point pair and their gradient orientations (Figure 1(b)). Under this setup, any point pair that is selected on the same circle (e.g., the pairs A and B) does not contain any information about the orientation of the shape, since they can simply match to many other pairs on the same circle. Hence a pair on the same circle is useless for the pose estimation task and should be identified and removed to improve both efficiency and robustness.

As mentioned above and more in [9,23,20], there are many methods to detect salient points on a scene or an object. Although these methods achieve major success for many tasks, it is unrealistic to expect a fixed recipe to be satisfactory for all the shapes and tasks. Instead, we propose a machine learning based approach to identify and rank important model features directly from the available data (image or a 3D model of an object) for the given task. A set of learned weights using our algorithm for the shape in Figure 1(a) is shown in Figure 1(c) with intensity values representing weights of the model features. As shown, our approach finds important points on the shape (the line segment that resolves rotation ambiguity) while removing unnecessary (on the circle) and unreliable points (points on the tip of the line has unreliable gradient orientations across synthesized rotations of the shape).

1.1 Contributions

The contributions of the paper are as follows:

- We present a max-margin learning framework for selecting and ranking discriminative features for voting-based 3D pose estimation and object recognition tasks.

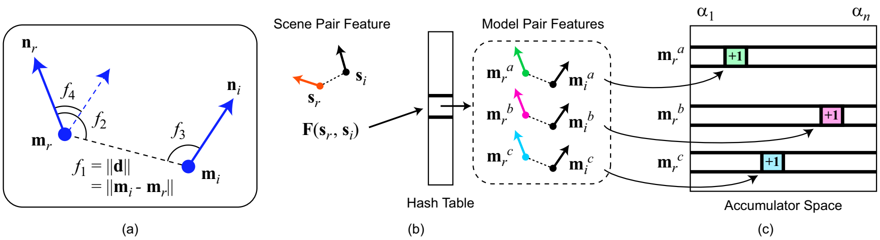

Fig. 2. Overview of the voting-based pose estimation algorithm [7,6]. (a) A point pair feature is defined using two oriented points. The 4D descriptor is defined by the distance between the points and angles between their orientations and the displacement vector. (b) Point pair features from the model are stored in a hash table using their descriptors as the keys. Each point pair feature from the scene is matched with multiple pair features from the model using the hash table. (c) Each matched model pair feature votes for an entry in a 2D accumulator space, corresponding to a particular pose.

- The learning task does not require manual labor for data collection and annotation, and uses data synthesized through rendering object models.
- The proposed approach enables various weighted voting strategies and can combine multiple different feature types.
- We present large scale real experiments on two challenging datasets and demonstrate improved results over the state-of-the-art.

2 Pose Estimation Using Point Pair Features

Figure 2 shows an overview of the voting-based pose estimation algorithm using point pair features [7,6]. A point pair feature consists of a pair of oriented points, denoted by their positions $\mathbf{m}_r$ and $\mathbf{m}_i$ and orientations $\mathbf{n}_r$ and $\mathbf{n}_i$ (e.g., normals or boundary directions). We refer to the first point $\mathbf{m}_r$ as the *reference* point. The descriptor of a point pair feature is given by

$$\mathbf{F}(\mathbf{m}_r, \mathbf{m}_i) = (f_1, f_2, f_3, f_4) = (\|\mathbf{d}\|_2, \angle(\mathbf{n}_r, \mathbf{d}), \angle(\mathbf{n}_i, \mathbf{d}), \angle(\mathbf{n}_r, \mathbf{n}_i)), \qquad (1)$$

where $\mathbf{d} = \mathbf{m}_i - \mathbf{m}_r$ and $\angle(. \;, \;.)$ denotes the acute angle between two vectors, as shown in Figure 2(a). Note that this descriptor is pose invariant.

To recover the object pose, pair features from the scene are matched with those from the model. For efficiency, all the model pair features are stored in a hash table by using their quantized descriptors as the hash keys in an offline process. During pose estimation, a scene reference point is sampled and paired with another scene point to form a point pair feature. Its descriptor is then used to retrieve matched model pair features using the hash table, as shown in Figure 2(b). Each of the retrieved model pair feature in the hash bin votes for an entry in a 2D accumulator space, as shown in Figure 2(c). Each entry in the accumulator space corresponds to a specific object pose. After votes are accumulated for the sampled reference scene point by pairing it with multiple sampled scene points, poses supported by a certain number of votes are retrieved. The process is repeated for different reference scene points. Finally, clustering is performed to the retrieved poses to collect support from several reference points.

Drost *et al.* [7] used points on object surfaces and performed voting using surface-to-surface (S2S) pair features. Choi *et al.* [6] extended it using points on object boundaries and defined surface-to-boundary (S2B) and boundary-to-boundary (B2B) pair features. They showed that for planar objects, pair features including boundary points encode more information for pose estimation and provide better performance. However, (1) they were limited to the use of the individual pair features and (2) they did not provide how to select an optimal set of pair features given an object. Our framework allows us to combine different pair features by jointly learning optimal weights for all pair features.

3 Learning to Rank Features

The predicted pose of the object is given by the pose hypothesis that receives the maximum number of votes among all pose hypotheses. The goal of the learning is to select and weight the features to improve the chance that the correct hypothesis receives more votes than the others.

3.1 Weighted Voting

There are several different ways of weighting the features. The simplest form is based on weighting each model point pair with a different weight, and any scene point pair that maps to a given model point pair casts a vote equal to this weight. Although this weighting scheme is general, learning would be severely underdetermined due to high dimensional weight space.

Alternatively, we can group the features and constrain all the features within a group to have the same weight. One such strategy is *weighting hash table bins*: A weight is defined for each entry of the hash table and any scene pair feature that maps to the same hash table bin votes with the same weight. Since the point pairs are grouped into clusters that are mapping to the same hash table entry, the dimension of the weight space is reduced to the number of hash table bins. Another important advantage of this method is that it is possible to learn a weight vector that is sparse. Then the voting algorithm immediately removes any scene feature mapping to a bin with zero weight, improving the speed of the algorithm. A second grouping strategy is *weighting model points*: A weight is defined for each model point and any scene pair that maps to (either first or second point of the pair) this model point votes with this weight. This approach reduces the dimension of the weight space by an order, allowing efficient learning, and directly identifies important points on the model surface.

Given a 3D point cloud of a scene containing the target object, let S be the set of all scene pair features that are computed from this scene. Without loss of generality here we use the *weighting hash table bins* scheme. Let $\mathbf{y} \in SE(3)$ be a pose hypothesis. The corresponding data vector $\mathbf{x}_{\mathbf{y}} \in \mathbb{R}^M$ is given by the mapping $\mathbf{x} = \Phi(S, \mathbf{y})$[1] where M is the size of the hash table. Each dimension of the data vector, x^m, counts the number of times a scene pair feature, $s \in S$, maps to the hash table entry m and votes for the pose $\mathbf{y}$:

$$x^m = \Phi^m(S, \mathbf{y}) = \sum_{s \in S} \sum_{\mathbf{v} \in \mathbf{y}(s)} \mathbb{1}_{(h(s)=m \ \& \ \mathbf{y}=\mathbf{v})}, \tag{2}$$

where x^m is the m^{th} dimension of the data vector $\mathbf{x}$, $\mathbb{1}_{(.)} \in \{0, 1\}$ is the indicator function, $h(s) \in \{1, \ldots, M\}$ is the hash function mapping the pair feature s to the corresponding hash table entry, and $\mathbf{y}(s)$ is the set of poses that the pair s votes. Note that a pair feature can cast votes for multiple poses, or even multiple votes for a single pose as explained in Section 2.

The weighted voting function can then be written using the linear map $\mathbf{w}^T\mathbf{x}$, where $\mathbf{w}$ is an M-dimensional weight vector. When $\mathbf{w} = \mathbf{1}_M$, this function is equivalent to the original (uniform) voting function.

3.2 Max-Margin Learning of Weights

Let I be a 3D scene and $\mathbf{y}^*$ be the true pose of the target object in the scene[2]. The learning to rank 3D features problem is the problem of finding a non-negative weight vector which satisfies the following constraints:

$$\mathbf{w}^T\mathbf{x}^* > \mathbf{w}^T\mathbf{x}, \ \forall \mathbf{y} \neq \mathbf{y}^*, \ \forall I. \tag{3}$$

This means that for all the 3D scenes I containing the target object, the true pose of the object $\mathbf{y}^*$ should have more *weighted* votes than any other poses.

Except for a few simple shapes, a closed form solution to this problem does not exist. Therefore, we proceed by optimizing the weights using a dataset of 3D scenes containing the target objects. Typically, collecting and labeling a dataset for each object is a labor intensive procedure. Since in our problem we have access to 3D models of the objects, we avoid this tedious task and generate dataset through simulation by rendering models according to the sampled poses. Let $\{(I_i, \mathbf{y}_i^*)\}_{i=1,\ldots,N}$ be the synthesized dataset consisting of N 3D scenes and ground truth poses of the target object. Still, the constraints given in (3) might not define a feasible set. Instead, we reformulate learning the weights as a regularized soft constraint optimization problem:

$$(\mathbf{w}^*, \xi^*) = \arg\min_{\mathbf{w}, \xi} \sum_i \xi_i + \lambda R(\mathbf{w}) \tag{4}$$

$$s.t. \quad \mathbf{w}^T(\mathbf{x}_i^* - \mathbf{x}) \geq \Delta(\mathbf{y}_i^*, \mathbf{y}) - \xi_i, \forall i, \ \forall \mathbf{y} \neq \mathbf{y}_i^* \tag{5}$$

$$w_u \geq \mathbf{w} \geq 0, \quad \xi_i \geq 0, \tag{6}$$

which is similar to the margin re-scaling formulation of structured SVM learning [30,31] where $R(\mathbf{w})$ is a convex regularizer, λ controls the tradeoff between

[1] For ease of notation, we drop the subscript representing the dependency of the data vector $\mathbf{x}$ to the pose $\mathbf{y}$, and write $\mathbf{x}$ instead of $\mathbf{x}_\mathbf{y}$. For example $\mathbf{x}^*$ represents $\mathbf{x}_{\mathbf{y}^*}$.

[2] We assume that there is a single instance of the target object in the scene.

the margin and the training error, the loss function $\Delta(\mathbf{y}_i^*, \mathbf{y})$ penalizes larger pose deviation from the true pose more, and ξ_i are the slack variables corresponding to the constraint violations. In addition, we have an explicit upper bound w_u on the maximum weight (vote) a pair feature can have.

We use the loss function

$$\Delta(\mathbf{y}_i, \mathbf{y}) = 1 + \lambda_\theta \theta(\mathbf{y}_i, \mathbf{y}), \tag{7}$$

where $\theta(\mathbf{y}_i, \mathbf{y}) = \| \log(\mathbf{R}_{\mathbf{y}_i}^{-1} \mathbf{R}_{\mathbf{y}}) \|_F$ is the geodesic distance between the two rotation matrices encoded in 6-DOF poses $\mathbf{y}_i$ and $\mathbf{y}$. The constant 1 puts a fixed margin between two non-identical poses (Equation (5)) and λ_θ is a weighting factor between fixed margin and its scaled counterpart (the orientation difference)[3]. We ignore the translational component of the pose $\mathbf{y}$ since the observed pair features are largely insensitive to translation.

The form of the regularization function plays an important role on the accuracy and efficiency of the voting algorithm. Unless otherwise stated, in our experiments we used the quadratic regularization function $R(\mathbf{w}) = \mathbf{w}^T \mathbf{w}$. Using $L1$-norm regularizer $\|\mathbf{w}\|_1$ sparsifies the weight vector, leading to feature selection.

3.3 Optimization

Unlike standard structured learning problems that contain a combinatorial number of constraints due to a discrete label space, our labels, $\mathbf{y}$, are defined on a continuous pose space; therefore to optimally solve the problem an infinite number of constraints should be included to (5). However, we can still employ a cutting plane style optimization method [16] which has largely been used for discrete label space problems.

At each iteration k of the cutting plane algorithm, we use the previous set of weights $\mathbf{w}^{(k-1)}$ and solve the pose estimation problem for each scene I_i using the algorithm given in Section 2. In addition to the best pose, this algorithm provides a set of likely poses which are sorted according to the number of votes that they receive. For all the likely poses we evaluate the margin constraint (5):

$$(\mathbf{w}^{(k-1)})^T (\mathbf{x}_i^* - \mathbf{x}) \geq \Delta(\mathbf{y}_i^*, \mathbf{y}) \tag{8}$$

and add the most violated constraint to the selected constraint list. Let $\mathbf{y}^{(1:k)}$ and $\mathbf{x}^{(1:k)}$ be the set of all selected poses and constraints up to iteration k. Then the optimization problem at iteration k is given by

$$(\mathbf{w}^{(k)}, \xi^{(k)}) = \underset{\mathbf{w}, \xi}{\arg\min} \sum_i \sum_j \xi_{i,j} + \lambda R(\mathbf{w}) \tag{9}$$

$$s.t. \;\; \mathbf{w}^T (\mathbf{x}_i^* - \mathbf{x}_{i,j}^{(1:k)}) \geq \Delta(\mathbf{y}_i^*, \mathbf{y}_{i,j}^{(1:k)}) - \xi_{i,j}, \; \forall i, \; j \tag{10}$$

$$w_u \geq \mathbf{w} \geq 0, \quad \xi_{i,j} \geq 0 \tag{11}$$

[3] Two poses are considered identical if they have less than 12 degrees rotational distance.

Algorithm 1. Learning to Rank 3D Features

```
Input: Sampled 3D scenes and corresponding true poses: $\{(I_i, \mathbf{y}_i^*)\}_{i=1...N}$
Initialize $\mathbf{w}^0 = \mathbf{1}^M$, empty constraint set $\mathbf{x}_{i,j}^{(0)} = \{\}$, $k = 1$
while $k \leq max\ iteration$ do
    For each scene $I_i$ solve pose estimation (Section 2) using weights $\mathbf{w}^{(k-1)}$
    For all the likely poses compute the most violated constraints using (8). Let $\mathbf{x}_{i,j}^{(k)}$ and $\mathbf{y}_{i,j}^{(k)}$ be the set of the most violated constraints and corresponding poses. If this list is empty break
    Add most violated constraints to the constraint list $\mathbf{x}_{i,j}^{(1:k)} = \mathbf{x}_{i,j}^{(1:k-1)} \cup \mathbf{x}_{i,j}^{(k)}$
    Solve optimization problem (9) using constraint set $\mathbf{x}_{i,j}^{(1:k)}$ to get new weights $\mathbf{w}^{(k)}$. $k = k + 1$
end while
Output: Learned weight vector $\mathbf{w}$
```

where $\mathbf{y}_{i,j}$ represents the j^{th} violated pose in the i^{th} scene. Optimization problem (9) is convex and has finite number of constraints which we solve optimally. Note that the dimensionality of the data vectors can be quite high $M >> 10^5$, and we can have as many as $N \simeq 10^5$ margin constraints, which require careful attention to memory and computation. Fortunately, data vectors are quite sparse and using sparse matrices and sparse linear algebra, optimization can be solved efficiently. We use an interior point solver [32] to solve (9).

In general, the algorithm requires fewer iterations when multiple violated constraints (8) are added to the supporting constraint set at a given time, leading to faster operation. We add the most violated 10% of all the violated constraints at a time and usually obtain the optimal solution in $k = 3 - 4$ iterations of the cutting plane method. We initialize the optimization with uniform weights, $\mathbf{w}^{(0)} = \mathbf{1}_M$, which provides a good initial constraint set and significantly speeds up the convergence of the algorithm. The parameters of the learning algorithm (λ, w_u) are tuned using a grid search on a synthesized validation set as explained in Section 4. The optimization algorithm is summarized in Algorithm 1.

4 Experiments

We conducted extensive experimental evaluation of the proposed algorithm using two challenging datasets: (1) Mian *et al.*'s dataset [22] which contains curvy objects, and (2) a large scale real dataset collected by us containing mostly piecewise planar industrial-style objects, and compared our results with the state-of-the-art.

4.1 Curvy Objects

In the first experiment, we tested our algorithm using the real dataset introduced by Mian *et al.* [22] which is the standard benchmark for evaluating 3D recognition and pose estimation algorithms. This dataset contains 50 scenes taken

Fig. 3. Examples of generated synthetic scenes used for training. Four target objects were rendered with random poses.

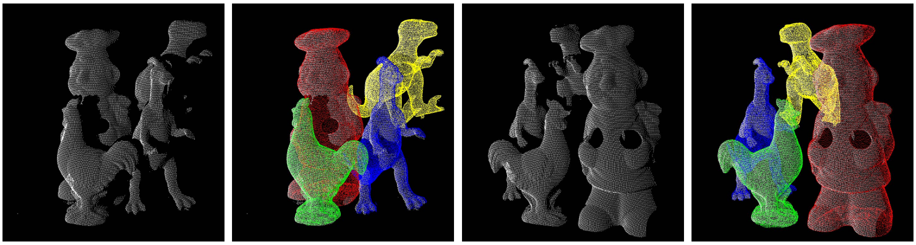

Fig. 4. Samples from Mian *et al.*'s dataset [22] and corresponding estimation results using learned weights (overlayed on the data)

using a Minolta range scanner where each scene contains four or five objects. These objects have smooth shapes with rich variations in surface normals. Since surface normals are more informative for these shapes, we used the S2S pair features in our algorithm. We used weighting hash table bins scheme to learn the weights. We generated two sets of synthetic datasets, one for training (used to learn the weights) and one for validation (used for tuning training parameters), each containing 500 3D scenes, by rendering 3D models of the objects according to randomly sampled rotation and translation parameters. In the simulation, maximum occlusion ratio was set to be 0.8. Example training images are shown in Figure 3.

We used the standard evaluation procedure defined in [7] which (1) excluded the rhino object and (2) labeled an estimation as correct if the retrieved translation and rotation are within the one-tenth of the object's diameter and 12° of the ground truth pose respectively. In Figure 5, we plot recognition rate vs. the occlusion ratio for our algorithm and four other leading algorithms: (1) Spin images [15], (2) Tensor matching [22], (3) Drost *et al.* [7] and (4) Support vector shape (SVS) [24]. In addition, we also included our implementation of the original Drost *et al.*'s algorithm which is equivalent to using uniform weights (labeled as "our implementation" in the figure).

As seen in Figure 5, the proposed algorithm produced equal or better results at all the occlusion rates than all the prior work. For all the objects with less

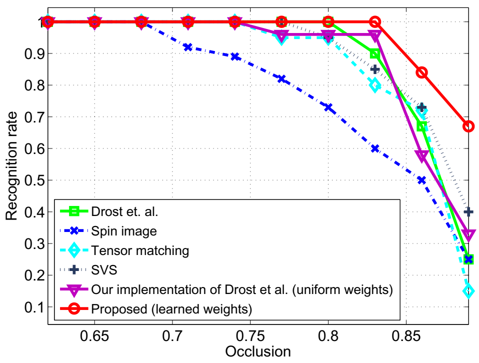

Fig. 5. Recognition rate vs. the occlusion rate on Mian *et al.*'s dataset. Our method improves the state-of-the-art significantly especially at larger occlusion rates.

than 0.84 occlusion, the reported recognition rates were 97% for Drost *et al.* and 96.7% for tensor matching, whereas our algorithm achieved 98.8% recognition rate. The improvement is even more prominent at larger occlusion rates. Our implementation using uniform weights is consistent with Drost *et al.* and achieved 95.8% recognition rate (only marginally lower) at less than 0.84 occlusion. By learning the weights, our algorithm significantly reduced the error by 71.4% for up to 0.84 occlusion and 55% for all the occlusion rates. This improvement was consistently observed for all of the four objects. Several examples of pose estimation using the proposed approach are shown in Figure 4. Note that Rodala *et al.* [26] have also reported competitive results on Mian *et al.*'s dataset. However, in that study the recognition rate was evaluated using the feature matching accuracy rather than the pose estimation error, therefore it is not possible to directly compare with our method.

4.2 Industrial Objects

In the second set of experiments, we conducted evaluation of the algorithm on both synthetic and real data using 5 industrial-style objects, namely circuit breaker (CB), clamp (CL), knob (KN), T-nut (TN), and L-shape (LS). These objects capture a wide spectrum of 3D object shape properties starting from simple planar shapes to more featured and semi-smooth surfaces to near and complete shape symmetries, generating various different challenges for recognition and pose estimation algorithms. Figure 6 shows renderings of 3D models of these objects on captured real scenes.

The real test scenes were captured using a structured light based 3D sensor under a challenging setup where all of these five shapes were placed in a bin in random poses to generate cluttered scenes with varying degrees of occlusion. We captured 450 such scenes and manually labeled ground truth poses of all 5 objects in each scene. A few example scenes and ground truth labels are shown in the top row of Figure 6, where labeled poses are overlayed on data. Since our

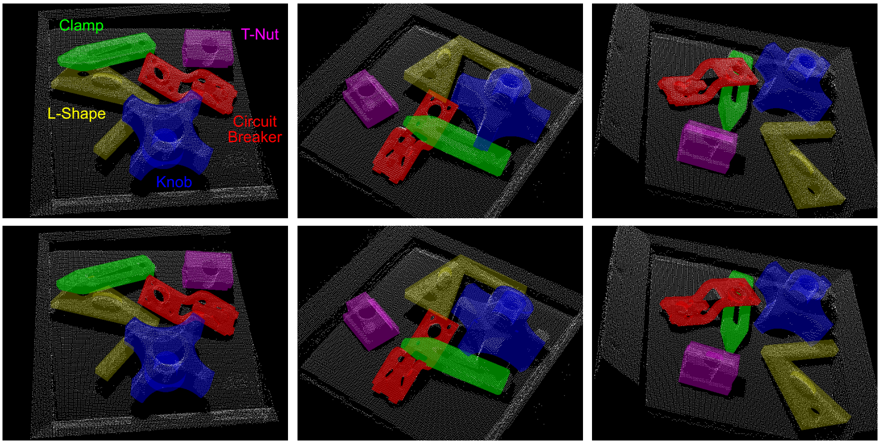

Fig. 6. Pose estimation examples for real industrial objects. (Top) Ground truth poses for each object. (Bottom) Estimated poses using weighted voting for each object.

results were saturated (only 1% failure case remained) using the Mian *et al.*'s dataset, this large scale and challenging test platform enabled us to better study the details of the proposed algorithm.

We replicated the real environment in simulation by rendering the 3D models of the objects under the same setup and generated training, validation and testing datasets, 500 each. In the simulation, maximum occlusion ratio was set to be 0.2 for training and validation and 0.6 for test scenes. Note that, similar to the previous experiment, training did not use any real data. In our evaluation, an estimated pose of the object was considered to be correct if the difference of the estimated pose and the ground truth pose was less than 5 millimeters in translation and 15 degrees in rotation, while the diameter of the largest object was 60 millimeters.

Many of these shapes contain large planar surfaces therefore edges encode valuable information. Unless otherwise stated, in all the experiments using this dataset we used the B2B pair feature, which was shown to provide a good compromise between speed and accuracy [6], and we used weighting hash table bins scheme to learn the weights.

Learning vs. Uniform Voting: We compared learning results with the baseline algorithm—uniform (equal) weights for each pair feature [7,6]. The results were computed for both synthetic and real data and are shown in Table 1 and Table 2, respectively. Synthetic and real experiments showed the same trend, and we achieved major performance boost (20% to 50% error reduction) with the learning framework except for the LS object. This shape is quite unique (distinct from other shapes and does not contain self symmetries), which was easy to identify using the B2B features; it was therefore reliably identified except for very large occlusions, leading to more modest improvement by learning. In

Table 1. Synthetic data evaluation (recognition rate) using the industrial objects

Objects	CB	CL	KN	TN	LS
Uniform	0.698	0.505	0.616	0.460	0.874
Learned	**0.834**	**0.708**	**0.852**	**0.528**	**0.876**

Table 2. Real data evaluation using the industrial objects

Objects	CB	CL	KN	TN	LS
Uniform	0.776	0.413	0.362	0.493	0.909
Learned	**0.871**	**0.720**	**0.522**	**0.684**	**0.922**

addition, synthetic and real data results were quite similar, showing that our synthesized scenes were good representations of the real scene leading to good generalization accuracy. Note that the performance difference for the KN object between synthetic and real data was due to small inaccuracy of the 3D model. In Figure 6, we superimpose the estimated poses on top of the captured data and observe that the estimations are very accurate.

Alternative Pair Features: We evaluated the learning algorithm for a different point pair feature, S2S, and also learned a fusion feature that is a combination of S2S and B2B features. In this experiment, we used the CB object and performed evaluation using real data. The S2S pair feature uses two points on the surface of the object. Since the CB object is mainly composed of two large planar surfaces, this feature is quite weak for the object. In addition, the scenes contained several large planes corresponding to the faces of the container bin, as seen in Figure 6. Due to these challenges, the S2S pair feature could only recover 19.6% of the poses in the scenes using uniform weights, and most of the time it confused the bin surfaces with the object (Table 3).

By learning the weights, the performance improved almost by an order to 36.4%. However, the performance was still far from the B2B feature due to the reasons explained above. Joint S2S and B2B feature retrieved 62.2% of the objects in the uniform case and this result improved to 83.7% using learning. Even though this result is not as high as just using B2B feature, we believe that an important factor contributing to the reduced performance was high dimensional feature space leading to overfitting and could be improved by generating more training data.

Weighting Strategies: We compared different weighting strategies that were discussed in Section 3: (1) weighting hash table bins and (2) weighting model points. The comparison is shown in Table 4. Both of the approaches led to significant improvement compared to the uniform weights. Weighting model points had only 1% less performance than weighting hash table bins strategy. For this experiment, using the CB object, learning the weights of hash table bins required optimization on a 39K dimensional space compared to 5.4K dimensions for weighting model points, one for each point.

Table 3. Comparison of different pair features for the CB object

Features	B2B	S2S	S2S+B2B
Uniform	0.776	0.196	0.622
Learned	**0.871**	**0.364**	**0.837**

Table 4. Comparison of weighting strategies for the CB object

Uniform	Weighting hash bins	Weighting model points
0.776	**0.871**	0.867

Figure 7 illustrates the output of the learning algorithm for the weighting model points strategy, where points with larger weights are drawn by bigger dots and more red in color. The learning algorithm identified non-robust boundary points, such as the ones on the small circles, which were not repeatable across different views, and deemphasized these points. This object has 180° near-symmetry, frequently causing confusion in pose estimation. This near-symmetry can only be resolved by the interior boundaries, which were highly weighted by the learning algorithm.

Effect of Regularization: We compared the different regularization schemes for the optimization problem. The results are given in Table 5. Although the performance difference is negligible, regularization based on $L1$-norm prefers a sparse set of weights, leading to a more strict feature selection. Compared to quadratic regularization which selects 24.5% of the hash table bins, $L1$-norm regularizer selects only 3.3%. This leads to an efficient implementation for the voting framework. However, rather than voting, the pose clustering process took the most computation in our implementation, hence no major speed improvement was observed. We note that, in other experiments, we observed that very sparse solutions can perform worse in challenging scenes such as large amount of noise and missing data.

Alternative Learning Strategies: We also tested other simple strategies for determining weights based on occupancy statistics: (1) Weights inversely proportional to the number of pairs that occupy the bin; (2) Weights inversely proportional to the square root of number of pairs within a bin. None of these simple strategies produced better results than uniform weighting. Even though some features may be repeated in the model many times or they frequently vote for incorrect poses, these features might still be necessary to resolve confusion among certain poses. Our max-margin framework optimizes for the voting function (difference of votes between correct and incorrect poses) jointly for all the features; thus it is optimal for the pose estimation task.

Cluttered Bin: We used the proposed algorithm to estimate the 3D poses of these objects in heavy clutter. Figure 8 shows three such scenes with our estimation results overlayed. As shown, the proposed algorithm was able to retrieve

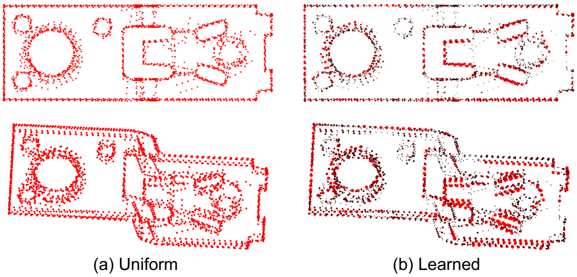

Fig. 7. Learning weights on model points. (a) Boundary points visualized with uniform weights used for B2B. (b) Boundary points visualized with weights (the larger the weights, the bigger and more red in color) learned by our algorithm.

Table 5. Comparison of regularization functions

Regularizer	Accuracy	# of selected hash bins	% of selected hash bins
Quadratic	0.871	9571	24.5%
$L1$-norm	**0.873**	**1275**	**3.3%**

several good pose candidates simultaneously. Please refer to the supplementary video for a robotic system using our algorithm in object grasping task.

Computational Requirement: Learning weights (offline) takes around 10 minutes per object model. Pose estimation time was less than 0.5s for industrial parts using 20% sampling of the B2B features (≈ 1000 points). For the smooth object experiment, we sampled 5% of the dense S2S features (≈ 4000 points). The pose estimation time in this experiment was 15s (vs. 85s in Drost *et al.* [7]).

5 Related Work and Discussion

Other than the local feature point based methods described in Section 1, there exists a large body of work that models 3D shapes globally such as using 2D/3D contours [3,4], shape templates [10,19], and feature histograms [33,12]. In general, these global methods require the target object in isolation and are more sensitive to occlusion. Also, the appearance change due to pose variations necessitates the use of a large number of shape templates, which has the drawback of increased memory footprint and matching time.

There exists many examples of learning interest point detectors and feature descriptors in computer vision, and here we discuss only a few closely related. Holzer *et al.* [11] presented a learning-based keypoint detector for range data. Unlike our approach, their learning model approximates an existing detector with a learned more efficient classifier, whereas our model identifies and weights features for a specific task and an object. Shotton *et al.* [28] combined simple

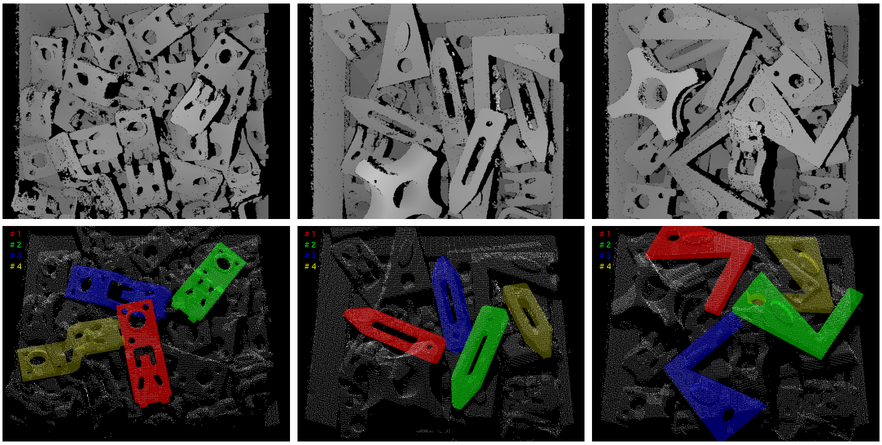

Fig. 8. Pose estimation for several different objects in cluttered bins

decision functions based on depth differences between pairs of points into a decision tree classifier for identification of human body parts. Moji and Malik [21] presented a 2D part based object detector where the contribution of each local part to a given object location is learned by optimizing a classification error cost. Similarly, Knopp *et al.* [17] proposed to weight the 3D features based on how often they vote for the correct shape center on a training set. In contrast, in our approach the 3D pose to vote is dynamically determined through matching features and the important features to match and their weights are learned by directly optimizing the voting function. In addition, our approach does not require a separate manually annotated training set and can learn the important features using only the 3D model of the object and simulation.

As explained in Section 3, to simplify the learning, we group sets of features and enforce them to have the same weight. The weighting scheme based on hash table bins has connection to well-known bag-of-words model, where hash keys can be considered the codewords mapping features to a histogram. However, even though the form of the function is the same, the setup of the learning problem is very different. In our problem, a hash table bin acts as a mid-level representation and votes for multiple different poses which are determined only during classification (based on matching between model and scene pairs features), rather than scoring a fixed set of classes which are predetermined as in bag-of-words model.

Similarly, learning the weights of a voting function has relations to learning ranking functions, which is extensively studied in machine learning community for information and image retrieval [14,5,13]. More notably, [25] learns mid-level object representations based on relative attributes. However, unlike these approaches, the relation of features to output is more indirect in our voting framework where a scene feature simultaneously votes for multiple poses and multiple different scene features vote for a given pose.

6 Conclusion

We presented a new framework to learn a weighted voting function for 3D object recognition and pose estimation tasks. Our approach identifies and ranks important features on a given 3D object by optimizing a discriminative cost function. We evaluated the algorithm on large scale synthetic and real datasets, and achieved major improvements compared to the state-of-the-art.

References

1. Bariya, P., Nishino, K.: Scale-hierarchical 3D object recognition in cluttered scenes. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), pp. 1657–1664 (2010)
2. Belongie, S., Malik, J., Puzicha, J.: Shape matching and object recognition using shape contexts. IEEE Trans. Pattern Anal. Mach. Intell. 24(4), 509–522 (2002)
3. Bookstein, F.L.: Principal warps: Thin-plate splines and the decomposition of deformations. IEEE Trans. Pattern Anal. Mach. Intell. 11(6), 567–585 (1989)
4. Borgefors, G.: Hierarchical chamfer matching: A parametric edge matching algorithm. IEEE Trans. Pattern Anal. Mach. Intell. 10(6), 849–865 (1988)
5. Cao, Z., Qin, T., Liu, T.Y., Tsai, M.F., Li, H.: Learning to rank: From pairwise approach to listwise approach. In: Proc. Int'l Conf. Mach. Learning (ICML), pp. 129–136 (2007)
6. Choi, C., Taguchi, Y., Tuzel, O., Liu, M.Y., Ramalingam, S.: Voting-based pose estimation for robotic assembly using a 3D sensor. In: Proc. IEEE Int'l Conf. Robotics Automation (ICRA), pp. 1724–1731 (May 2012)
7. Drost, B., Ulrich, M., Navab, N., Ilic, S.: Model globally, match locally: Efficient and robust 3D object recognition. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), pp. 998–1005 (June 2010)
8. Frome, A., Huber, D., Kolluri, R., Bülow, T., Malik, J.: Recognizing objects in range data using regional point descriptors. In: Pajdla, T., Matas, J. (eds.) ECCV 2004. LNCS, vol. 3023, pp. 224–237. Springer, Heidelberg (2004)
9. Harris, C., Stephens, M.: A combined corner and edge detector. In: Alvey Vision Conference, Manchester, UK, vol. 15, p. 50 (1988)
10. Hinterstoisser, S., Holzer, S., Cagniart, C., Ilic, S., Konolige, K., Navab, N., Lepetit, V.: Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In: Proc. IEEE Int'l Conf. Computer Vision (ICCV), pp. 858–865 (November 2011)
11. Holzer, S., Shotton, J., Kohli, P.: Learning to efficiently detect repeatable interest points in depth data. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part I. LNCS, vol. 7572, pp. 200–213. Springer, Heidelberg (2012)
12. Horn, B.K.P.: Extended gaussian images. Proceedings of the IEEE 72(12), 1671–1686 (1984)
13. Jain, V., Varma, M.: Learning to re-rank: query-dependent image re-ranking using click data. In: Proc. Int'l Conf. World Wide Web, pp. 277–286 (2011)
14. Joachims, T.: Optimizing search engines using clickthrough data. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge discovery and Data Mining, pp. 133–142 (2002)

15. Johnson, A.E., Hebert, M.: Using spin images for efficient object recognition in cluttered 3D scenes. IEEE Trans. Pattern Anal. Mach. Intell. 21(5), 433–449 (1999)
16. Kelley Jr., J.E.: The cutting-plane method for solving convex programs. Journal of the Society for Industrial & Applied Mathematics 8(4), 703–712 (1960)
17. Knopp, J., Prasad, M., Willems, G., Timofte, R., Van Gool, L.: Hough transform and 3D SURF for robust three dimensional classification. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part VI. LNCS, vol. 6316, pp. 589–602. Springer, Heidelberg (2010)
18. Lamdan, Y., Wolfson, H.J.: Geometric hashing: A general and efficient model-based recognition scheme. In: Proc. IEEE Int'l Conf. Computer Vision (ICCV), vol. 88, pp. 238–249 (1988)
19. Liu, M.Y., Tuzel, O., Veeraraghavan, A., Chellappa, R.: Fast directional chamfer matching. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), pp. 1696–1703 (June 2010)
20. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int'l J. Computer Vision 60(2), 91–110 (2004)
21. Maji, S., Malik, J.: Object detection using a max-margin hough transform. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), pp. 1038–1045 (2009)
22. Mian, A.S., Bennamoun, M., Owens, R.: Three-dimensional model-based object recognition and segmentation in cluttered scenes. IEEE Trans. Pattern Anal. Mach. Intell. 28, 1584–1601 (2006)
23. Mikolajczyk, K., Schmid, C.: Indexing based on scale invariant interest points. In: Proc. IEEE Int'l Conf. Computer Vision (ICCV), vol. 1, pp. 525–531 (2001)
24. Nguyen, H., Porikli, F.: Support vector shape: A classifier based shape representation. IEEE Trans. Pattern Anal. Mach. Intell. 35(4), 970–982 (2013)
25. Parikh, D., Grauman, K.: Relative attributes. In: Proc. IEEE Int'l Conf. Computer Vision (ICCV), pp. 503–510 (2011)
26. Rodolà, E., Albarelli, A., Bergamasco, F., Torsello, A.: A scale independent selection process for 3D object recognition in cluttered scenes. Int'l J. Computer Vision 102(1-3), 129–145 (2013)
27. Rusu, R.B., Blodow, N., Beetz, M.: Fast point feature histograms (FPFH) for 3D registration. In: Proc. IEEE Int'l Conf. Robotics Automation (ICRA), pp. 3212–3217 (May 2009)
28. Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A.: Real-time human pose recognition in parts from single depth images. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), pp. 1297–1304 (June 2011)
29. Stein, F., Medioni, G.: Structural indexing: Efficient 3-D object recognition. IEEE Trans. Pattern Anal. Mach. Intell. 14(2), 125–145 (1992)
30. Taskar, B., Guestrin, C., Koller, D.: Max-margin markov networks. In: Proc. Neural Information Processing Systems (NIPS), vol. 16 (2003)
31. Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y.: Support vector machine learning for interdependent and structured output spaces. In: Proc. Int'l Conf. Mach. Learning (ICML), p. 104 (2004)
32. Tutuncu, R., Toh, K., Todd, M.: Solving semidefinite-quadratic-linear programs using sdpt3. Mathematical Programming Ser. B 95, 198–217 (2003)
33. Wahl, E., Hillenbrand, U., Hirzinger, G.: Surflet-pair-relation histograms: A statistical 3D-shape representation for rapid classification. In: Proc. Int'l Conf. 3-D Digital Imaging and Modeling (3DIM), pp. 474–481 (October 2003)