InstancePose: Fast 6DoF Pose Estimation for Multiple Objects from a Single RGB Image

Lee Aing[1], Wen-Nung Lie[1,2,3], Jui-Chiu Chiang[1,2], Guo-Shiang Lin[4]

[1]Department of Electrical Engineering
[2]Center for Innovative Research on Aging Society (CIRAS)
[3]Advanced Institute of Manufacturing with High-tech Innovations (AIM-HI)
National Chung Cheng University (CCU), Taiwan
[4]Dept. of Computer Science and Information Engineering, National Chin-Yi University of Technology

Abstract

6DoF object pose estimation depends on positional accuracy, implementation complexity and processing speed. This study presents a method to estimate 6DoF object poses for multi-instance object detection that requires less time and is accurate. The proposed method uses a deep neural network, which outputs 4 types of feature maps: the error object mask, semantic object masks, center vector maps (CVM) and 6D coordinate maps. These feature maps are combined in post processing to detect and estimate multi-object 2D-3D correspondences in parallel for PnP RANSAC estimation. The experiments show that the method can process input RGB images containing 7 different object categories/ instances at a speed of 25 frames per second with competitive accuracy, compared with current state-of-the-art methods, which focus only on some specific conditions.

1. Introduction

Many studies [8, 14, 9, 10, 27, 15, 1, 18, 19, 25, 17, 4, 13, 24, 11] involve 6DoF object pose estimation using a single RGB image. Most seek to achieve greater accuracy in recognition/estimation, but do not consider practical issues, such as speed and memory requirement. Common application scenarios involve single and several, objects. Some objects are duplicated (with different poses) but some are different and can be occluded. The inference time for pose estimation for all objects in a scenario is also important.

This study uses more features to give a more realistic estimation system. The proposed method would be very accurate and simultaneously estimates the poses of all instance objects in a single RGB image, even if they belong to different categories (see the example in Fig. 1). The proposed method uses a bottom-up approach [4] to firstly detect all object features and identify the object categories and locations later. This type of approach allows the system to run fast at the expense of detection accuracy.

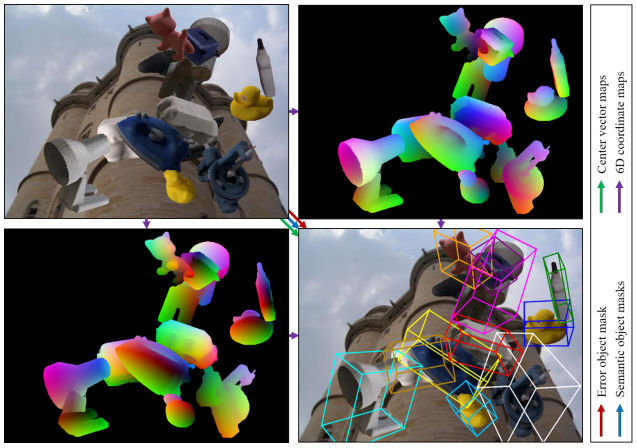


Figure 1. The proposed InstancePose uses a single RGB image composed from LINEMOD rendered objects [7] with PASCAL-VOC [5] background. The top-right and bottom-left image represent the 6D coordinate maps: one is the 3D coordinate map from the front-view and the other is the 3D coordinate map from the rear-view of the objects toward the camera. The bottom-right image shows all estimated 6DoF object poses.

The proposed system uses one of the deep neural network architectures of the Res2Net [6] family as the backbone and outputs four types of feature maps: the error object mask, the semantic object masks, the CVM, and the 6D coordinate maps (see Fig.1). These are used to derive the

poses of all instance objects. The error object mask is used to refine the predicted semantic object masks. This feature's ground truth is generated and trained based on the quality of the predicted 6D coordinate maps.

When the predicted object masks in different categories are refined, CVM is used to distinguish masks in the same category. This local mask discrimination is achieved using an instance center-voting procedure that is called Non-Maximum Preservation (NMP). In contrast to Non-Maximum Suppression (NMS) [21], NMP generates the voting hypotheses by preserving the non-maximum elements that are used to determine the locally dominant detections. Therefore, none of the objects in the image are missed.

Using this detection by NMP, the 2D-3D correspondences for each instance object are constructed and then PnP RANSAC [12] is used to estimate the 6DoF object poses. This study uses the 6D coordinate maps, which are derived from two concatenated 3D coordinate maps that are captured from two opposite viewpoints. Executing PnP RANSAC with 6D coordinate maps is more robust than the use of traditional 3D coordinate maps alone because they would provide more abundant information and more constraints for the PnP solver.

A review of related work is detailed in Section 2. In Section 3, the methodology is described and the experimental results and ablation study are detailed in Section 4. Finally, a conclusion is drawn in Section 5.

The main contributions of this research are as follows.

- A fast bottom-up approach is proposed that estimates the 6DoF poses for multiple instance objects in the same or different object categories in a single RGB image.
- A convolutional neural network with a compact architecture is used and this performs well for multiple instance objects in a single RGB image.
- Novel output feature maps, such as a Center Vector Map (CVM) and 6D coordinate maps, identify the dominant detections and restrict the PnP solver, respectively.
- Non-Maximum Preservation (NMP) is used, which is a new, robust, fast, and feasible post-processing system that preserves non-maximum elements to distinguish 2D-3D correspondences in parallel between different instance objects for a PnP RANSAC estimation.

2. Related Work and Analysis

In the field of 6DoF object pose estimation, most studies concentrate on correctness or accuracy, and only a few consider practical issues for real situations. In reality, systems that are specially designed to achieve high performance can be difficult to upgrade and are not applicable to real situations because of the speed and hardware requirement. After adding and stacking several stages to fulfill a condition of multiple object pose estimation, some systems [8, 14, 19] become slow or the network structures become more complex and performance decreases significantly. The proposed method addresses these disadvantages. The related works can be classified into several categories.

Simple but insufficient: The pioneer framework predicts target outputs directly. Some proposed methods [1, 26, 14] predict the object pose (3-translation and 3-rotation or 4-quaternion vectors). These methods involve initially cropping the regions of interest (ROIs) and then these are sent for inference of pose parameters. These pipelines are simple and straightforward to implement, but they lack information about the detected objects. To increase the quality of the pose estimation, [26, 14] predicted more feature maps: center distances and 3D location fields are used to guide training. These additional features are still not sufficient because training with a loss function that is related only to the direct rotation and translation vector is very limited.

Efficient but restricted: An object detection scheme [22] uses an end-to-end network and real-time detection [23, 10]. This functions well but performance is limited by the output feature that is used. Two studies, [9, 19] predict the unit-vector fields to estimate the pose. This gives a good prediction but the post processing that is used to recover back the key points requires much time for each single object. To allow multi-object pose estimation, the system must be re-designed.

Complete but complex: For practical use, some studies [15, 4, 8] proposed complete systems to estimate multiple object poses. Their systems are complicated to implement because they involve multiple stages of training and tangled network features. One study [4] uses a bottom-up approach to estimate multiple human joints and these are later grouped and connected as multiple skeletons. To extend this process with more object categories, more features were added and predicted. Another study [8] predicts many network layers, in order to determine which fragment coordinate map belongs to which object instance. This increase the complexity of post processing and the inference time.

In conclusion, direct pose estimation does not give accurate results. The unit-vector field gives robustness because it involves voting but does not reliably estimate multiple object poses. However, a 3D coordinate map [15] can fulfil this requirement.

3. InstancePose

Fig. 2 shows that the proposed convolutional neural network uses the Res2Net50 [6] architecture as the back-

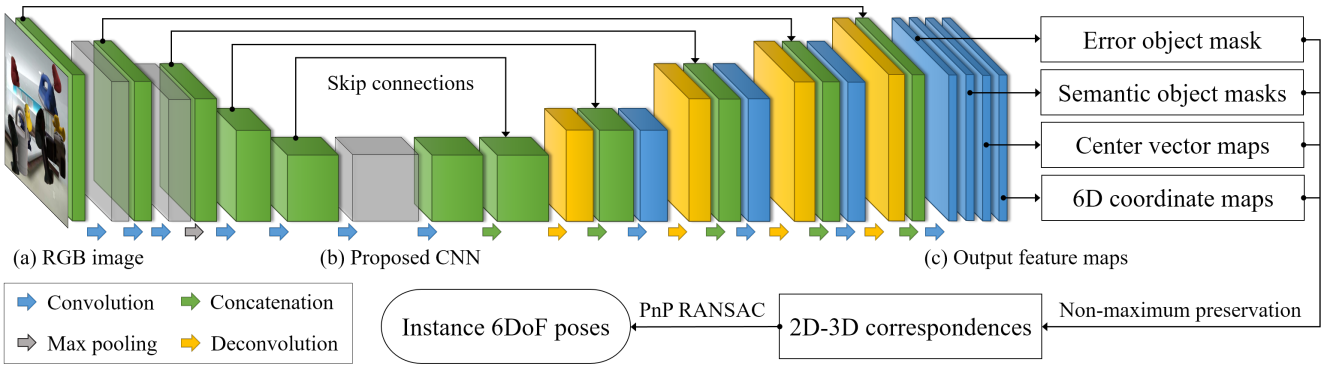


Figure 2. An overview of the training and testing network architecture for InstancePose: (a) the input is an RGB image (480×640×3), (b) the proposed CNN uses Res2Net50 [6] as the backbone and (c) the network output has 4 types of feature maps.

bone. It uses a single RGB image as input and outputs 4 types of feature maps: the error object mask, semantic object masks, center vector maps (CVM) and 6D coordinate maps. These features are combined to generate the instance 2D-3D correspondences using non-maximum preservation (NMP) and the 6DoF object poses are then etimated using PnP RANSAC.

3.1. Neural Network Design

For the backbone of Res2Net50, some parts are modified. The average pooling and fully connected layers are replaced by a 2D convolution layer with a kernel size of 3×3 and stride 1 with an output depth of 512, batch normalization and ReLU activation. As it passes through the encoder, six intermediate feature maps are used as the skip connections. These have different dimensions and depths of channels. In the decoder, the last two skip connections with the lowest dimensions are combined to produce a new feature map, which has a depth channel of 512, before up-sampling and concatenation. This is a special case that involves deconvolution and the following 4 skip connections with different dimension perform the concatenation and deconvolution in a similar way, as shown in Fig. 2. Finally, the output feature maps have dimensions of H×W×(1+C+1+2+6), where H and W are the height and width of the input, respectively, and C is the number of object classes.

3.2. Output Feature Maps

There are 4 output feature maps, as shown in Fig. 3, and each is trained using supervised ground truths that are calculated from the corresponding 3D object models, except for the error object mask. At the test/inference stage, the initially predicted CVM and 6D coordinate maps of the multiple instance objects are refined (see Section 3.3) by referring to the predicted error object mask and the semantic object mask. The function for total loss is:

$$l_{total} = \alpha l_{error} + \beta l_{mask} + \gamma l_{CVM} + \lambda l_{6D}, \quad (1)$$

where α, β, γ, and λ are the loss weights which are used to balance all sub-loss functions.

3.2.1 Error Object Mask

The ground truth for the error object mask is not generated directly from the 3D object models. This feature shows the pixel-wise confidence in the quality of the predicted 6D coordinate maps. In contrast to other score metrics, less confidence is better. In Fig. 3(b), the confidence score for the background is high (yellow background) and the confidence score for the predicted object masks is really low, except for the object, "*holepuncher*" (in the red dashed rectangle), which is occluded by "*can*" and "*duck*". This error object mask, which is denoted as e_p (at a pixel p), is evaluated using the error in the 6D coordinate maps:

$$e_p = \underset{i=1\sim6}{\text{Avg}}\,(e_p^i),\ e_p^i = \begin{cases} 1 & \text{, if } |\hat{e}_p^i| > \theta_e \\ |\hat{e}_p^i| & \text{, otherwise} \end{cases}, \quad (2)$$

where $\hat{e}_p^i = \hat{c}_p^i - c_p^i$ is the 6D coordinate error, $\hat{c}_p^i$ and c_p^i are the predicted i-th channel of the 6D coordinate map and its ground truth for the p-th pixel, respectively, θ_e is the error threshold and Avg(.) is an operator getting average along all 6 channels in the 6D coordinate map. If the error between the ground truth and the prediction is greater than a threshold, Eq. 2 gives a value of 1; otherwise, the error is retained and assigned as the confidence score. Therefore, the average error loss is defined by dividing the sum of errors by M, which is the set of pixels in the image:

$$l_{error} = \frac{1}{|M|} \sum_{p \in M} \|\hat{e}_p - e_p\|_2^2, \quad (3)$$

3.2.2 Semantic Object Masks

To predict the semantic segmentation (each detected object is assigned with a distinct label value, as shown in Fig.

(a) (b) (c) (d) (e) (f)

Figure 3. Output feature maps: (a) A single RGB image that is rendered from LINEMOD objects with a cluttered background, (b) the error object mask, (c) the semantic object masks (pixel labels are assigned according to Occlusion LINEMOD), (d) Center vector maps visualized with three color components — the third of which is set to be 1, and (e) and (f) 3D coordinate maps for the front and rear view, respectively

3(c)), the focal loss [16] is used instead of the cross-entropy loss because it balances the object classes better for small objects. Using focal loss in training allows more objects to be detected for multiple object segmentation, which helps a lot since the training system tries to focus only on feature maps, such as the CVM and 6D coordinate maps, if the cross entropy loss is used.

$$l_{mask} = \frac{1}{|M|} \sum_{p \in M} -\alpha_c (1 - \hat{y}_p)^{\gamma_c} \log(\hat{y}_p), \text{if } y_p = 1, \quad (4)$$

where α_c and γ_c are the hyper-parameters, and $\hat{y}_p$ and y_p are the predicted semantic object masks and their ground truth at p-th pixel, respectively.

3.2.3 Center Vector Maps

After semantic segmentation, only labels/ masks in each object category can be identified, but the number of objects in the same category is unknown. Center vector maps (CVM) address this problem by calculating the distances between all pixels and their corresponding object centers. This feature has only two components (horizontal and vertical), as shown in Fig. 3(d). The third component of the color space is 1. This feature is to allows pixels in each object category to vote for the object centers to which they belong. This is used to determine which pixels go with which objects. There are two ways to generate the ground truth for training: using the direct vectors between all object pixels and their corresponding centers and the vectors that are normalized to the width and height of the image. The experiments for this study use a direct vector technique because it can be trained more easily and gives more accurate predictions. The CVM loss function is written as Eq. 5:

$$l_{CVM} = \frac{1}{|M|} \sum_{p \in M} \frac{|\hat{v}_p - v_p|}{2s}, \quad (5)$$

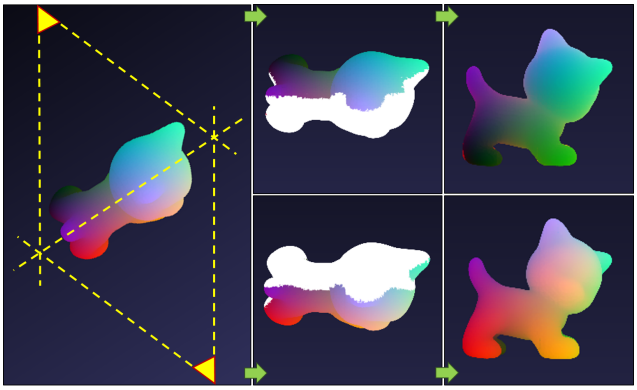
where $\hat{v}_p$ and v_p are the predicted CVM and its ground truth at p-th pixel, and s is a scale factor. Dividing by 2 means that the CVM map has two components.

3.2.4 6D Coordinate Maps

The 6D coordinate map is extended from the original 3D coordinate map [15, 18] by capturing another 3D coordinate map in an opposite viewpoint relative to a symmetrical plane. Fig. 4(a) shows the top view of the object CAD model, as well as the two viewpoints (i.e., front and rear) which are opposite and symmetrical to a plane constituted by the two main eigenvectors of the object's points clouds distribution. This study uses both the front- and rear-view features for training to impose more constraints on PnP transform solving. The experiments show that 3D coordinate maps give good performance but there are problems with predictions along the depth direction. For training using only 3D coordinate maps, using the ground truth to determine the object poses is 76% effective, which is much lower than the ideal performance. However, if the predicted erroneous depths are corrected by 1cm and all the other predicted parameter values are left the same, the accuracy with which a 3D transformation is evaluated is increased to 94%.

This initial verification shows that the depth is a key element so other 3D coordinate maps from the object's rear view in the depth direction are used to restrict the 3D transformation. The result for 6D coordinate maps is 98% without any corrections. This shows that a rear-view indeed improves the performance, similar to the conventional case that multi-view RGB input provides more information than a single RGB input. The same is true for the 6D coordinate maps, which are a multi-view depth input.

In order to generate the 6D coordinate map ground truth for training, a hidden point removal tool [28] is used to capture the point clouds in the front and rear views and these 3D coordinates are projected onto the same image plane and used as the color attributes for the pixels on which they are



(a) (b) (c)

Figure 4. 6D coordinate map: (a) The CAD model for the object "*cat*", viewed from the top, is observed from two points of view: the front and the rear view, (b) two groups of point clouds are captured from the front and rear view using hidden point removal and (c) the two groups of point clouds are projected onto the same image plane with the 3D coordinate color.

projected. Fig. 3(e)&(f) show the masked 3D coordinate map for the front and rear view, respectively. This process is shown in Fig. 4. To train a 6D coordinate feature, a smooth-L1 loss function is used, as shown in Eq. 6:

$$m_p = \underset{i=1\sim 6}{\text{Sum}}(m_p^i), \quad m_p^i = \begin{cases} \frac{1}{2\sigma}\hat{e}_p^{i\,2} & , \text{if } |\hat{e}_p^i| < \sigma \\ |\hat{e}_p^i| - \frac{\sigma}{2} & , \text{otherwise} \end{cases}, \quad (6)$$

where $\hat{e}_p^i = \hat{c}_p^i - c_p^i$ is the 6D coordinate error, $\hat{c}_p^i$ and c_p^i are the predicted i-th channel of the 6D coordinate maps and their ground truth at the p-th pixel and i-th channel, σ is a threshold and Sum(.) is the summation operator along all 6 channels in the 6D coordinate map. Therefore, the 6D coordinate loss is defined as Eq. 7, where division by 2 refers to the fact that two 3D coordinate maps must be optimized.

$$l_{6D} = \frac{1}{|M|}\sum_{p\in M}\frac{m_p}{2}, \quad (7)$$

3.3. Non-Maximum Preservation

As long as all of the feature maps are predicted by the network, they are combined to determine the instance 2D-3D correspondences. Firstly, the error object mask and the semantic object masks are merged to create a refined object mask by binarizing the error mask with a threshold ϕ_e and then applying pixel-wise multiplication. This refined object mask filters out faulty pixels, which cause large errors in the 6D coordinate maps.

Secondly, the NMP which is similar to the traditional non-maximum suppression (NMS) technique [21] is applied. First of all, CVM is initially used and combined with the refined object masks to generate the voting hypotheses for all instance objects. These voting hypotheses contain 3 data (coordinates of the object center and its label) and have less number than the pixels in the object masks. From these voting hypotheses, the dominant detections are then derived. This is an instance-center-voting procedure that determines the dominant detections without the use of the predicted confidences as the initial detections. The non-maximum elements or voting hypotheses which support those dominant detections are preserved to construct the 2D-3D correspondences for each instance object. This voting process occurs in a highly dimensional matrix and is executed in parallel so the system defines the 2D-3D correspondences quickly and without running in loops. The final step of NMP is to sample the supporting voting hypotheses randomly using an amount that is related to the size of the object masks. Sampling of the 2D-3D correspondences also accelerates the process, without a significant decrease in performance (see later experiments). The entire NMP algorithm is summarized in the flowchart in Fig. 5 and further detail about NMP is included in the supplementary material

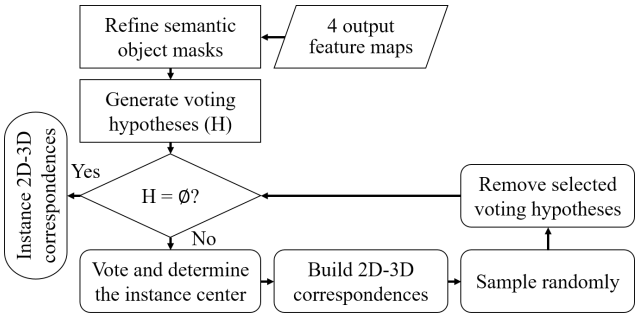


Figure 5. The flowchart for non-maximum preservation.

3.4. 6DoF Pose Estimation

When the 2D-3D correspondences for each instance object are randomly selected, PnP RANSAC is used to estimate the 6DoF poses. This process is performed individually for each object but it is much quicker than NMP. Since there will be a pair of 3D coordinates (from the front and rear views) corresponding to each 2D object pixel, randomization of one of these (front-view or rear-view correspondence) is conducted for generating PnP candidates. The experiments for this study show that randomizing the PnP candidates gives better results than concatenation.

4. Experiment and Analysis

4.1. Experimental Details

All experiments used a platform of Core i9 CPU with GeForce RTX 2080 Ti GPU in Ubuntu 18.04 environment. The maximum processing speed for this method with an

Metrics	2DPro					ADD(S)					
Methods	PoseCNN [26]	S-Driven [10]	PVNet [19]	S-Stage [9]	Ours	PoseCNN [26]	S-Driven [10]	Pix2Pose [18]	PVNet [19]	S-Stage [9]	Ours
ape	34.60	59.10	69.14	**70.30**	62.67	9.60	12.10	22.00	15.81	19.20	**22.52**
can	15.10	59.80	**86.09**	85.20	80.36	45.20	39.90	44.70	63.50	**65.10**	60.56
cat	10.40	46.90	65.12	**67.20**	54.98	0.90	8.20	**22.70**	16.68	18.90	21.20
driller	7.40	59.00	61.44	71.80	**83.03**	41.40	45.20	44.70	65.65	69.00	**71.99**
duck	31.80	42.60	**73.06**	63.60	67.35	19.60	17.20	15.00	25.24	25.30	**27.52**
eggbox	1.90	11.90	8.43	**12.70**	0.34	22.00	22.10	25.20	50.17	**52.00**	41.44
glue	13.80	16.50	55.37	56.50	**58.99**	38.50	35.80	32.40	49.62	51.40	**56.20**
hpuncher	23.10	63.60	69.84	71.00	**77.24**	22.10	36.00	49.50	39.67	45.60	**46.28**
Average	17.20	44.90	61.06	**62.30**	60.62	24.90	27.00	32.00	40.77	43.30	**43.46**

Table 1. The 2D projection error "2DPro" and the average 3D distance error "ADD(S)", compared with other state-of-the-art methods for the Occlusion LINEMOD dataset.

input image of 480×640 pixels and 7 different object categories is about 25 FPS (frames per second), which is an average of 40ms per frame. Data loading requires 3ms, the network inference requires 25ms and the remainder of the time is required for post processing for all of the instance objects.

The proposed system is trained using the Adam optimizer with 140 epochs and a batch size of 6. The learning rate is initially 0.001 and is reduced by half automatically every 20 epochs. The error threshold for the computation of error object mask is $\theta_e = 0.1$ (Eq. 2) and for the training of semantic segmentation, the hyper-parameters are $\alpha_c = 1$ and $\gamma_c = 2$ (Eq. 4) for all object categories. The scale factor that is used in the CVM is $s = 50$ (Eq. 5) and for the 6D loss function, $\sigma = 0.1$ (Eq. 6). The loss weights α, β, γ , and λ (Eq. 1) are [1, 1, 1, 1]. The binarization threshold for the error object mask (Section 3.3) is $\phi_e = 0.4$.

4.2. Datasets

The datasets that are used for training come from the Normal LINEMOD dataset [7] and the rendered dataset [19]. The Normal LINEMOD dataset contains 15,783 images from 13 benchmark objects in lighting conditions that are cluttered, texture-less and poor: each object category accounts for about 1,200 images. Only 8 object categories are used for training and testing and these are also included in Occlusion LINEMOD [2]. Only 15% of the total images for each object in the Normal LINEMOD dataset were used for training, and the remaining 85% were used for testing. The Occlusion LINEMOD was used for testing only [23].

The testing dataset contains 1,214 images that are heavily occluded form all objects. The original Normal LINEMOD dataset cannot be used directly because other surrounding objects are not annotated so the target objects are cropped and placed arbitrarily, but occluding each other, in a cluttered background [5]. For the rendered dataset, CAD are used models to generate synthetic data with different backgrounds from PASCAL-VOC dataset [5]. There are more than 20,000 training images: half from the real synthetic dataset (real objects in a cluttered background) and half from the rendered synthetic dataset (rendered objects in clutter background). During training, the data is augmented by cropping, shifting, rotating, coloring and resizing, in order to avoid overfitting.

4.3. Evaluation Metrics

There are three evaluation metrics: the 2D projected error "2DPro" [3], the average distance error "ADD(S)" [26] and the 5-centimeter 5-degree "5CMD" [20]. 2DPro measures the average pixel error between the projections of all point clouds that are transformed using the estimated and the target 6DoF poses. If the average error is less than 5 pixels, the estimated object pose is correct; otherwise, it is pruned. ADD(S) measures the average 3D error between the prior two sets of transformed point clouds, without projection. The correctness in terms of ADD(S) is determined by counting the samples with errors that are less than 10% of the target object's diameter. For 5CMD, if the estimated transformation error is less than 5 centimeters for the translation and less than 5 degrees for the rotation, the prediction is correct; otherwise, it is discarded.

4.4. Results for Occlusion LINEMOD

The results in Table 1 show that the correctness score does not outperform all of the state-of-the-art methods. However, in terms of the processing speed, the results in Table 3 show it to be the best method. Table 1 shows that for in 2DPro metric, there is a really low score for "*eggbox*" because this object is symmetrical in shape and occluded by other objects. To allow evaluation for symmetrical objects, the ADD(S) metric searches for the smallest error between the transformed point clouds for the estimation and the ground truth. Ambiguity or confusion are features of 2DPro that reduce performance, since the appearances in 2D images might be similar even the poses in training and testing are totally different. However, for the object "*glue*",

Metrics	5CMD		2CMD	5CMD	10CMD
Methods	DeepIM [13]	PVNet [19]	Ours		
ape	**51.80**	39.40	3.68	29.79	66.78
can	35.80	68.60	23.94	**69.01**	92.54
cat	12.80	**20.90**	1.60	15.88	42.82
driller	45.20	63.90	29.82	**73.56**	94.32
duck	**22.50**	15.60	3.01	21.15	57.96
eggbox	**17.80**	0.60	0.00	0.08	0.77
glue	**42.70**	19.80	3.46	32.51	67.93
hpuncher	18.80	47.70	11.55	**56.32**	90.29
Average	30.93	34.56	9.63	**37.29**	64.18

Table 2. The performance in terms of "5CMD" and other similar metrics, compared with other state-of-the-art methods for the Occlusion LINEMOD dataset.

Metrics	Time consumption (ms)				
Methods	[10]	[18]	[19]	[9]	Ours
Data loading	-	-	10.90	-	2.48
Net. inference	30.00	76.00	3.30	14.00	25.59
Post-processing	20.00	25.00	25.90	8.00	11.44
Total time	50.00	101.00	40.01	22.00	39.51
Object number	5	1	1	1	**7**

Table 3. Performance in terms of average time required, which includes the time for data loading, network inference and post-processing, using the LINEMOD dataset.

which is only locally symmetrical, the performance is acceptable. This shows that the proposed 6D coordinates feature partially eliminates ambiguity. Some examples of predictions of the output feature maps and 6DoF object poses using the proposed method are included in the supplementary material.

Table 2 shows the performance for the proposed method in terms of the 5CMD metric, compared with two state-of-the-art methods. The performance for different criteria, such as 2 centimeters 2 degrees (2CMD), and 10 centimeters 10 degrees (10CMD), are also shown.

In Table 3, there are three time curricula: the time of data loading, the network inference and post-processing. The proposed method requires the least time, with an average of only 39ms for 7 objects in different categories. Most of the other methods use less than 7 objects and require more time.

4.5. Results for Synthetic LINEMOD

Table 4 shows the performance for testing the synthetic LINEMOD datasets, which contain 3 different types of collections that are generated from real images in the Normal LINEMOD dataset and rendered images using LINEMOD CAD models. The real test images contain 8 objects only and 85% of the Normal LINEMOD, which are not used for training. These are cropped and pasted arbitrarily on cluttered backgrounds [5] with heavy occlusion. Objects in the same category with different poses are also used, in order to verify that the proposed system can estimate the instance object poses in less time.

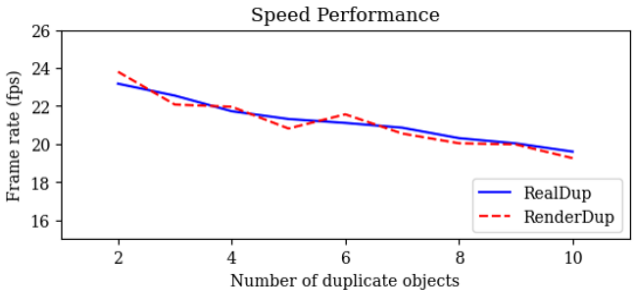


Figure 6. The speed plot for the proposed system (frame rate) with respect to the number of duplicated objects in two synthetic datasets.

4,000 testing images are generated by randomly placing a maximum of three objects in the same category in each RGB image. There are 13 objects with different poses. This dataset is named, RealDup. The same process is used for the rendered images with rendered objects and different lighting conditions, and this is named, RenderDup. The last test dataset does not include the duplicated objects, but which has all object categories, is called, RenderSyn. The number of detected objects is counted with reference to the ground truth. This is named the detected objects "DO" metric (measured as a percentage with respect to the number of all ground truth objects).

The results in Table 4 show that the proposed system does not detect all of the objects because small objects are heavily occluded. Some objects, such as "*driller*", are detected more than 100% because the semantic segmentation is faulty. However, the system still performs well in terms of speed, even if objects in the same category are duplicated when the results for "RenderDup" and "RenderSyn" are compared. For this synthetic dataset experiment, RenderDup and RenderSyn are more general than RealDup so the performance is degraded for rendered datasets.

Fig. 6 shows the frame rate for the proposed system with respect to the number of duplicate objects. The datasets for this experiment contain 200 images with 15 objects for each frame and are randomly generated using duplicate objects from 2 to 10. The plot shows that the frame rate slightly decreases from about 23 FPS to 19 FPS as the number of duplicated objects increases.

4.6. Ablation Study

In order to obtain the best performance for the proposed method, some parameters must be tuned and strategies are added. These strategies are simple but they save time and stabilize the system. Table 5 shows the results for the abla-

Datasets	RealDup				RenderDup				RenderSyn			
Metrics	2DPro	ADD(S)	5CMD	DO	2DPro	ADD(S)	5CMD	DO	2DPro	ADD(S)	5CMD	DO
ape	91.88	42.30	73.36	95.70	87.17	45.57	68.11	95.76	86.07	43.47	68.17	95.34
can	95.61	83.28	91.74	99.07	89.26	77.12	83.98	98.93	89.96	77.42	84.51	98.62
cat	93.04	65.33	80.62	96.83	86.73	61.08	72.23	96.69	86.59	61.53	72.70	96.14
driller	95.16	91.36	92.33	100.27	90.47	87.60	86.84	101.11	90.89	87.06	87.09	99.35
duck	92.18	52.98	79.36	96.92	87.06	53.31	74.03	96.34	87.62	51.31	72.95	96.72
eggbox	94.41	93.54	88.97	97.50	88.35	89.66	81.65	97.07	89.64	90.79	82.82	97.03
glue	93.02	86.19	77.19	96.79	86.39	82.59	70.85	96.09	85.53	82.73	70.31	95.48
hpuncher	93.36	72.00	86.74	97.69	89.12	67.24	81.90	98.35	89.31	67.24	82.51	97.74
Average	93.58	73.37	83.79	97.60	88.07	70.52	77.45	97.54	88.20	70.19	77.63	97.05
FPS	23.16				23.03				23.65			

Table 4. Performance for "2DPro", "ADD(S), "5CMD", "DO" and "FPS" for the synthetic datasets: RealDup, RenderDup, and RenderSyn. These are generated using the Normal LINEMOD dataset and rendered objects with heavy occlusion and cluttered backgrounds.

Metrics	2DPro	ADD(S)	5CMD	FPS	DO
RR	**60.62**	**43.46**	37.29	**25.31**	89.94
FF	60.58	43.46	37.29	18.90	89.94
FR	60.60	43.28	37.20	22.12	89.94
RF	60.62	43.32	**37.36**	22.37	89.94
3DF	54.15	34.78	32.30	22.47	89.94
3DR	52.95	35.37	27.27	22.27	89.94
3DFGT	99.64	82.80	97.78	22.42	100
3DRGT	99.71	76.91	99.11	22.38	100
6DGT	99.99	98.33	99.95	17.89	100

Table 5. The ablation study for randomized 2D-3D correspondences and PnP candidates for the Occlusion LINEMOD dataset, and performance for the 3D/6D coordinates map ground truths.

tion study for the proposed method for randomly generated voting hypotheses and PnP candidates.

In constructing 2D-3D correspondences for each instance object, many candidates are derived using the predicted CVM and 6D coordinates map. These are either fully or partially/randomly used. The disadvantage of fully using these candidates is that much time is required. Partial/random usage can reduce performance because important information is lost. Solving the PnP involves a similar issued because the 6D coordinates can be fully used or randomly selected from the front-view 3D or the rear-view 3D coordinates.

There are 4 combinations to generate the voting hypotheses and PnP candidates: random-random (RR), full-full (FF), full-random (FR) and random-full (RF). These are shown in Table 5. The performance for other variations, such as the front-view 3D coordinates map (3DF) or the rear-view 3D coordinates map (3DR), is also shown. The final study recovers and uses the ground truths for the 3D coordinates maps from the front and rear-view (3DFGT and 3DRGT) and 6D coordinate maps (6DGT). The performance for these is also shown.

The results in Table 5 show that RR outperforms the others. 3DF and 3DR are given poorer results than 6D coordinate maps in terms of all evaluation metrics. A comparison of 6DGT with 3DFGT and 3DRGT also shows that 6DoF pose estimation using 6D coordinates map is more accurate. The ADD(S) scores for 3DFGT and 3DRGT are very different because there is a blank projection (or, hole) when generating the ground truth for the 3D coordinates map using the rear-view. The average DO is about 90%, which reduces performance. More details are given in the supplementary material.

5. Conclusion

This study proposes a novel methodology to estimate multiple instance 6DoF object poses. The system is fast enough for practical use. The proposed output feature maps give good performance but the prediction for semantic segmentation and 6D coordinates map requires development. The results in Table 5 show that about 10% of the total instance objects in the Occlusion LINEMOD dataset are lost because they are small or heavily occluded. To generate the 3D coordinates map from the rear view, hidden point removal is used, but some small concavity parts that are visible in the front view may be invisible or have no information in the rear view. These are the principal problems to be addressed in future studies. The experiments should also use more challenging objects, such as texture-less and fast moving objects.

Acknowledgement: This study is financially supported by the Center for Innovative Research in Aging Society (CIRAS) and the Advanced Institute of Manufacturing with High-tech Innovations (AIM-HI) of The Featured Areas Research Center Program within the framework of the Higher Education Sprout Project of the Ministry of Education (MOE) in Taiwan.

References

[1] G. Billings and M. Johnson-Roberson. Silhonet: An rgb method for 6d object pose estimation. *IEEE Robotics and Automation Letters*, 4(4):3727–3734, 2019. 1, 2

[2] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 536–551, Cham, 2014. Springer International Publishing. 6

[3] E. Brachmann, F. Michel, A. Krull, M. Y. Yang, S. Gumhold, and C. Rother. Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3364–3372, 2016. 6

[4] Z. Cao, G. Hidalgo, T. Simon, S. E. Wei, and Y. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1):172–186, 2021. 1, 2

[5] Mark Everingham, S. Eslami, Luc Van Gool, Christopher Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111, 2014. 1, 6, 7

[6] S. H. Gao, M. M. Cheng, K. Zhao, X. Y. Zhang, M. H. Yang, and P. Torr. Res2net: A new multi-scale backbone architecture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(2):652–662, 2021. 1, 2, 3

[7] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In Kyoung Mu Lee, Yasuyuki Matsushita, James M. Rehg, and Zhanyi Hu, editors, *Computer Vision – ACCV 2012*, pages 548–562, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. 1, 6

[8] T. Hodaň, D. Baráth, and J. Matas. Epos: Estimating 6d pose of objects with symmetries. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11700–11709, 2020. 1, 2

[9] Y. Hu, P. Fua, W. Wang, and M. Salzmann. Single-stage 6d object pose estimation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2927–2936, 2020. 1, 2, 6, 7

[10] Y. Hu, J. Hugonot, P. Fua, and M. Salzmann. Segmentation-driven 6d object pose estimation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3380–3389, 2019. 1, 2, 6, 7

[11] 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 *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1530–1538, 2017. 1

[12] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81(2):155, 2008. 2

[13] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation. *International Journal of Computer Vision*, 128(3):657–678, 2020. 1, 7

[14] Z. Li and X. Ji. Pose-guided auto-encoder and feature-based refinement for 6-dof object pose regression. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8397–8403, May 2020. 1, 2

[15] Z. Li, G. Wang, and X. Ji. Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7677–7686, 2019. 1, 2, 4

[16] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327, 2020. 4

[17] Fabian Manhardt, Wadim Kehl, Nassir Navab, and Federico Tombari. Deep model-based 6d pose refinement in rgb. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 1

[18] K. Park, T. Patten, and M. Vincze. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7667–7676, 2019. 1, 4, 6, 7

[19] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4556–4565, 2019. 1, 2, 6, 7

[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 *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3848–3856, 2017. 6

[21] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. 2, 5

[22] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018. 2

[23] B. Tekin, S. N. Sinha, and P. Fua. Real-time seamless single shot 6d object pose prediction. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 292–301, 2018. 2, 6

[24] H. Tjaden, U. Schwanecke, and E. Schömer. Real-time monocular pose estimation of 3d objects using temporally consistent local color histograms. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 124–132, 2017. 1

[25] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2637–2646, 2019. 1

[26] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *Robotics: Science and Systems (RSS)*, 2018. 2, 6

[27] S. Zakharov, I. Shugurov, and S. Ilic. Dpod: 6d pose object detector and refiner. In *2019 IEEE/CVF International*

Conference on Computer Vision (ICCV), pages 1941–1950, 2019. 1

[28] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *CoRR*, abs/1801.09847, 2018. 4