Confidence-based 6D Object Pose Estimation

Wei-Lun Huang*, Chun-Yi Hung*, and I-Chen Lin†, *Member, IEEE*

***Abstract*—The aim of this paper is to estimate the six-degree-of-freedom (6DOF) poses of objects from a single RGB image in which the target objects are partially occluded. Most recent studies have formulated methods for predicting the projected two-dimensional (2D) locations of three-dimensional keypoints through a deep neural network and then used a PnP algorithm to compute the 6DOF poses. Several researchers have pointed out the uncertainty of the predicted locations and modelled it according to predefined rules or functions, but the performance of such approaches may still be degraded if occlusion is present.**

To address this problem, we formulated 2D keypoint locations as probabilistic distributions in our novel loss function and developed a confidence-based pose estimation network. This network not only predicts the 2D keypoint locations from each visible patch of a target object but also provides the corresponding confidence values in an unsupervised fashion. Through the proper fusion of the most reliable local predictions, the proposed method can improve the accuracy of pose estimation when target objects are partially occluded. Experiments demonstrated that our method outperforms state-of-the-art methods on a main occlusion data set used for estimating 6D object poses. Moreover, this framework is efficient and feasible for realtime multimedia applications.

***Index Terms*—6D pose estimation, prediction confidence formulation, deep neural network.**

I. Introduction

DETECTING objects in an image and estimating their six-degree-of-freedom (6DOF) poses, namely their three-dimensional (3D) rotations and translations, has been an important topic in the fields of multimedia and computer vision. It plays a crucial role in several applications, such as image content analysis, augmented reality, and robotic manipulation. Recently, many methods [1]–[4] involving depth data of an input image have produced compelling results. However, depth sensors require additional power consumption and are not extensively applied to portable computing devices, such as smartphones and tablets. They limit the applications of these methods based on red-green-blue-depth (RGB-D) information.

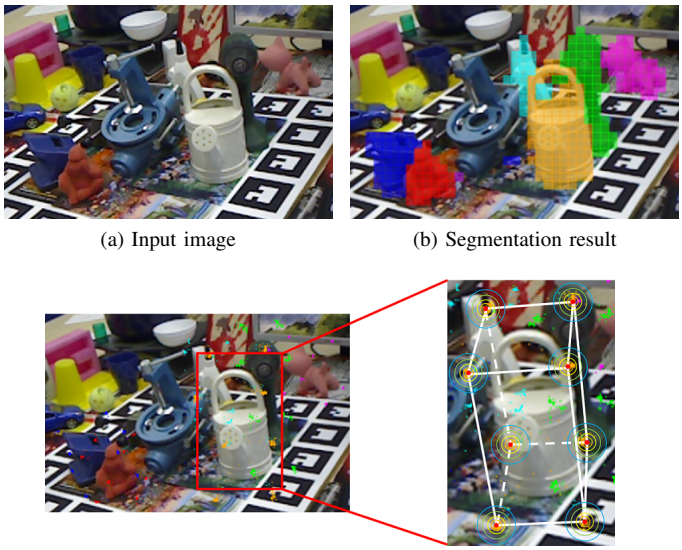
RGB images can be acquired with ordinary cameras, but 6D pose estimation from RGB images involves further challenges. In real scenes, target objects are usually not completely visible. Without depth information to ease segmentation, such a method must estimate postures from partially occluded and cluttered color data. Moreover, 6D pose estimation applications typically require instant feedback. A pose estimation method should be concise and capable of providing precise poses as soon as possible.

W.-L. Huang, C.-Y. Hung, and I.-C. Lin are with the Institute of Multimedia Engineering, College of Computer Science, National Yang Ming Chiao Tung University (former National Chiao Tung University), Taiwan.

* equal contribution to this work.

† corresponding author, e-mail: ichenlin@cs.nctu.edu.tw.

The published manuscript and supplementary document are available at http://ieeexplore.ieee.org. DOI: 10.1109/TMM.2021.3092149



(a) Input image (b) Segmentation result

(c) Conceptual diagram of predicted 2D points (distributions).

(d) Estimated poses

Fig. 1. Example of 6DOF pose estimation of multiple objects with the proposed method. Given a single input RGB image (a), our network can evaluate two outputs, specifically the multiobject segmentation results presented in (b) and 2D keypoint locations with confidences as illustrated in (c). The 6D poses estimated from the fusion of these outputs are displayed in (d), where white bounding boxes denote the ground-truth poses and the bounding boxes in other colors denote the predicted poses of different classes.

Generally, 6D pose estimation methods predict the projected two-dimensional (2D) locations of predefined 3D keypoints of target objects. These 2D-to-3D correspondences are then used to evaluate the poses with a perspective-*n*-point (PnP) algorithm [5]. Traditionally, these 2D locations are detected according to handcrafted feature extractors, such as SIFT [6] or SURF [7]. However, these handcrafted features are not applicable to weakly textured or textureless objects. Some recently developed 6D pose estimation methods rely on convolutional neural networks (CNNs). Given an input image, a CNN can output the features of objects for object detection [8] or even directly regress the 6D object pose [9]. These methods involve the use of holistic information and achieve remarkable performance. However, when applied to a scene with partially occluded objects, their performance easily degrades because parts of salient features are not visible or confused with

others. A few researchers [10]–[13] have also recognized this key problem and combined multiple predictions from local information to gain more robust poses. However, these researchers have modeled the uncertainty or confidence of the predicted location during training according to predefined rules, such as the proximity of the predicted location to the ground truth.

In this work, we argue that incorporating uncertainty in formulation is helpful for both training and inference. Instead of basing predictions on global information or combining them according to predefined rules, we assume that the accuracy of 6D pose estimation can be improved through the proper integration of multiple predictions of different locations of an object. We propose a confidence-based 6D pose estimation network that can predict the projected 2D locations of 3D keypoints and the corresponding confidence values from each visible patch of an object. After the selection of valid patches from candidates, the predicted 2D keypoints are regarded as several probabilistic distributions and fused together according to the predicted locations and confidence values (which can be transformed into variances).

To realize the aforementioned approach, we derived a novel loss function from the statistical distance between the predicted and ground-truth distributions. The derived loss function contains two competing terms: the first term is a weighted L1 distance that focuses on the most concentrated distributions, and the second term is an unsupervised term for concentrating the predicted distributions. The mutual competition provides the proposed network with an intuitive means of leveraging the location loss and variance loss. Finally, the number of deviating cases can be reduced, and more robust poses can be obtained through the proper integration of the most concentrated distributions. The fused 2D keypoint locations are then input into a PnP algorithm for the 6D pose of each target object to be obtained. We also conducted experiments to analyze the effect of applying two popular 3D keypoint sets, namely the eight corners of the 3D bounding box (BBox) and the farthest-point-sampling (FPS) keypoints in [13]. Our strategy for selecting 3D keypoint sets is described in the Experiments section.

We evaluated the performance of our method on the Occlusion LINEMOD [14] data set, which is one of the most extensively used and challenging benchmarks because the background of each image is cluttered and the target objects frequently occlude each other. Our method introduces the unsupervised confidence into the loss function during training and combines the most reliable local predictions during inference to mitigate deviation caused by occlusion, thereby achieving state-of-the-art performance. In addition to being effective, the proposed method is also feasible for real-time applications. Fig. 1 presents an example of 6D pose estimation with the proposed method.

To summarize, the contributions of our work are as follows:

- A confidence-based distributional 6D pose estimation framework that can be applied to partially occluded images is proposed.
- A novel loss function is derived for our proposed network to apply training for location prediction and confidence prediction through mutual competition. The predicted confidence values can be utilized to integrate the predicted locations.
- We experimentally evaluated two different sets of 3D keypoint locations with our framework and report our strategy for keypoint selection.
- The proposed network achieves state-of-the-art accuracy on the Occlusion LINEMOD dataset and real-time performance.

II. Related Work

This section introduces several representative methods related to 6DOF object pose estimation. We first discuss the pros and cons of classical methods and then present recent deep-learning-based methods. In addition to discussing work related to 6D pose estimation, we also introduce several representative methods for object detection because such methods are typically employed in object pose estimation frameworks to distinguish target objects from their background.

A. Classic methods for object pose estimation

Classic object pose estimation techniques generally fall into two categories: feature-based and template-based approaches. Feature-based approaches [6], [7], [15]–[17] are employed to extract image features, such as shape, texture, and color, which match those in a target image. Such methods can tolerate high levels of background clutter and changes in illumination. However, they cannot be used to estimate the poses of weakly textured or textureless objects because no salient feature is available for extraction. Template-based approaches [18]–[25] produce the templates of objects from different viewpoints and identify the template that matches the target image. These methods can estimate the poses of textureless objects but perform poorly when occlusion is present because they typically compare the entire projected shape of a target object.

B. CNN-based object detection

Well-known two-stage detectors [26]–[31] first select regions of interest from the input image and generate a set of region proposals. They then classify these proposals to identify the target objects. By contrast, one-stage detectors [32]–[39] produce region proposals and classifications simultaneously. Although one-stage detectors are much faster than two-stage detectors are, their prediction accuracy is often worse than that of multistage detectors. Lin et al. [40] proposed RetinaNet and introduced a focal loss to solve the foreground–background class imbalance problem during training. You Only Look Once, version 3 (YOLOv3) [36] is the result of integrating FPN [41] and ResNet [42] into YOLOv2 [34]. It has comparable accuracy to that of RetinaNet but with faster detection. In our work, we employed YOLOv3 as the backbone network because of its efficiency and effectiveness and applied a focal loss function to optimize the detection accuracy.

C. CNN-based object pose estimation

CNN-based pose estimation methods can generally be categorized as pose-regression-based and keypoint-based methods. Pose regression methods [9], [43]–[47] directly estimate the 6D pose of a target object. PoseCNN [43] localizes target objects in an image and predicts their depths to obtain their 3D locations. However, its performance degrades when the input is an RGB image without depth information. Liu *et al*. [9] utilized a triplet network to create a strong correlation between an object's features and 6D pose for regression. To address the nonlinearity rotation space, several methods [44]–[47] have been developed for discretizing the rotation space and transforming the rotation estimation into a classification task. These approaches typically estimate coarse results first and rely on pose refinement methods [48], [49] to obtain precise 6D object poses.

Instead of directly predicting 6D object poses, keypoint-based methods [10]–[13], [50]–[55] use a two-stage strategy. This strategy involves first detecting the 2D keypoints of a target object and then using a PnP algorithm to calculate the 6DOF pose according to 2D-to-3D correspondences. For recent neural network structures, predicting 2D keypoint locations is more straightforward than estimating nonlinear 3D rotations and translations. BB8 [50] identifies different targets through segmentation and regresses 2D keypoints from each segmented region. YOLO-6D [51] employs YOLOv2 [34] architecture for the prediction of object keypoints based on the confidence value at each location of a low-resolution feature map. YOLO-6D defined their confidence as a function of the distance between a predicted point and the ground truth. These methods exhibit impressive performance on the LINEMOD [24] data set but are easily distracted by occlusion and noise due to their heavy reliance on global information.

Several recent approaches have been proposed for overcoming the occlusion problem. Oberweger et al. [11] used an encoder–decoder architecture to obtain keypoint heatmaps, which are then aggregated to serve as the input of a PnP algorithm. They applied the predictor in a sliding window fashion, which is unsuitable for real-time processing. PVNet, developed by Peng et al. [13], estimates pixel-wise unit vectors pointing to the object keypoints and generates 2D keypoint hypotheses according to random-sample-consensus (RANSAC) based voting. An uncertainty-driven PnP algorithm is then used to estimate the object 6D pose according to the spatial distributions, i.e. means and covariances, of keypoints. These spatial distributions are characterized by weighted hypotheses of keypoint locations, where the weight of a hypothesis is higher if it coincides with more predicted directions. Their vector fields can predict the positions of keypoints of an occluded or even truncated target. However, this network is trained individually for each class. Under such conditions, it cannot concurrently detect multiple objects of different classes.

Zakharov et al. [52] adopted the same backbone network as that of PVNet [13] but with the utilization of three decoders. Instead of predicting pixel-wise vector fields, they encoded each target object with UV maps, where a color denotes a 3D location. With these correspondence maps, they turned the regression problem into a classification problem. Song et al. [55] also designed a network based on PVNet. In addition to predicting 2D keypoint locations, they also estimated edge vectors and symmetry correspondences and integrated them into the input of the following refinement modules.

Hu et al. [12] adopted YOLOv3 [36] architecture as a backbone network with two branches, namely a segmentation branch and a regression branch, where the segmentation branch identifies objects of different classes and the regression branch predicts the 2D locations of the projected 3D keypoints. They gathered predictions of each visible object patch according to predicted confidence values, which are defined to reflect the proximity of the predicted 2D locations to the ground truth. The RANSAC EPnP [5] algorithm is then used to evaluate the 6D pose on the basis of predicted 2D keypoints. Hu et al. later proposed a method [56] that directly regresses 6D poses from the predicted correspondences. Our network architecture is derived from their work [12]. However, we regard the 2D keypoint locations as probabilistic distributions, where the confidence can be determined in an unsupervised fashion. Moreover, the proposed confidence can be applied to integrate multiple predictions for EPnP.

III. Method

Our goal is to detect and estimate the 6DOF pose of each target object in real time given a single RGB image. These target objects are rigid, and their 3D models are provided in advance. A 6D pose is defined as a rigid transformation $(R; t)$ from the object space to camera space. R and t represent the 3D rotation and translation, respectively.

Inspired by recent work [12], [13], [52], [55], we adopted a two-stage strategy. An encoder–decoder architecture is adopted to predict the 2D keypoint locations of a predefined 3D target object along with the confidence values, which are subsequently used as the inputs of the inference module for 6D pose estimation. In contrast to related methods, in our method, 2D keypoint locations are regarded as normal distributions for the derivation of our loss function and fusion of multiple predictions according to confidence values. Our proposed framework contains a shared encoder and two separate decoders responsible for distinct tasks. One decoder produces coarse semantic segmentation of the input image, and the other predicts the 2D keypoint locations along with the confidence values of the segmented regions. Fig. 2 depicts the pipeline of our proposed network. In the remainder of this section, we introduce our network architecture and then elaborate on the functions of the two decoder branches. Finally, we describe our inference module.

A. Network architecture

To focus the benefits of our new loss function and inference module, we adopted the same base architecture proposed by Hu et al. [12], which contains a shared encoder with two decoders for different tasks. The encoder is the Darknet-53 of YOLOv3 [36], which is pretrained with COCO data set. The two decoders are feature pyramids [41] based on the results

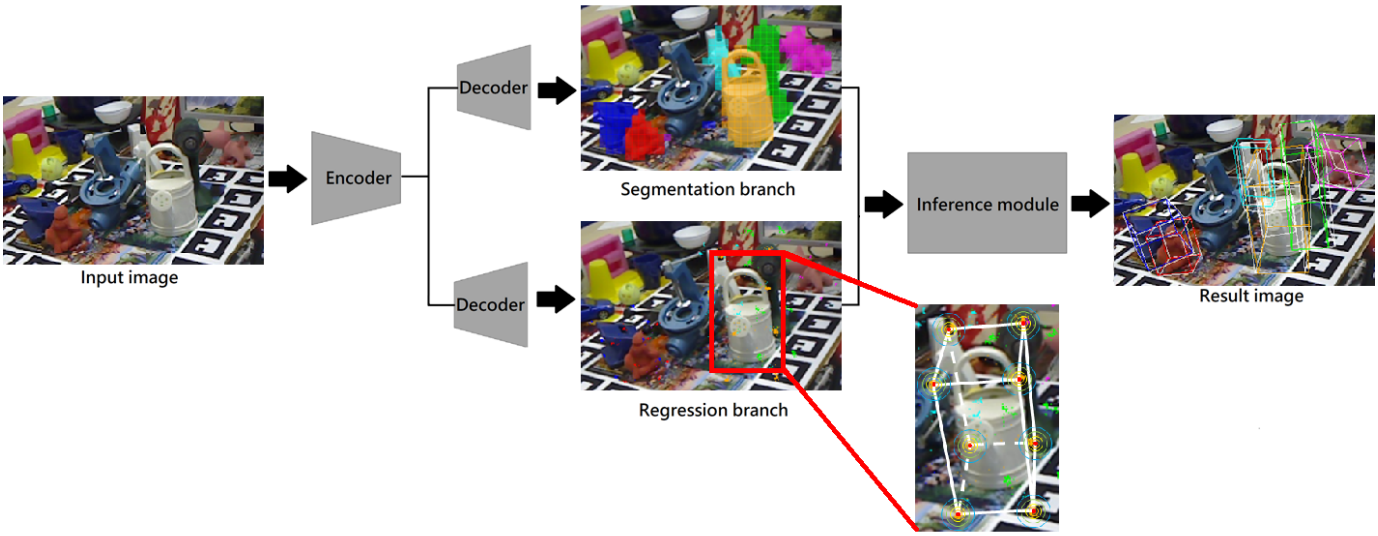


Fig. 2. Overview of our framework. The input image is processed first by an encoder and then by two separate decoders, which produce a coarse segmentation and the 2D keypoint locations with confidences, respectively. The two output tensors are then fed into an inference module, which fuses the predictions belonging to the same class and finally outputs the 6DOF pose of each visible target object.

of the encoder. We adopted an identical architecture for these decoders, except for the channel size of the final convolutional layers. One decoder performs semantic segmentation, and the other performs regression. The output channel sizes of the the segmentation and regression branches are denoted D_{seg} and D_{reg}, respectively. Please refer to the supplementary material for the detailed network structure.

B. Segmentation branch

The input image space is divided into a grid of $H \times W$ patches based on the spatial resolution of the output tensor, where the grid element is referred to as the *cell*. The segmentation branch estimates the probability of every object class for each cell and adopts the class with the highest probability as the predicted label. Fig. 3a depicts the predicted result of the segmentation branch. The size of the output channel of the segmentation branch is $D_{seg} = C+1$, where C is the number of object classes, and one additional channel is used for the background.

During the training stage, we projected the 3D models of target objects onto an input image space according to the given ground-truth poses and camera intrinsic matrix to acquire the segmentation masks. Because most areas of an image have variations in the background and object sizes, we adopted the focal loss function proposed by Lin et al. [40] to resolve the class imbalance problem by dynamically weighting the cross-entropy loss.

We first computed the pixel-wise class frequency f_i in the training dataset and then defined the median frequency weights w_i proposed in [57] as follows:

$$w_i = \frac{median(\{f_i | i = 0, 1 \ldots, C\})}{f_i}, \tag{1}$$

(a) Object segmentation (b) 2D keypoint locations from a foreground cell

Fig. 3. Outputs of our proposed network. For a virtual cell superposed on an image, our network estimates a class label and K offset vectors with confidence values. The vectors point from the center of a cell to K projected keypoint locations. The final projected position of a keypoint is inferred according to the predictions of multiple cells.

where w_i is the median frequency weight of the ith class and $median()$ returns the median of the values in the input set. Finally, we adopted a class-balanced version of the focal loss function as our segmentation loss function:

$$L_{seg} = \sum_{i=0}^{C} -\delta_{ih_{gt}} w_i (1 - p_i)^{\tau} \log(p_i) \tag{2}$$

where $\delta_{ih_{gt}}$ is the Kronecker delta function. The output is 1 when $i = h_{gt}$, which is the ground-truth class; otherwise, it outputs 0. p_i is the predicted probability for class i, and τ is a positive hyperparameter that adjusts for the effect of the modulating factor $(1 - p_i)$.

C. Regression branch

The goal of the regression branch is to predict the projected 2D locations of the predefined K 3D keypoints of each visible target object. The configurations of 3D keypoints can vary as long as they are not coplanar. Following the configurations of state-of-the-art methods [11], [12], [51], we first selected the corners of the model bounding box as the 3D predefined keypoints ($K = 8$ in our case). We compared the results by using two representative configurations of predefined 3D keypoints mentioned in Section V.

For each cell (image patch) that belongs to a target object, the regression branch estimates the 2D coordinates (u, v) of the projected K 2D keypoints along with the confidence values. Inspired by [58], in which the 2D bounding box locations are regarded as distributions, we formulated the 2D keypoint locations as several probabilistic distributions. The output channel size of the regression branch D_{reg} is $3K$, which is composed of $2K$ channels for the 2D coordinates (u, v) and K channels for the confidence values of K projected 2D keypoints.

In the following paragraphs, we provide a step-by-step overview of how the the loss functions for the regression branch are derived. We represent the ground-truth distribution of a keypoint location g_j as a Dirac delta function:

$$P_D(x) = \delta(x - g_j), \tag{3}$$

For an estimated keypoint location e_j, we formulate it as a Gaussian distribution $P_\Theta(x)$ with standard deviation σ_j:

$$P_\Theta(x) = \frac{1}{\sqrt{2\pi{\sigma_j}^2}} e^{-\frac{(x-e_j)^2}{2{\sigma_j}^2}}, \tag{4}$$

where Θ is a set of learnable parameters.

The goal is to determine the $\hat{\Theta}$ value that maximizes the similarity between the $P_\Theta(x)$ and $P_D(x)$ values. We adopt Kullback–Leibler (KL) divergence as the basis of the loss functions of the regression branch. The concept can be formulated as follows:

$$\hat{\Theta} = \underset{\Theta}{argmin} D_{KL}(P_D(x)||P_\Theta(x)). \tag{5}$$

The KL-Divergence loss can be expanded as follows:

$$\begin{aligned}
&D_{KL}(P_D(x)||P_\Theta(x)) \\
&= \int P_D(x) \log P_D(x) dx - \int P_D(x) \log P_\Theta(x) dx \\
&= H(P_D(x)) - \log P_\Theta(g_j) \\
&= H(P_D(x)) - \log(\frac{1}{\sqrt{2\pi{\sigma_j}^2}} e^{-\frac{(g_j-e_j)^2}{2{\sigma_j}^2}}) \\
&= H(P_D(x)) - \log(\frac{1}{\sqrt{2\pi{\sigma_j}^2}}) + \frac{(g_j - e_j)^2}{2{\sigma_j}^2} \\
&= H(P_D(x)) + \frac{(g_j - e_j)^2}{2{\sigma_j}^2} + \frac{\log(2\pi)}{2} + \frac{\log({\sigma_j}^2)}{2},
\end{aligned} \tag{6}$$

where $H(P_D(x))$ is shorthand for $\int P_D(x) \log P_D(x) dx$. Because $H(P_D(x))$ and $\log(2\pi)/2$ do not depend on the value of the parameter set Θ, the formulation is abstracted as follows:

$$D_{KL}(P_D(x)||P_\Theta(x)) \propto \frac{(g_j - e_j)^2}{{\sigma_j}^2} + \log({\sigma_j}^2). \tag{7}$$

Because ${\sigma_j}^2$ is a denominator, it may cause numerical problems during training, such as division by zero. Therefore, we instead make the regression branch predict the *localization confidence value* $c_j = \frac{1}{{\sigma_j}^2}$. The KL divergence loss function can then be rewritten as follows:

$$D_{KL}(P_D(x)||P_\Theta(x)) \propto c_j(g_j - e_j)^2 - \log(c_j). \tag{8}$$

We subsequently split the KL divergence loss function into regression loss L_{reg} and confidence loss L_{conf} functions as follows:

$$\begin{aligned}
L_{reg} &= \sum_{j=1}^{K} c_j(g_j - e_j)^2 \\
L_{conf} &= \sum_{j=1}^{K} -\log(c_j),
\end{aligned} \tag{9}$$

where e_j, c_j, and g_j represent the estimated 2D location, estimated confidence value and 2D ground-truth location of the jth keypoint, respectively. The regression loss L_{reg} is a confidence-weighted squared distance, and the confidence loss L_{conf} is a negative log likelihood.

Instead of directly predicting the absolute 2D keypoint coordinates, we predict the offsets from the center of each foreground cell to the K 2D keypoint locations. Fig. 3b depicts the ground-truth offsets assigned for a foreground cell (patch) of a target object. In practice, for the regression loss L_{reg}, we replace the squared distance with the L1 distance because the L1 distance is less sensitive to outliers. Similar to the segmentation loss, the regression loss and confidence loss face the problem of class imbalance. Therefore, we again apply the median frequency weight to both losses. The class-balanced version of the two losses is formulated as follows:

$$\begin{aligned}
L_{reg} &= \sum_{i=1}^{C} \sum_{j=1}^{K} \delta_{ih_{gt}} w_i c_j \|g_j - e_j\|_1 \\
L_{conf} &= \sum_{i=1}^{C} \sum_{j=1}^{K} -\delta_{ih_{gt}} w_i \log(c_j),
\end{aligned} \tag{10}$$

where $\delta_{ih_{gt}}$ is the Kronecker delta function and w_i is defined in Eq. (1).

D. Total loss

The total loss L_{total} of the whole network is defined as follows:

$$L_{total} = \alpha L_{seg} + \beta L_{reg} + \gamma L_{conf}, \tag{11}$$

where α, β, and γ are the weights to balance the influences of the three losses. L_{reg} and L_{conf} in the regression branch compete with each other. L_{conf} seeks high confidence values, whereas L_{reg} seeks low confidence values to downweight the L1 distance term. The inclusion of these competing terms enables our network to regress the 2D keypoint locations and assign confidence values concurrently. When the network identifies a prediction that should have a higher confidence value, the higher confidence weight causes the prediction to contribute a greater portion of regression loss for quick convergence, and vice versa. The performance of the network

in the inference stage is also facilitated because our method selects only predictions with high locational confidence values for fusion.

E. Inference module

Given an image, our network predicts the object class and 2D locations with the localization confidence values of the K projected keypoints for each foreground cell in the $H \times W$ grid. We present a *confidence-based voting strategy* with a PnP algorithm to fuse information from multiple patches and estimate 6DOF poses.

Our inference module first clusters cells (patches) belonging to the same class as representing one or multiple object instances according to the centers of the 2D keypoint locations. The cell with the highest mean confidence value of keypoints is selected as the leader cell of the corresponding object instance. A cell is inserted in the valid cell pool of the corresponding object only when its center surrounds the center of the leader cell and when it possess confidence values greater than or equal to th.

Given the valid pool, the inference model takes the 2D keypoint locations with their confidence values from the filtered cells and fuses them as follows:

$$e_j^{fused} = \frac{\sum_{k=1}^{N} c_{j,k} e_{j,k}}{\sum_{k=1}^{N} c_{j,k}}, \tag{12}$$

where e_j^{fused} is the fused 2D keypoint location of the jth keypoint. $c_{j,k}$ and $e_{j,k}$ denote the confidence value and 2D keypoint location, respectively, of the k^{th} filtered prediction of the j^{th} keypoint. N is the candidate number that we select for properly fusing the predictions, as discussed later in Section V.

The aggregated 2D keypoint locations and their corresponding 3D keypoints then form the 2D-to-3D correspondences for the input of the PnP algorithm to evaluate the 6 pose. The confidence-based voting strategy is adopted to fuse the predictions of each 2D keypoint location by weighting them according to their confidence values. Because a more accurate prediction corresponds to a higher confidence value and the formulation conforms to spatially probabilistic distribution, we can obtain a more precise estimation of 2D keypoint locations through this strategy. Furthermore, this strategy is more efficient than RANSAC-based methods because it does not apply an iterative procedure to determination of the optimal 2D-to-3D correspondence set.

IV. Implementation Details

Because we selected YOLOv3 [36] as the backbone network, input images were first reshaped into a resolution of 608×608 pixels. The segmentation branch outputs a $76 \times 76 \times (C+1)$ tensor, where C is the number of object classes and one channel is used for the background. The softmax function is applied as the activation function of our segmentation branch. The regression branch outputs a $76 \times 76 \times 3K$ tensor, where K denotes the number of predefined 3D keypoints. The first K channels are the predicted confidence values of the K 2D keypoint locations, and the remaining $2K$ channels are the 2D coordinates of K projected 3D keypoints. We adopted a sigmoid function as the activation function to normalize the confidence values.

However, in the original definition $c_j = \frac{1}{{\sigma_j}^2}$, the range of the confidence value c_j is $[0, \infty]$, which differs from the output range of the normalized values, which is $[0, 1]$. Therefore, we designed a mapping function f for mapping the range of the normalized confidence values to $[0, \infty]$. f is defined as

$$f(x) = \log(\frac{1}{1-x}), 0 \leq x \leq 1. \tag{13}$$

Then, c_j can be expressed as:

$$c_j = f(c_{norm}), \tag{14}$$

where c_{norm} is the normalized confidence value. c_{norm} is an internal variable designed for the network output. It does not influence the original loss functions in (10). We also normalized the output 2D coordinates to the range $[0, 10]$, as in [12].

Our encoder was pretrained on the COCO data set [59], and the two separate decoders were initialized with values sampled from normal distributions. For the convolutional layers, which are followed by a batch normalization layer, we set the mean and standard deviation of the normal distribution to 0 and 0.01, respectively. For other convolutional layers, we applied the same mean and standard deviation but set their bias to 0. For the batch normalization layers, we selected 1 and 0.01 as the mean and standard deviation of the normal distribution, respectively, and set their bias to 0.

We trained our network for 300 epochs with the training data sets described in SectionV. The initial learning rate was set to 0.001, and it was divided by 10 after 50%, 75%, and 90% of the total number of epochs. We adopted the stochastic gradient descent (SGD) as our optimizer, with a momentum of 0.9 and weight decay of 0.0005. With the batch size set to 18, we trained our network on two 2080 Ti GPUs in parallel and tested them on a 2080 Ti GPU. The frequency weights w_i in (1) were calculated for the training data set. In all experiments, the hyperparameter τ in (2) was set to 2, and the weights α, β, and γ used for the total loss function were 64, 8, and 1, respectively.

V. Experiments

A. Training data preparation

Most methods developed for 6DOF pose estimation of partially occluded objects have been tested on the Occlusion LINEMOD data set [14]. However, recently developed state-of-the-art methods have adopted two approaches to collecting training data. For a fair comparison, we prepared and applied two training data sets according to these two different approaches.

Existing studies have mostly extracted training data from the LINEMOD data set [24]. To synthesize the training data from the LINEMOD data set, we applied the cut-and-paste method proposed in [60]. We segmented the target objects in the training set of the LINEMOD data set with the provided ground-truth poses and masks and pasted them onto the random background sampled from the SUN397 data set [61].

These objects were pasted at random locations, orientations, and scales in an arbitrary order to increase the diversity of the data set, including that of the occlusion situations. We generated the training data set only from 8 out of the 13 objects in the LINEMOD data set because only these 8 objects appeared in the Occlusion LINEMOD data set [14]. Finally, we generated 20000 samples for training.

To prevent overfitting, we applied online data augmentation techniques that included random cropping, resizing, rotation, blurring, and color jittering during training. To simulate occlusion situations and improve the robustness of our framework, we further applied the random erasing method proposed in [62]. For all object instances in the synthetic images, a proportion of their 2D bounding boxes were filled with random noise.

Unlike most of its counterparts, the original HybridPose [55] paper took a fraction of the Occlusion LineMOD data set as the training data set, whereas the conventional setting was adopted for the updated version of this method. Therefore, in comparisons with HybridPose, we listed both sets of results.

B. Evaluation metrics

We evaluated our predicted 6DOF poses with ADD(-S) [24], [43]. Given an estimated pose and the corresponding ground-truth pose, ADD projects the 3D model of the target object onto the camera space for each of the two poses and computes the mean distance between the two projected point sets. For symmetrical objects, we applied ADD-S to calculate the mean distance on the basis of the closest point distance. The estimated pose was considered correct if the ADD(-S) distance between the point sets was less than 10% of the diameter of the model (0.1d).

C. Ablation studies

1) Derived KL-divergence loss: To analyze the effectiveness of our derived KL divergence loss, we re-implemented segmentation-driven 6D object pose estimation (SegDriven) [12] and fine-tuned this network with our training data set for comparison. For our system, we replaced our inference module with the RANSAC EPnP algorithm [5], which is the inference strategy used in SegDriven. With this configuration, the only difference between the two frameworks is the loss function used. In SegDriven, the regression loss is an L1 distance without confidence-based reweighting and the confidence loss is in a supervised form, where the ground-truth confidence value is determined on the basis of the current regression loss.

For a fair comparison, we selected completely identical training settings. Table I presents the results of an ADD(-S) comparison between our method with the RANSAC EPnP inference module and the re-implemented version of SegDriven. In this analysis, our model outperformed SegDriven by 1.43%. The reason for this result is that we regarded the predicted 2D coordinates as distributions to formulate our loss function. With the confidence-based reweighting in our regression loss function, the predictions with higher confidence values contributed a larger proportion of the loss. Because the confidence values are learned in an unsupervised fashion, our training process can focus more on improving predictions according to the effective (high-confidence-value) regions; in other words, the predictions have more opportunities to converge.

TABLE I
COMPARISON BETWEEN THE RE-IMPLEMENTED VERSION OF SEGDRIVEN [12] AND OUR METHOD WITH RANSAC-BASED INFERENCE MODULE (R.-INF.) IN TERMS OF ADD(-S)-0.1D ON THE OCCLUSION LINEMOD DATASET.(*: SYMMETRIC OBJECTS)

	Re-imp. SegDriven	Ours with R.-inf.
Ape	**19.32**	15.30
Can	65.53	**65.87**
Cat	17.10	**19.46**
Driller	65.40	**66.56**
Duck	28.08	**30.27**
Eggbox*	25.70	**33.62**
Glue*	**44.30**	40.75
Holepuncher	50.00	**55.04**
Average	39.43	**40.86**

TABLE II
COMPARISON BETWEEN RANSAC EPNP INFERENCE [5] (R.-INF.) AND CONFIDENCE-BASED VOTING WITH PNP (C.-INF.) WITH OUR MODEL OUTPUT. THE EXPERIMENT WAS PERFORMED ON THE OCCLUSION LINEMOD DATA SET IN TERMS OF ADD(-S)-0.1D . (*: SYMMETRICAL OBJECTS)

	Ours with R.-inf.	Ours with C.-inf. (Proposed)
Ape	**15.30**	14.36
Can	65.87	**68.35**
Cat	**19.46**	18.20
Driller	66.56	**66.89**
Duck	30.27	**31.50**
Eggbox*	33.62	**34.38**
Glue*	**40.75**	40.64
Holepuncher	55.04	**55.54**
Average	40.86	**41.23**

2) Inference strategies: After performing the aforementioned experiment, we compared the effect of including RANSAC EPnP [5] and the proposed *confidence-based voting* with PnP in the inference module. The results are presented in Table II in terms of ADD(-S). Our network with the proposed confidence-based voting inference module exhibited improved performance. Although the accuracy for estimating the poses of the majority of object classes was improved, we observed that the accuracy was lower for small objects (*Ape, Cat, Glue*). We believe that this result is attributable to the insufficient number of predictions. These small objects occupy fewer cells (image patches) than their larger counterparts. With fewer predictions, incorrect predictions with imperfect confidence values are more likely to affect the fused results. Moreover, when occluded by other objects, small target objects typically lose a greater proportion of image patches than do larger objects. These factors increased the error rate of our estimated confidence values. Although RANSAC EPnP does not involve the use of confidence values for fusion, thus avoiding this problem, it does not possess the advantages of fusion from multiple predictions.

TABLE III
COMPARISONS OF DIFFERENT THRESHOLD VALUES USED IN THE CONFIDENCE-VOTING STRATEGY IN TERMS OF ADD(-S)-0.1D ON THE OCCLUSION LINEMOD VALIDATION DATASET. WE USED 10% OF THE OCCLUSION LINEMOD DATA AS THE VALIDATION DATASET. (*: SYMMETRIC OBJECTS)

Threshold th_{norm}	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Ape	**6.90**	**6.90**	**6.90**	**6.90**	6.03	6.03	6.03	6.03	6.03
Can	69.42	70.25	69.42	70.25	70.25	70.25	69.42	**71.07**	69.42
Cat	**17.24**	**17.24**	**17.24**	**17.24**	**17.24**	**17.24**	16.38	15.52	14.66
Driller	59.50	60.33	**61.98**	61.16	61.16	61.16	**61.98**	61.16	**61.98**
Duck	29.57	29.57	29.57	29.57	29.57	**30.43**	29.57	29.57	29.57
Eggbox*	39.32	**40.17**	**40.17**	**40.17**	39.32	39.32	**40.17**	37.61	36.75
Glue*	38.89	38.89	**40.00**	37.78	37.78	37.78	37.78	36.67	34.44
Holepuncher	59.17	60.00	60.00	60.00	**60.83**	**60.83**	**60.83**	**60.83**	60.00
Average	40.00	40.42	**40.66**	40.38	40.27	40.38	40.27	39.81	39.11

TABLE IV
COMPARISONS OF THE RESULTS OF USING DIFFERENT CANDIDATE NUMBERS USED IN THE CONFIDENCE-VOTING STRATEGY IN TERMS OF ADD(-S)-0.1D ON THE OCCLUSION LINEMOD VALIDATION DATASET. WE USED 10% OF THE OCCLUSION LINEMOD DATA AS THE VALIDATION DATASET. (*: SYMMETRIC OBJECTS)

Candidate Number	1	5	7	9	11	14	16
Ape	**10.34**	6.90	7.76	8.62	7.76	7.76	6.90
Can	69.42	**70.25**	69.42	67.77	68.60	67.77	67.77
Cat	12.93	14.66	14.66	15.52	15.52	**16.38**	**16.38**
Driller	63.64	**64.46**	63.64	62.81	63.64	63.64	62.81
Duck	26.96	26.09	**27.83**	**27.83**	**27.83**	**27.83**	**27.83**
Eggbox*	**43.59**	42.74	**43.59**	**43.59**	42.74	42.74	42.74
Glue*	38.89	**41.11**	38.89	**41.11**	**41.11**	40.00	40.00
Holepuncher	**60.83**	60.00	60.00	60.00	60.00	60.00	60.00
Average	40.83	40.77	40.72	**40.91**	40.90	40.76	40.55

3) Confidence threshold: As mentioned in Section III, we set a threshold th for the predicted confidence values to better integrate the predicted 2D keypoint locations. Any prediction with a confidence value lower than th is discarded before entering the inference module. For simplicity, we directly applied the threshold to the normalized confidence value c_{norm} instead of using the formal confidence value c_j, which has a range of $[0, \infty]$. Such a range renders the threshold difficult to define. The results of using different th_{norm} values are presented in Table III. These results indicate that discarding predictions with low confidence values improves the accuracy of 6D pose estimation. We selected 0.3 as the confidence threshold for our framework.

4) Candidate number: To further optimize the integration of the predicted 2D locations, we analyzed the influence of the candidate number N used in the inference module. When N is set to 1, only the valid prediction with the highest confidence value is selected for inference. Table IV presents the results of applying different candidate numbers in terms of ADD(-S). The results indicate that fusing a limited number of top predictions can further improve the accuracy of pose estimation. Moreover, aggregating fewer predictions reduces the inference time. We selected nine as the candidate number for our framework.

5) Keypoint location selection: The methods developed in [11], [12], [51] employ the eight corners of the model bounding box (BBox) as the 3D predefined keypoints, whereas those developed in [13], [55] employ keypoints extracted with the farthest point sampling (FPS) algorithm, which repeatedly includes a new surface keypoint that is farthest from the current keypoint set. To investigate the influence of keypoint sets, we applied BBox keypoints and FPS keypoints in our model. For each keypoint set, we trained a single model for multiple object classes. As indicated in the upper part of Table V, the overall accuracy was higher when BBox keypoints were used than when FPS keypoints were used. However, the two models exhibited similar performance for *Eggbox*, and the model with FPS keypoints outperformed the model with BBox keypoints for *Duck* and *Glue*.

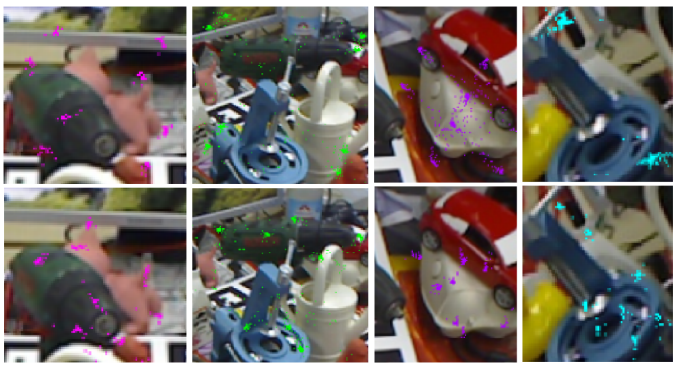


Fig. 4. Predicted 2D keypoint locations by our models (multi-class) for *Cat*, *Driller* and symmetric objects *Eggbox*, *Glue*. (The upper row) results with the BBox keypoint set; (the lower row) results with the FPS keypoint set.

To avoid interclass interference during training, we further trained one model for each of these three classes with each of the two keypoint sets. As indicated in the lower part of Table V, the models with each of the keypoint sets exhibited

TABLE V
COMPARISONS OF APPLYING BOUNDING BOX (BBOX) KEYPOINTS AND FARTHEST POINT SAMPLING (FPS) KEYPOINTS IN OUR MODEL WITH CONFIDENCE-VOTING WITH PNP. THE EXPERIMENT WAS TESTED ON THE OCCLUSION LINEMOD DATASET IN TERMS OF ADD(-S)-0.1D . (*: SYMMETRIC OBJECTS)

Our model (multi-class)	BBox keypoints	FPS keypoints
Ape	**14.36**	11.71
Can	**68.35**	60.98
Cat	**18.20**	16.85
Driller	**66.89**	63.18
Duck	31.50	**33.42**
Eggbox*	**34.38**	33.36
Glue*	40.64	**43.19**
Holepuncher	**55.54**	48.93
Average	**41.23**	38.95

Our model (ind.-class)	BBox keypoints	FPS keypoints
Duck	35.70	35.96
Eggbox*	35.83	**41.70**
Glue*	41.09	**46.95**

similar performance to that of *Duck*, and the models with the FPS keypoint set outperformed the models with the BBox keypoint set for symmetrical objects *Eggbox* and *Glue*.

The examples presented in Fig. 4 may offer an explanation for the experimental results listed in Table V. For common (asymmetrical) objects, such as *Cat* and *Driller* in Fig. 4, the BBox keypoint predicted locations were more concentrated than the FPS keypoint predicted locations. Moreover, the spans between BBox keypoints were greater and more orthogonal. Such circumstances are beneficial for the application of the PnP algorithm. However, for symmetrical objects, *Eggbox* and *Glue*, the use of BBox keypoint sets sometimes resulted in symmetrical ambiguity when objects were seriously occluded; using FPS keypoint sets might alleviate such ambiguity during training.

Therefore, our strategy to select keypoints is as follows:

- When a single prediction model is applied to multiple classes, the object BBox should be adopted as the keypoints.
- When a prediction model is applied to an individual class, BBox keypoints should be adopted for common objects and FPS keypoints should be adopted for symmetrical objects.

D. Comparison with modern approaches

We compared our approach with state-of-the-art methods in which a single RGB image is used as the input, including *SingleStage*(+*SegDriven*) [12], [56], *Pix2Pose* [53], *SingleStage*(+*PVNet*) [13], [56], and *HybridPose* [55]. SegDriven [12] and PVNet [13] perform pose estimation by integrating predictions from valid patches and the pixels of each visible target object, respectively. SingleStage [56] employs a small network for the evaluation of 6D poses according to the 2D-to-3D correspondences obtained from the aforementioned two methods. Pix2Pose [53] estimates the 3D coordinates and expected errors of valid pixels for input into an RANSAC PnP algorithm. HybridPose [55] predicts keypoints, edge vectors, and symmetrical correspondences for pose estimation.

Table VI summarizes the results of comparing the proposed method with the methods developed in [53], [55], [56], tested on the Occlusion LINEMOD data set. For each class, the highest accuracy value is presented in bold and the second highest accuracy value is underlined. When a single model was trained for multiple classes, the proposed *Ours*(*multi-class*) model outperformed Pix2Pose and SingleStage(+Segdriven) in terms of the overall ADD(-S), but its accuracy was slightly lower than that of SingleStage(+PVNet) and HybridPose. To our knowledge, PVNet involves training an individual model for every class. This configuration can help the training process focus on the visible characteristics in each object class and avoid interclass interference. Therefore, we also trained the proposed framework with such a configuration, named *Ours*(*ind.-class*). When preparing the models for individual classes, we followed the keypoint selection strategy detailed in the V-C5 subsection. BBox corners were used as the 3D keypoints of common objects, and FPS keypoints were used for symmetrical objects.

Regarding the individual classes in Table VI, the accuracy of our model was lower than that of Pix2Pose [53] for the small object *Ape*, SingleStage(+PVNet) [56] and HybridPose [55] for the symmetrical objects *Eggbox* and *Glue*. We believe that this result may be attributable to the grid structure of our network output. The grid structure efficiently provides sparse predictions, but the number of ambiguous cases may be higher when the the visible regions are small. Nevertheless, our method trained for multiple classes, *Ours*(*multi-class*), is capable of estimating the 6D poses of multiple objects at simultaneously, and the average accuracy of this model is comparable to that of other models. Moreover, our model trained for individual classes, *Ours*(*ind.-class*), achieved state-of-the-art accuracy.

Table VII summarizes the results of comparing the computational performance of the proposed method with that of other methods [12], [53], [55], [56] in terms of speed in frames per second. Although the reported speeds of SingeStage(+PVNet) and HybridPose are sufficient for real-time pose estimation, as far as we know, they were measured under single-class circumstances.

The proposed method can estimate object poses for multiple classes simultaneously while retaining reasonable accuracy. On a computer with an Intel i7-5930k CPU, Nvidia RTX 2080 Ti graphics card, and 32 GB of memory, our system requires approximately 27 ms for forward propagation and 7 ms for the inference module. This means that the proposed method is suitable for modern multimedia applications. The performance could be further improved through the use of more advanced backbone networks, such as [37]. Fig. 5 illustrates the pose prediction results obtained with our method.

VI. CONCLUSION

This paper presents a confidence-based distributional framework that can efficiently estimate the 6DOF poses of multiple objects from a single RGB image in real time or accurately estimate objects in individual classes. By regarding 2D keypoint

TABLE VI
COMPARISON OF THE PROPOSED METHOD WITH STATE-OF-THE-ART METHODS IN TERMS OF ADD(-S)-0.1D, TESTED ON THE OCCLUSION LINEMOD DATASET. FOR OURS(IND.-CLASS), WE TRAINED AN INDIVIDUAL MODEL FOR EACH OF THE EIGHT OBJECTS. (*: SYMMETRIC OBJECTS)

	Trained on LINEMOD						Trained on Occlusion LINEMOD	
	SingleStage (+SegDriven) [56]	Pix2Pose [53]	SingleStage (+PVNet) [56]	HybridPose (update) [55]	Ours (multi-class)	Ours (ind.-class)	HybridPose [55]	Ours
Ape	14.80	**22.00**	19.20	20.9	14.36	18.03	53.30	**70.09**
Can	45.50	44.70	65.10	75.3	68.35	**86.41**	86.50	**94.21**
Cat	12.10	22.70	18.90	24.9	18.20	**27.38**	**73.40**	68.64
Driller	54.60	44.70	69.00	70.2	66.89	**77.27**	92.80	**97.94**
Duck	18.30	15.00	25.30	27.9	31.50	**35.70**	62.80	**72.25**
Eggbox*	30.20	25.20	52.00	**52.4**	34.38	41.70	**95.30**	92.74
Glue*	45.80	32.40	51.40	**53.8**	40.64	46.95	**92.50**	90.16
Holepuncher	37.40	49.50	45.60	54.2	55.54	**58.26**	76.70	**90.95**
Average	32.30	32.00	43.31	47.5	41.23	**48.96**	79.20	**84.62**

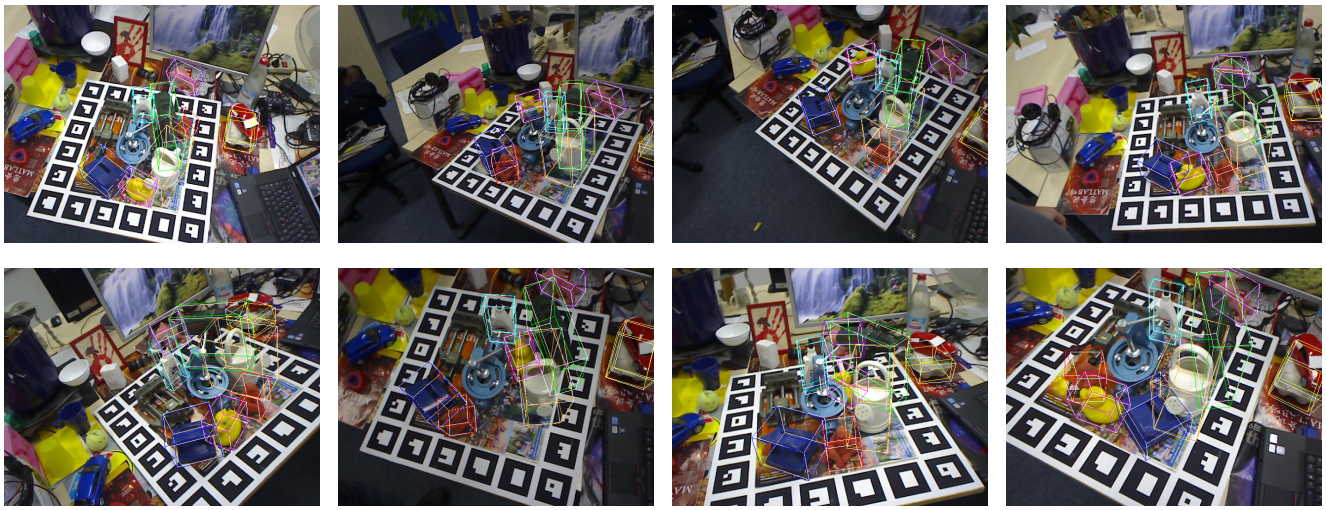


Fig. 5. Qualitative results of our method on the Occlusion LINEMOD dataset. White bounding boxes depict ground-truth poses and the bounding boxes in other colors depict our predicted poses of different classes.

TABLE VII
COMPARISON OF THE PROPOSED METHOD WITH STATE-OF-THE-ART METHODS IN TERMS OF FRAMES PER SECOND (FPS). THE OBJECTS COLUMN INDICATES THAT THE REPORTED FPS IS FOR A MODEL TRAINED FOR A SINGLE OBJECT OR FOR MULTIPLE OBJECTS (MULTIPLE CLASSES CONCURRENTLY).

	FPS	Objects
SegDriven [12]	22	multiple
Pix2Pose [53]	8-10	single
SingleStage(+PVNet) [56]	45	single
HybridPose [55]	30	single
Ours	29	multiple

locations as probabilistic distributions, we formulated a novel loss function that includes the regression loss and unsupervised confidence loss. The competitive losses facilitate training and improve the performance of our method. Furthermore, our inference module can properly aggregate 2D coordinate predictions based on the predicted confidence values. We evaluated the performance of our method on the Occlusion LINEMOD data set, and it achieved state-of-the-art accuracy. The results demonstrate that our method is robust to partial occlusion and suitable for modern multimedia applications.

ACKNOWLEDGMENT

The authors would like to thank Yu-Lun Wang and other CAIG lab members for their assistance in experiments. This work was partially supported by the Ministry of Science and Technology, Taiwan under grant no. MOST 109-2221-E-009-122-MY3.

REFERENCES

[1] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, 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.

[2] 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 *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2637–2646.

[3] C. Wang, R. Martin-Martin, D. Xu, J. Lv, C. Lu, L. Fei-Fei, S. Savarese, and Y. Zhu, "6-pack: Category-level 6d pose tracker with anchor-based keypoints," *arXiv preprint arXiv:1910.10750*, 2019.

[4] G. Zhou, Y. Yan, D. Wang, and Q. Chen, "A novel depth and color feature fusion framework for 6d object pose estimation," *IEEE Transactions on Multimedia*, vol. 23, pp. 1630–1639, 2020.

[5] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem," *International Journal of Computer Vision*, vol. 81, no. 2, p. 155, 2009.

[6] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[7] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," *European Conference on Computer Vision*, vol. 3951, pp. 404–417, 2006.

[8] K. Fu, Q. Zhao, and I. Y.-H. Gu, "Refinet: A deep segmentation assisted refinement network for salient object detection," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 457–469, 2019.

[9] Y. Liu, L. Zhou, H. Zong, X. Gong, Q. Wu, L. Liang, and J. Wang, "Regression-based three-dimensional pose estimation for texture-less objects," *IEEE Transactions on Multimedia*, vol. 21, no. 11, pp. 2776–2789, 2019.

[10] O. H. Jafari, S. K. Mustikovela, K. Pertsch, E. Brachmann, and C. Rother, "ipose: instance-aware 6d pose estimation of partly occluded objects," in *Asian Conference on Computer Vision*. Springer, 2018, pp. 477–492.

[11] M. Oberweger, M. Rad, and V. Lepetit, "Making deep heatmaps robust to partial occlusions for 3d object pose estimation," in *European Conference on Computer Vision*, 2018, pp. 119–134.

[12] Y. Hu, J. Hugonot, P. Fua, and M. Salzmann, "Segmentation-driven 6d object pose estimation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3385–3394.

[13] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "Pvnet: Pixel-wise voting network for 6dof pose estimation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4561–4570.

[14] 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.

[15] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg, "Pose tracking from natural features on mobile phones," in *IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE Computer Society, 2008, pp. 125–134.

[16] V. Lepetit, P. Fua *et al.*, "Monocular model-based 3d tracking of rigid objects: A survey," *Foundations and Trends® in Computer Graphics and Vision*, vol. 1, no. 1, pp. 1–89, 2005.

[17] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce, "3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints," *International Journal of Computer Vision*, vol. 66, no. 3, pp. 231–259, 2006.

[18] M. Stark, M. Goesele, and B. Schiele, "Back to the future: Learning shape models from 3d cad data," in *British Machine Vision Conference*, 2010, pp. 1–11.

[19] J. Liebelt and C. Schmid, "Multi-view object class detection with a 3d geometric model," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 1688–1695.

[20] S. Hinterstoisser, V. Lepetit, S. Ilic, P. Fua, and N. Navab, "Dominant orientation templates for real-time detection of texture-less objects," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2257–2264.

[21] N. Payet and S. Todorovic, "From contours to 3d object detection and pose estimation," in *International Conference on Computer Vision*, 2011, pp. 983–990.

[22] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *International Conference on Computer Vision*. IEEE, 2011, pp. 858–865.

[23] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, "Gradient response maps for real-time detection of textureless objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 5, pp. 876–888, 2012.

[24] 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.

[25] L.-C. Wu, I.-C. Lin, and M.-H. Tsai, "Augmented reality instruction for object assembly based on markerless tracking," in *ACM Symposium on Interactive 3D Graphics and Games*, Febuary 2016, pp. 95–102.

[26] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[27] R. Girshick, "Fast r-cnn," in *IEEE International Conference on Computer Vision*, 2015, pp. 1440–1448.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[29] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 91–99.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[31] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *IEEE International Conference on Computer Vision*, 2017, pp. 2961–2969.

[32] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[33] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European Conference on Computer Vision*. Springer, 2016, pp. 21–37.

[34] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7263–7271.

[35] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "Dssd: Deconvolutional single shot detector," *arXiv preprint arXiv:1701.06659*, 2017.

[36] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[37] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[38] L. Zhang, Y. Gao, Y. Xia, K. Lu, J. Shen, and R. Ji, "Representative discovery of structure cues for weakly-supervised image segmentation," *IEEE Transactions on Multimedia*, vol. 16, no. 2, pp. 470–479, 2014.

[39] Y. Li, Y. Guo, J. Guo, Z. Ma, X. Kong, and Q. Liu, "Joint crf and locality-consistent dictionary learning for semantic segmentation," *IEEE Transactions on Multimedia*, vol. 21, no. 4, pp. 875–886, 2019.

[40] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *IEEE International Conference on Computer Vision*, 2017, pp. 2980–2988.

[41] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2117–2125.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," pp. 770–778, 2016.

[43] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *arXiv preprint arXiv:1711.00199*, 2017.

[44] S. Tulsiani and J. Malik, "Viewpoints and keypoints," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1510–1519.

[45] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, "Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views," in *IEEE International Conference on Computer Vision*, 2015, pp. 2686–2694.

[46] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3d orientation learning for 6d object detection from rgb images," in *European Conference on Computer Vision*, 2018, pp. 699–715.

[47] 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.

[48] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "Deepim: Deep iterative matching for 6d pose estimation," in *European Conference on Computer Vision*, 2018, pp. 683–698.

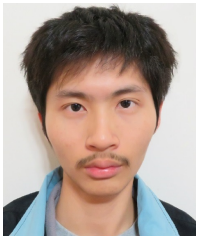
[49] F. Manhardt, W. Kehl, N. Navab, and F. Tombari, "Deep model-based 6d pose refinement in rgb," in *European Conference on Computer Vision*, 2018, pp. 800–815.

[50] 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.

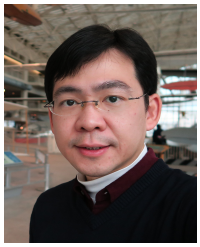
[51] B. Tekin, S. N. Sinha, and P. Fua, "Real-time seamless single shot 6d object pose prediction," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 292–301.

[52] S. Zakharov, I. Shugurov, and S. Ilic, "Dpod: 6d pose object detector and refiner," in *IEEE Conference on Computer Vision*, 2019, pp. 1941–1950.
[53] K. Park, T. Patten, and M. Vincze, "Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation," in *IEEE Conference on Computer Vision*, 2019, pp. 7668–7677.
[54] Z. Li, G. Wang, and X. Ji, "Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation," in *IEEE Conference on Computer Vision*, 2019, pp. 7678–7687.
[55] C. Song, J. Song, and Q. Huang, "Hybridpose: 6d object pose estimation under hybrid representations," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 431–440.
[56] Y. Hu, P. Fua, W. Wang, and M. Salzmann, "Single-stage 6d object pose estimation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2930–2939.
[57] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *IEEE International Conference on Computer Vision*, 2015, pp. 2650–2658.
[58] Y. He, C. Zhu, J. Wang, M. Savvides, and X. Zhang, "Bounding box regression with uncertainty for accurate object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2888–2897.
[59] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision*. Springer, 2014, pp. 740–755.
[60] D. Dwibedi, I. Misra, and M. Hebert, "Cut, paste and learn: Surprisingly easy synthesis for instance detection," in *IEEE International Conference on Computer Vision*, 2017, pp. 1301–1310.
[61] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 3485–3492.
[62] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," *arXiv preprint arXiv:1708.04896*, 2017.

Wei-Lun Huang received B.S. and M.S. degrees in computer science and multimedia engineering from National Chiao Tung University in 2017 and 2019, respectively. His research interests include multimedia content analysis, deep neural networks and computer vision.



Chun-Yi Hung received B.S. and M.S. degrees in computer science and multimedia engineering from National Chiao Tung University in 2018 and 2020, respectively. His research interests include computer vision, machine learning, and object pose estimation.



I-Chen Lin received B.S. and Ph.D. degrees in computer science from National Taiwan University, in 1998 and 2003, respectively. In 2005, he joined Dept. of Computer Science and Inst. of Multimedia Engineering, National Chiao Tung University. He is currently an associate professor in National Yang Ming Chiao Tung University. His research interests include computer graphics, computer vision, and interactive multimedia systems. He is a member of IEEE.