

# KeyMatchNet: Zero-Shot Pose Estimation in 3D Point Clouds by Generalized Keypoint Matching

Frederik Hagelskjær and Rasmus Laurvig Haugaard

**Abstract**—In this paper, we present KeyMatchNet, a novel network for zero-shot pose estimation in 3D point clouds. Our method uses only depth information, making it more applicable for many industrial use cases, as color information is seldom available. The network is composed of two parallel components for computing object and scene features. The features are then combined to create matches used for pose estimation. The parallel structure allows for pre-processing of the individual parts, which decreases the run-time.

Using a zero-shot network allows for a very short set-up time, as it is not necessary to train models for new objects. However, as the network is not trained for the specific object, zero-shot pose estimation methods generally have lower accuracy compared with conventional methods. To address this, we reduce the complexity of the task by including the scenario information during training. This is typically not feasible as collecting real data for new tasks drastically increases the cost. However, for zero-shot pose estimation, training for new objects is not necessary and the expensive data collection can thus be performed only once.

Our method is trained on 1,500 objects and is only tested on unseen objects. We demonstrate that the trained network can not only accurately estimate poses for novel objects, but also demonstrate the ability of the network on objects outside of the trained class. Test results are also shown on real data. We believe that the presented method is valuable for many real-world scenarios. Project page available at [keymatchnet.github.io](https://keymatchnet.github.io)

## I. INTRODUCTION

Pose estimation enables greater flexibility in robotics as new objects can be manipulated without the need for mechanical fixtures or teaching specific robot positions. This enables shorter changeover times and faster adaptation to production demands. However, the set-up of computer vision algorithms can itself be a very time-consuming task [2]. There is, therefore, great interest in pose estimation solutions with simple set-ups. Deep learning has allowed learning the specifics of the object and the scenario, thus giving much better performance than human fine-tuning [3], [4]. However, collecting the data for training the deep neural networks can be very time-consuming, thus limiting the usability. To avoid this data collection task, synthetic data has gained widespread use [3]. But generating large amounts of data, and then training the network for each new object is computationally expensive and increases set-up time. To address these problems, we introduce KeyMatchNet, a neural network for zero-shot pose estimation.

A feature of KeyMatchNet is that it only uses depth information and exclude color. To the best of our knowledge

Both authors are with SDU Robotics, Mærsk Mc-Kinney Møller Institute, University of Southern Denmark, 5230 Odense M, Denmark  
`{frhag, rlha}@mmmi.sdu.dk`

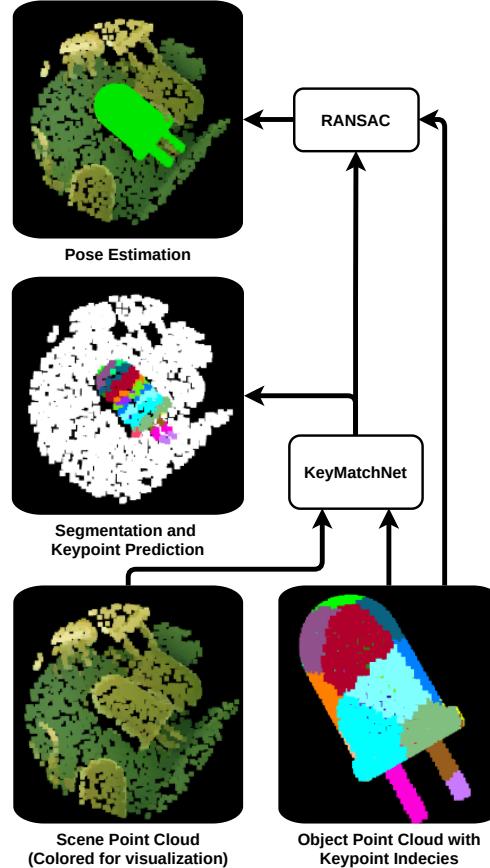


Fig. 1. Illustration of the pose estimation method on an unseen object. The input to the network is a scene point cloud and an object point cloud. The object keypoints are visualized by the different color segments. The network output is both instance segmentation and keypoint predictions, which are combined to provide predictions only for the object. Finally these predictions are used in RANSAC [1] for pose estimation. The striped keypoint prediction pattern is the result of the objects' rotational symmetry.

this is the first colorless zero-shot network for pose estimation. The color information is omitted as most industrial objects does not include detailed surface information. Thus by omitting color our method is also usable for real industrial use cases. This is in contrast to most other methods that, does obtain high scores on benchmark datasets, but are not usable for many real objects.

Another important aspect of our developed network is reusability. The reusability consists of two parts. Firstly, rather than training a single model for each object, a zero-shot pose estimation algorithm is built with a single network for all objects. This is accomplished by learning to match keypoints from the object to the scene, instead of learning

specific features from the object. Thus, the network input is both scene and object information. For a novel object, the network can be reused without re-training. Additionally, compared with similar methods [5] our network is split into parallel computation of object- and scene-features. This allows for pre-computing the object features, giving a significant speed-up at run-time. Additionally, if different objects are pose estimated the scene feature can be reused.

Zero-shot pose estimation methods are, generally, less accurate compared with methods trained for specific objects [6], as these methods can integrate object information into the model. However, these methods often use synthetic training data and, therefore, do not integrate real scene information. As zero-shot methods do not require training for new objects, it is much more feasible to use real training data and integrate scene information into the network.

For many robotic set-ups this much better fits the application, i.e. novel objects, but unchanging scene. New objects can be introduced faster as training is not required, and the power consumption for training will be removed. In this paper we focus on the task of bin picking with homogeneous bins, which is a difficult challenge that often occurs in industry [7]. The homogeneous bin also removes the need for object detection and allows us to only focus on pose estimation. We recognize that the current approach as shown in e.g., [4] has huge importance, but also state it is not the best solution for all tasks. In this paper we show that generalized pose estimation can obtain very good performance when restricting the scenario. We believe that these results invite further research into this topic, as creating flexible set-ups with lower energy consumption is an important topic. In this work the following contributions are presented.

- A zero-shot pose estimation method for colorless point clouds.
- A parallel structure allowing pre-processing of individual parts.
- A dataset of 1,500 objects in homogeneous bins.
- Demonstrating the feasibility of zero-shot pose estimation for industrial use cases.

## II. RELATED WORKS

The importance of visual pose estimation along with the complexity has resulted in a large amount of different solutions [3], [4], [8]. In this section an overview will be given of classic pose estimation methods, deep learning based methods and current zero-shot pose estimation methods.

### A. Classic Pose Estimation

In the classic pose estimation case two different approaches are generally used, template matching and feature matching. Template matching [9], [10] is performed using a cascade search of where templates of the object are matched at different positions in the image. To perform the search, images have to be generated of all poses that the object can appear in.

In feature matching pose estimation is based on matching features between the scene and object, and computing the

pose by e.g. Kabsch-RANSAC [1], [11] or pose voting [12]. The matches are computed using handcrafted features, which are generally computed in 3D point clouds. A huge amount of handcrafted features have been developed [13], with Fast Point Feature Histograms (FPFH) [14] being one of the best performing features. The benefit of the classic method is that the set-up can be performed using only the CAD model. However, these methods are weak towards clutter and occlusion, and thus are often not usable in real scenarios.

### B. Deep Learning Based

Generally deep learning based methods are based on color information [3], [4]. This is possibly a result of many deep learning based methods developed for this space, with huge pre-trained networks available. These methods have vastly outperformed the classical methods, however, a network is often trained per object [3], [4]. Deep learning for pose estimation has also been performed in point clouds with methods such as PointVoteNet [15]. We base our method on PointVoteNet, but only train a single network for all objects.

**2D methods:** SSD-6D [16] is similar to template matching methods, in that it employs the cascade search. However, unlike template matching the comparison is performed using a neural network. The network is trained to classify the presence and correct orientation of an object. Thus by cascade search in the image the correct object pose is found. The position and orientation can also be computed independently [17], [18]. In PoseCNN [17] the orientation is computed using a regression network.

An alternate approach is by detecting keypoints of the object [18], [19]. This is performed in BB-8 [20] where the bounding box is predicted and then used for pose estimation. PVNet [21] compute keypoint locations by first segmentation the object, and then computing the relative position of keypoint for all pixels belonging to the object.

A method more similar to our approach is EPOS [22] where both object segmentation and dense keypoint predictions are calculated. This approach is also performed in DPOD [23] which also employs pose refinement. Unlike our methods these methods use RGB information and train a single network per object.

**3D methods:** DenseFusion [24] is a method that combines both 2D and 3D information. Initial segmentations are found in 2D, and features computed in 2D are then integrated with 3D features. The 3D features are computed using PointNet [25]. Finally a PnP is used to perform pose estimation using keypoints from the network. PVN3D [26] is another method that combines 2D and 3D features. It also computes keypoints for which are used for pose estimation. Unlike our method the keypoints are the object bounding box, and not keypoints on the object.

A method more similar to our method is PointVoteNet [15]. Here PointNet [25] is used to compute keypoints for each scene point. Similar to our method the computation is also performed without color information. In the extended version [27] DGCNN [28] is used and the segmentation and feature computation is separated, both of which appears in

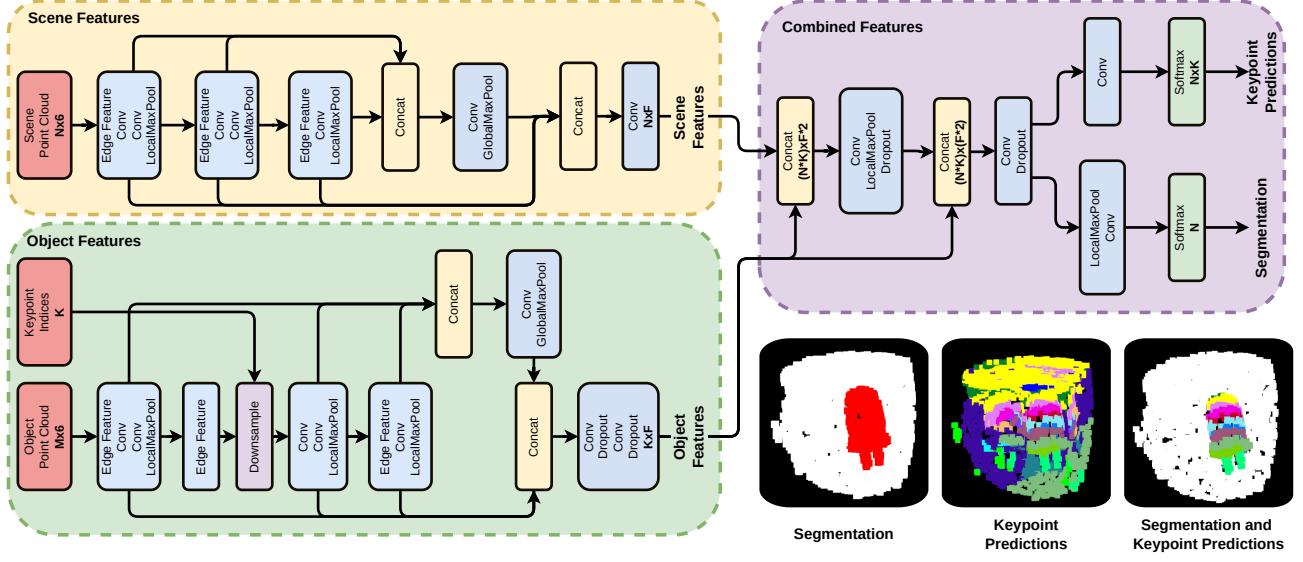


Fig. 2. The network structure of the developed method. Bold text indicates the matrix size, where  $N$  is the number points in the scene,  $M$  is the number of points in the object,  $K$  is the number of keypoints, and  $F$  is the feature size at each point. The input point clouds consist of the combined xyz and normal vector information and are thus size **6**. Object and scene features are computed independently, which allow for precomputed object features. The Segmentation and Keypoint Prediction outputs are shown along with the combination into matches.

our method. ParaPose [29] use this method combined with automatic parameter to obtain state-of-the-art performance on the Occlusion dataset [30]. However, dissimilar to our method, for PointVoteNet a network is trained for each object.

### C. Generalized Pose Estimation

Several approaches have been developed for generalized pose estimation. As in the single object based pose estimation, the field of generalized pose estimation is also dominated by color-based methods [6], [31], [32]. The general approach for these methods is to match templates of the object with the real image, similar to SSD-6D. These templates can either be generated synthetically as in [33] or with a few real images as in FS6D [32]. The same approach have also been used for tracking of unknown objects [34]. As opposed to SSD-6D the network does not learn to recognize specific poses of objects, but instead to compare how well an image match the real scene. By generating synthetic views of novel objects, these methods are thus able to perform pose estimation without training new networks. MegaPose6D [6] is a notable example where the network is trained on a huge dataset with 2 million images.

The method most similar to ours is a point cloud based method [5]. It also uses the object point cloud as input along with the scene point cloud. The method also employs a segmentation step as in our method, however the pose estimation part of the method whereas our approach matches specific keypoint. Several other differences are also present compared with our approach; the method includes color information which is often much more difficult to obtain. It does not limit the scenario and thus does not obtain the increased performance from this. And it does not separate the object and scene features, thus the object features must be computed for each pose estimation.

### III. METHOD

The developed method is a network that matches scene-points to object keypoints. These matches are then fed into a pose estimation solver, in our case Kabsch-RANSAC [1], [11], and a final pose is found. The scene and object data are point clouds without color information. Color information is not used as it is seldom available for CAD (Computer Aided Design) models and would limit the usability of the method. The network structure is made with two parallel components computing both scene- and object-features independently. The network structure is shown in Fig. 2. The scene features are computed for all points using the standard DGCNN [35] structure. However, for the object only the keypoint features are computed. Thus, after the second neighbor computation in the DGCNN, the object point cloud is down-sampled to only contain the keypoints. To ensure that the all keypoints contain knowledge about all other keypoint features, the number of neighbors is set to match the number of keypoints.

After both object- and scene-features are computed, the features are combined. Pairs of object-scene-feature vectors are created concatenating each scene-feature with each object-feature. The resulting number of pairs is the number of scene-points and key-points multiplied.

An MLP (Multi Layer Perceptron) then processes each scene-keypoint pair independently and a local maxpool combines all keypoint information at the scene point.

This new scene feature is again combined with the keypoint features, and processed by an MLP. Two different MLPs are then used to compute the segmentation and the keypoint predictions. To compute the segmentation a local maxpool is applied, followed by an MLP which gives segmentation for each point. To compute the keypoint matches an MLP is applied to the combined features giving a score for each scene-keypoint pair. A softmax is then applied across

the keypoint domain for each scene point.

Finally, Kabsch-RANSAC is used with the segmentation and keypoint predictions to compute the object pose. We employ the vote threshold as in [15] to allow multiple keypoint matches at a single scene point. Compared with the vote threshold of 0.95 of [27], our threshold is set at 0.7 as the zero-shot method is expected to be less precise.

#### A. GPU based RANSAC:

To compare our developed method with classic pose estimation methods the RANSAC implemented in Open3D [36] is used. ICP [37] is then used for refinement. However, as the RANSAC method is running on the CPU a significant increase in run-time occurs. To decrease the run-time a RANSAC has also been implemented on the GPU using PyTorch [38].

The GPU based RANSAC is implemented as a Coarse-to-Fine RANSAC, with five iterations. Initially, 1000 poses are computed using the Kabsch [11] algorithm on triplets sampled from the matches. The poses are then sorted according to the amount of inlying matches. An additional requirement is that the normal vector between matches cannot be larger than 30 degrees. This ensures that only the parts of the object pointing towards the camera are matched.

The best scoring pose is then selected, and further refinement is performed. The pose is refined by computing Kabsch using only matches within the inlier distance. By gradually decreasing the inlier distance the pose is iteratively refined. As this method includes refinement, ICP [37] is not used for the GPU based RANSAC.

#### B. Generating object data:

The object point clouds is generated using Poisson sampling to obtain 2048 evenly sampled points on the surface. Farthest point sampling is then used to obtain the keypoints spread evenly on the object.

#### C. Computing object features offline:

As shown in Fig. 2 the features computed from the object point cloud are independent of the features computed from the scene point cloud. This is opposed to [5]. This allows us to pre-compute the object features, reducing the run-time and the computational cost. During training the keypoints are continuously re-sampled, to avoid the network over-fitting to a specific combination.

#### D. Generating scene data:

The scene point clouds are generated using BlenderBin<sup>1</sup>. BlenderBin is an open-source tool built using the Blender-Proc [39] simulator. It allows to easily generate synthetic data with objects placed in bins. For each object images are created with the number of objects in the bin ranging from one to twenty. The bin model is kept the same in all scenes. Examples of the training data is shown in Fig. 3.

As the scenario is homogeneous bin-picking the detection is vastly simplified. By segmenting the known bin, all



Fig. 3. Examples of the training data. Colors are only for visualization.

remaining points belong to objects. As the contents are homogeneous any random point is known to belong to a correct object. By then extracting a point cloud around the sampled point using the object diagonal, all points in the scene belonging to the object will be obtained. The point cloud is then centered around the sampled point to allow for instance segmentation.

## IV. EXPERIMENTS

For all experiments a single model is trained. The training data consists of 1,500 CAD models from an online database of electronic components<sup>2</sup>. Fifty objects are excluded for validation.

The test dataset consists of seven electrical components introduced in a different dataset [40]. The seven test objects are shown at the top of Fig. 4.

Additionally we test the ability of the network on out of class objects. Seven industrial objects from the WRS [7] dataset is used for testing. On this dataset we show the networks ability to generalize to other objects outside of the training scope. The objects from the WRS dataset are shown in the bottom of Fig. 4.

Finally, results are shown for real test data. The method is compared with a state-of-the-art colorless pose estimation algorithm [29]. Results are not shown for zero-shot methods requiring RGB information as this is not available for the objects.

All point cloud processing was performed using the Open3D framework [36]. The network processing was performed using PyTorch [38].

#### A. Network Training

The network was trained on a PC environment with two NVIDIA GeForce RTX 2080 GPUs. The network was trained for 120 epochs lasting approximately six days (141 hours). For each object we generate 160 point clouds giving an epoch size of 232,000.

The network is trained with a batch size of 14, with 7 on each GPU, using the Adam optimizer [41], with an initial learning rate of 0.0001. We use a step scheduler with a step size of 20 and the gamma parameter set to 0.7. The loss is calculated using cross entropy with segmentation and keypoint loss weighted 0.2 and 0.8, respectively. For the keypoint loss only points belonging to the object are used. Group norm [42] with size 32 is used as opposed to Batch Norm as a result of the small batch size.

<sup>1</sup><https://github.com/hansaskov/BlenderBin/>

<sup>2</sup>[https://www.pcb-3d.com/membership\\_type/free/](https://www.pcb-3d.com/membership_type/free/)

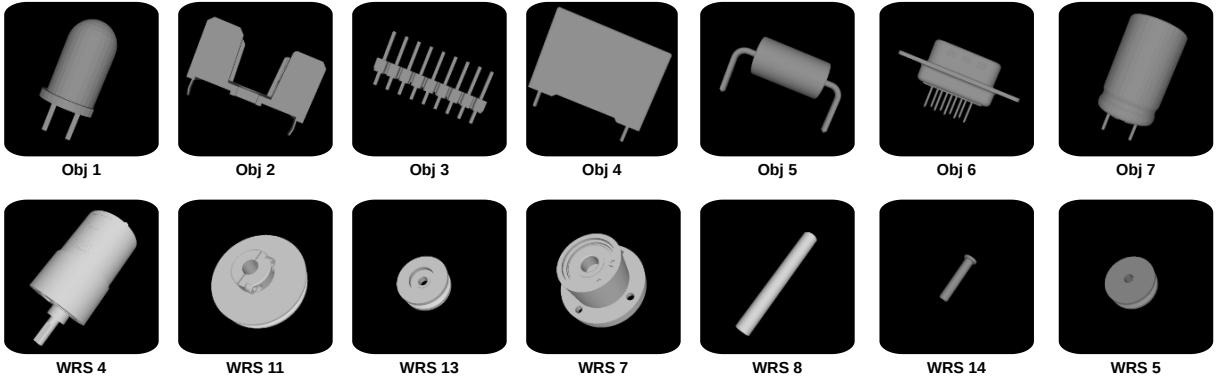


Fig. 4. Top: The seven electronic components in the test dataset. Bottom: The seven components from the WRS dataset used for out of class tests.

TABLE I

THE LOSS AND ACCURACY OF THE NETWORK. THE PARENTHESES INDICATE USING 10% OF THE TRAINING DATA.

Split	Training	Validation	Test
Loss	1.59 (1.63)	1.33 (1.40)	1.56 (1.65)
Seg. Acc	0.98 (0.98)	0.96 (0.94)	0.95 (0.94)
Key. Acc	0.27 (0.28)	0.31 (0.29)	0.27 (0.24)

The dropout is set to 40 %, and applied as shown in Fig. 2. The dropout on the object features is used as the network should not overfit to specific parts of the object. Additionally, up to 0.7 % Gaussian noise is applied to the object and scene point clouds, and 10 % position shift is applied to the object point cloud.

### B. Training and test performance

The performance for the trained networks is show in Tab. I. Performance is shown for both the loss, segmentation accuracy, and keypoint accuracy. We present the performance on the training, validation and test set. The network does not appear to over-fit to the training data, and actually shows better performance on the validation set. On the test set the performance is also comparable to the training set. As the objects are symmetric to varying levels, the keypoint accuracy despite being low, still gives good pose estimations. This is seen in Fig. 1, where the symmetry of the object results in striped matching of keypoints. However, these matches are still very useful for the pose estimation. This is further shown in Fig. 5.

To analyze the network, performance for each component is shown in Tab. II. The two objects with the smallest loss is "3" and "5". These two object are both very similar to objects in the training data. The most challenging object is "2". The split between the two parts of the object makes it very dissimilar to the training data. The other components perform very well, especially for the segmentation task.

**Training Samples:** An experiment was performed to test the influence of the number of training objects. Ten percent of the objects were randomly sampled to be used for training. The number of iterations per epoch was kept the same as in the original training. While the results in Tab. I show a slight decrease in accuracy, it is seen that even by 10 % of the training data the model generalizes well. This is possibly a

TABLE II

LOSS AND ACCURACY OF EACH OF THE INDIVIDUAL TEST OBJECTS.

Object	1	2	3	4	5	6	7
Loss	1.50	2.03	1.12	1.53	1.42	1.84	1.53
Seg. Acc	0.99	0.81	0.98	0.98	0.95	0.97	0.99
Key. Acc	0.24	0.16	0.43	0.29	0.31	0.22	0.22

TABLE III

POSE ESTIMATION RECALL FOR EACH OF THE TEST COMPONENTS. THE "%" SIGN INDICATES THE AMOUNT OF NOISE ADDED TO THE SCENE POINT CLOUD.

Object	1	2	3	4	5	6	7	Avg.
Ours	0.95	0.82	0.99	0.77	0.91	0.90	0.92	0.89
FPFH	0.55	0.62	0.61	0.30	0.36	0.67	0.51	0.52
Ours 1%	0.93	0.79	0.99	0.74	0.86	0.88	0.90	0.87
FPFH 1%	0.43	0.57	0.50	0.25	0.31	0.66	0.40	0.44
Ours 5%	0.55	0.60	0.91	0.62	0.83	0.82	0.53	0.69
FPFH 5%	0.30	0.54	0.45	0.20	0.43	0.56	0.25	0.39

result of the simplified domain problem and indicates that the amount of real training data necessary is not overwhelmingly large.

### C. Pose Estimation Performance

To test the pose estimation accuracy of the developed method we compare it with the classic pose estimation method FPFH [14]. For both methods, RANSAC is used [1]. Additionally, we test our method using the GPU based RANSAC. The performance is measured by the ADI score presented in [10], as it is well suited for the symmetric objects in the dataset. The classic method obtains a 0.57% mean accuracy whereas our method obtains a 0.95% mean accuracy. Using the GPU based RANSAC the mean accuracy is 0.86%. A slightly lower performance. Our method thus vastly outperforms the classic pose estimation method.

As in [12] Gaussian noise is added to the points clouds to further test the robustness of the system. The Gaussian noise is tested at 1 % and 5 % of the longest axis of the object. The results are shown in Tab. III. It can be seen that our method outperforms the classic method for all objects. When adding noise the difference becomes even more pronounced. However, for the 5% noise the performance also drops significantly for our method.

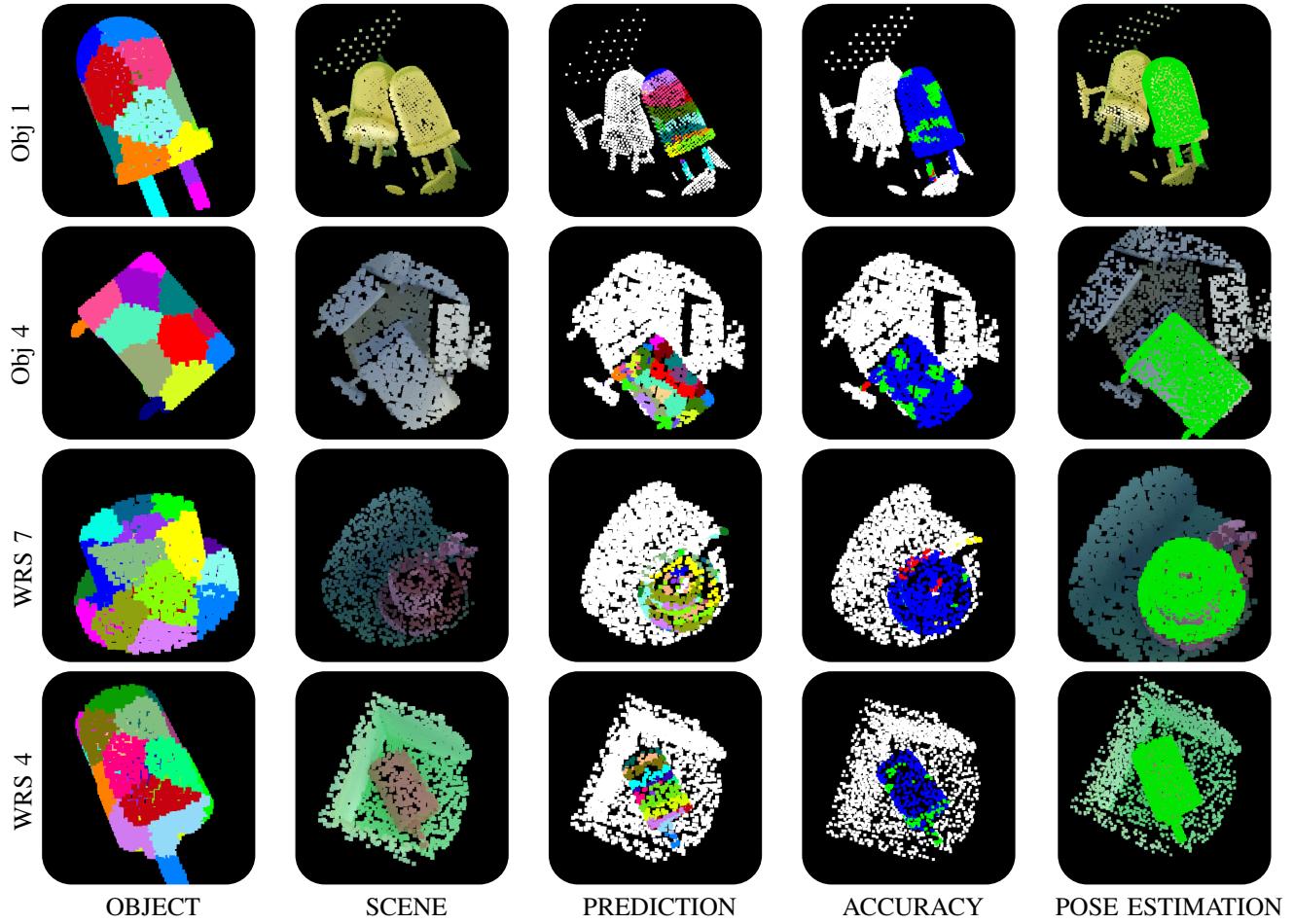


Fig. 5. Visualization of network output and resulting pose estimation using the RANSAC from Open3D [36]. The prediction accuracy is shown as follows: "White" represents correctly predicted background, "red" false negatives and "yellow" false positives. "Blue" is correctly predicted segmentation of the object, but wrong keypoints and "green" is both correct segmentation and keypoint prediction. It is seen that segmentation prediction is very high, and while the keypoint accuracy is not very high the pose estimation are correct.

TABLE IV

POSE ESTIMATION RECALL ON THE OUT-OF-CLASS DATASET. THE OBJECTS ARE NUMBERED ACCORDING TO [7]. THE "%" SIGN INDICATES THE AMOUNT OF NOISE ADDED TO THE SCENE POINT CLOUD.

Number Type	4 Motor	11 Pulley	13 Idler	7 Bear.	8 Shaft	14 Screw	5 Pulley	Avg.
Ours	1.00	0.99	0.90	0.91	0.93	0.93	0.96	0.95
GPU	1.00	0.99	0.98	0.96	0.90	0.97	0.98	0.97
FPFH	0.98	0.93	0.78	0.78	0.67	0.37	0.90	0.77
Ours 1%	1.00	0.99	0.89	0.94	0.93	0.92	0.96	0.95
GPU 1%	1.00	0.99	0.97	0.97	0.89	0.96	0.98	0.97
FPFH 1%	0.95	0.91	0.75	0.79	0.65	0.30	0.89	0.95
Ours 5%	0.97	0.98	0.87	0.94	0.82	0.73	0.95	0.89
GPU 5%	0.97	0.97	0.88	0.95	0.67	0.70	0.92	0.87
FPFH 5%	0.67	0.89	0.74	0.82	0.57	0.28	0.80	0.68

**Testing out of class:** The out-of-class dataset consists of industrial objects, such as motors and pulleys. The objects were used for the WRS assembly challenge held in 2018 [7]. The objects were chosen as they represent an industrial challenge, and because of the variety. To further test the robustness of the system, varying levels of noise is added to the points clouds.

The results of the network performance and pose estimation is shown in Tab. IV. It is seen that our developed method obtains very high recall on these objects that appear quite different compared with the training data. When adding noise the difference becomes even more pronounced. The ability to correctly compute features for both in and out-of-class objects is shown in Fig. 5. The GPU based method actually shows higher recall than the comparison method. However, there is a noticeable drop for WRS 8 (Shaft) at 5 % noise.

#### D. Experiments on real data

To further verify the effectiveness of the developed method, experiments have been performed on real data. Real point clouds have been collected for "Obj 4" and "Obj 7" of the test set. The point clouds were collected using a Zivid2<sup>3</sup> sensor. For both objects twenty different scenes were created, with the number of objects in each scene increasing from one to twenty. For each scene, five point clouds were obtained with different viewpoints. By omitting poses not visible the

<sup>3</sup><https://www.zivid.com/zivid-2>

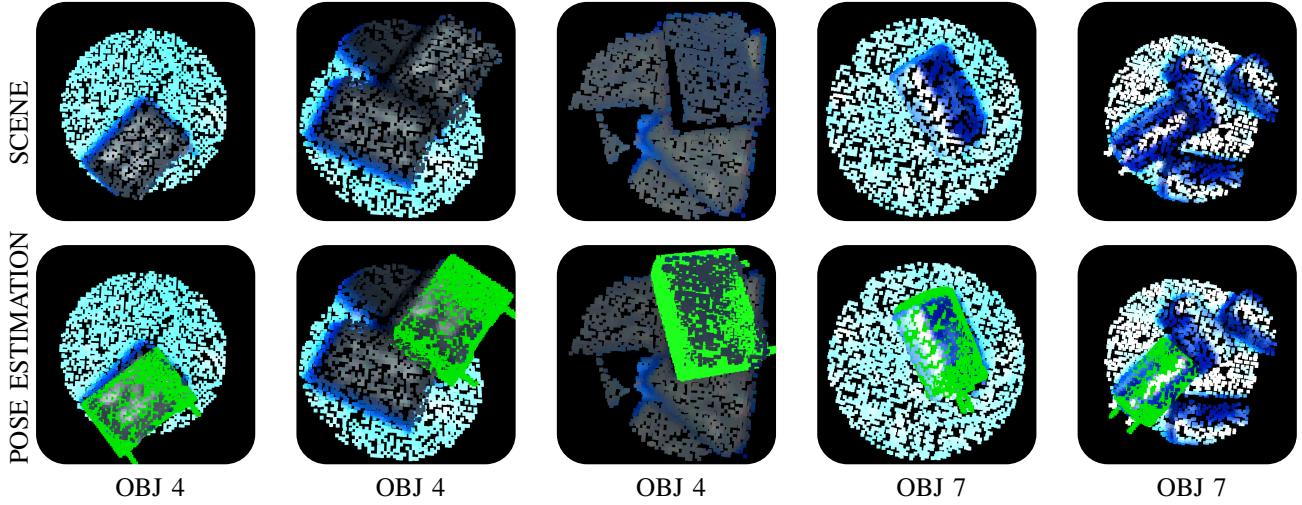


Fig. 6. Examples of pose estimations in the real dataset. For each point cloud the scene is shown above and the pose estimation is shown underneath.

TABLE V  
POSE ESTIMATION RECALL ON THE REAL DATA.

	FPFH	ParaPose	Ours	GPU	GPU w/o normal
Obj 4	0.16	0.62	0.38	0.69	0.41
Obj 7	0.18	0.59	0.33	0.77	0.63

final number of poses for "Obj 4" and "Obj 7" is 724 and 642, respectively.

Our method is compared with both the classical method and a state-of-the-art pose estimation method. The state-of-the-art method is a variant of ParaPose [29] adapted for colorless pose estimation [43]. The method is trained according to [29] with synthetic data of the CAD models in the bin, similar to the training data shown in Fig. 3. The results for both objects are shown in Tab. V.

It is seen that our method outperforms the classic pose estimation method by a large margin, and as expected the recall is lower compared with the results on the synthetic test data. ParaPose [29] also shows very good recall, and outperforms our method by a large degree. However, when using the GPU based RANSAC our method show performance comparable with ParaPose. The increase in performance from using the GPU based RANSAC is possibly as a result of the normal vector check as shown by the decrease in recall when omitted.

The results demonstrate that our method is able to perform pose estimation in real data, even when trained on synthetic data. We also show that the results are comparable to a state-of-the-art pose estimation method trained on the CAD model.

### E. Run-time

The run-time of the network was tested both with and without computing the object features at run-time. With the object features computed at run-time the processing lasts 14.9 ms. While by pre-computing the features the run-time is only 7.9 ms. The separation of object and scene feature computations is thus a significant speed-up.

Using Open3D [36] for pose estimation the full run-time is currently 84.7 ms. This is mainly related to the RANSAC

which is computed on the CPU. By using the GPU based RANSAC the run-time is reduced to 19.9 ms. This run-time allows the processing of 50 point clouds per second.

### V. CONCLUSION

This paper presents a novel method for colorless zero-shot pose estimation. The main contribution of the paper is the novel network structure with independent object and scene feature computation, along with a dataset of 1,500 electrical components in a homogeneous bin-picking scenario. The method shows very good generalizability across different objects, including out-of-class objects. The method is also demonstrated on real data where it obtains performance comparable with a state-of-the-art method trained on the test object. This proves the validity of creating object independent networks for specific scenarios, which can be useful for many real world applications.

In future work, it will be very interesting to obtain real training data to test the networks ability to learn a real scene.

The objects from the MegaPose6D [6] dataset could also be used to diversify the object types, and test on benchmark datasets.

### ACKNOWLEDGEMENT

This project was funded in part by Innovation Fund Denmark through the project MADE FAST, in part by the SDU I4.0-Lab.

### REFERENCES

- [1] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
- [2] F. Hagelskjær, A. G. Buch, and N. Krüger, "Does vision work well enough for industry?" in *VISIGRAPP (4: VISAPP)*, 2018, pp. 198–205.
- [3] T. Hodaň, M. Sundermeyer, B. Drost, Y. Labb  , E. Brachmann, F. Michel, C. Rother, and J. Matas, "Bop challenge 2020 on 6d object localization," in *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 2020, pp. 577–594.

- [4] M. Sundermeyer, T. Hodan, Y. Labbe, G. Wang, E. Brachmann, B. Drost, C. Rother, and J. Matas, “Bop challenge 2022 on detection, segmentation and pose estimation of specific rigid objects,” *arXiv preprint arXiv:2302.13075*, 2023.
- [5] M. Gou, H. Pan, H.-S. Fang, Z. Liu, C. Lu, and P. Tan, “Unseen object 6d pose estimation: a benchmark and baselines,” *arXiv preprint arXiv:2206.11808*, 2022.
- [6] Y. Labb  , L. Manuelli, A. Mousavian, S. Tyree, S. Birchfield, J. Tremblay, J. Carpenter, M. Aubry, D. Fox, and J. Sivic, “Megapose: 6d pose estimation of novel objects via render & compare,” *arXiv preprint arXiv:2212.06870*, 2022.
- [7] Y. Yokokohji, Y. Kawai, M. Shibata, Y. Aiyama, S. Kotosaka, W. Uemura, A. Noda, H. Dobashi, T. Sakaguchi, and K. Yokoi, “Assembly challenge: a robot competition of the industrial robotics category, world robot summit—summary of the pre-competition in 2018,” *Advanced Robotics*, vol. 33, no. 17, pp. 876–899, 2019.
- [8] T. Hodan, F. Michel, E. Brachmann, W. Kehl, A. GlentBuch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis *et al.*, “Bop: Benchmark for 6d object pose estimation,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 19–34.
- [9] M. Ulrich, C. Wiedemann, and C. Steger, “Combining scale-space and similarity-based aspect graphs for fast 3d object recognition,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 10, pp. 1902–1914, 2012.
- [10] 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 *Asian conference on computer vision*. Springer, 2012, pp. 548–562.
- [11] W. Kabsch, “A solution for the best rotation to relate two sets of vectors,” *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, vol. 32, no. 5, pp. 922–923, 1976.
- [12] A. G. Buch, L. Kiforenko, and D. Kraft, “Rotational subgroup voting and pose clustering for robust 3d object recognition,” in *2017 IEEE international conference on computer vision (ICCV)*. IEEE, 2017, pp. 4137–4145.
- [13] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, and N. M. Kwok, “A comprehensive performance evaluation of 3d local feature descriptors,” *International Journal of Computer Vision*, vol. 116, pp. 66–89, 2016.
- [14] R. B. Rusu, N. Blodow, and M. Beetz, “Fast point feature histograms (fpfh) for 3d registration,” in *2009 IEEE international conference on robotics and automation*. IEEE, 2009, pp. 3212–3217.
- [15] F. Hagelsk  r and A. G. Buch, “Pointvotenet: Accurate object detection and 6 dof pose estimation in point clouds,” in *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 2641–2645.
- [16] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, “Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again,” in *IEEE International Conference on Computer Vision*, 2017, pp. 1521–1529.
- [17] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, “Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes,” *Robotics: Science and Systems*, 2018.
- [18] Z. Li, G. Wang, and X. Ji, “Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 7678–7687.
- [19] B. Tekin, S. N. Sinha, and P. Fua, “Real-time seamless single shot 6d object pose prediction,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 292–301.
- [20] M. Rad and V. Lepetit, “Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth,” in *IEEE International Conference on Computer Vision*, 2017, pp. 3828–3836.
- [21] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, “Pvnnet: Pixel-wise voting network for 6dof pose estimation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4561–4570.
- [22] T. Hodan, D. Barath, and J. Matas, “Epos: Estimating 6d pose of objects with symmetries,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11703–11712.
- [23] S. Zakharov, I. Shugurov, and S. Ilic, “Dpod: 6d pose object detector and refiner,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1941–1950.
- [24] C. Wang, D. Xu, Y. Zhu, R. Mart  n-Mart  n, C. Lu, L. Fei-Fei, and S. Savarese, “Densefusion: 6d object pose estimation by iterative dense fusion,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3343–3352.
- [25] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.
- [26] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun, “Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11632–11641.
- [27] F. Hagelsk  r and A. G. Buch, “Bridging the reality gap for pose estimation networks using sensor-based domain randomization,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 935–944.
- [28] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, “Dynamic graph cnn for learning on point clouds,” *ACM Transactions on Graphics*, 2019.
- [29] F. Hagelsk  r and A. G. Buch, “Parapose: Parameter and domain randomization optimization for pose estimation using synthetic data,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 6788–6795.
- [30] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, “Learning 6d object pose estimation using 3d object coordinates,” in *European conference on computer vision*. Springer, 2014, pp. 536–551.
- [31] I. Shugurov, F. Li, B. Busam, and S. Ilic, “Osop: a multi-stage one shot object pose estimation framework,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 6835–6844.
- [32] Y. He, Y. Wang, H. Fan, J. Sun, and Q. Chen, “Fs6d: Few-shot 6d pose estimation of novel objects,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 6814–6824.
- [33] V. N. Nguyen, Y. Hu, Y. Xiao, M. Salzmann, and V. Lepetit, “Templates for 3d object pose estimation revisited: generalization to new objects and robustness to occlusions,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 6771–6780.
- [34] V. N. Nguyen, Y. Du, Y. Xiao, M. Ramamonjisoa, and V. Lepetit, “Pizza: A powerful image-only zero-shot zero-cad approach to 6 dof tracking,” *arXiv preprint arXiv:2209.07589*, 2022.
- [35] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, “Dynamic graph cnn for learning on point clouds,” *AcM Transactions On Graphics (tog)*, vol. 38, no. 5, pp. 1–12, 2019.
- [36] Q.-Y. Zhou, J. Park, and V. Koltun, “Open3D: A modern library for 3D data processing,” *arXiv:1801.09847*, 2018.
- [37] K. S. Arun, T. S. Huang, and S. D. Blostein, “Least-squares fitting of two 3-d point sets,” *IEEE Transactions on pattern analysis and machine intelligence*, no. 5, pp. 698–700, 1987.
- [38] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alch   Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.
- [39] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi, and H. Katam, “Blenderproc,” *arXiv preprint arXiv:1911.01911*, 2019.
- [40] F. Hagelsk  r and D. Kraft, “In-hand pose estimation and pin inspection for insertion of through-hole components,” in *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2022, pp. 382–389.
- [41] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [42] Y. Wu and K. He, “Group normalization,” in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.
- [43] F. Hagelsk  r, K. H. Lorenzen, and D. Kraft, “Off-the-shelf bin picking workcell with visual pose estimation: A case study on the world robot summit 2018 kitting task,” *arXiv preprint arXiv:2309.16221*, 2023.