An Evaluation Review on 6D Pose Estimation Benchmark Dataset with RGB-based Appoaches

I-Chien Lai
Department of Electrical Engineering
Stanford University
lai1998@stanford.edu

Pei-Wei Kao
Department of Electrical Engineering
Stanford University
pwkao@stanford.edu

Annie Ho
Department of Electrical Engineering
Stanford University
annieho@stanford.edu

Abstract

In this paper, we evaluate three state-of-the-art approaches for 6D pose estimation. The three models are Pixel-wise Voting Network [8], EfficientPose[2] and Augmented Autoencoder[12]. Our experiment covers testing their provided weight on LineMOD[5] and Occlusion LineMOD[1]. We evaluate the performance of three models with the ADD(-S) [6, 17] metrics. Also, we test the robustness against noise on the three model by applying different amount of noise. Finally, we compare their runtime on Nvidia Tesla K80 machine. With the experiments, we provide a clearer view of three models on accuracy, robustness and speed aspects. In the conclusion section, we further compare the pros and cons between three model type and discuss the limitations for models.

1. Introduction

6D pose estimation is an important topic in computer vision. Tasks such as autonomous driving, robot manipulation and augmented reality relies on accurate and robust pose estimation.

Traditional methods like [6] estimate pose by finding and matching 3D model to image. Recent studies like [17, 15] uses RGB-D as input data achieves satisfying results with novel deep learning methods. However, depth cameras may be more expensive that depth information might not always be available in most camera devices. Thus, there are many studies that solve the task with only RGB data.

6D pose estimation would be difficult in many condition. Textureless object would make feature mapping methods hard to locate points. Symmetric object might mislead the models to predicting an opposite pose. Different scenes that varies in brightness, object reflection or cluttered background are important aspects that hugely impact the performance we need to address. Runtime should also be considered as an important key for future realtime applications.

In this paper, we choose three RGB-based models and evaluate their performance. The three models are Pixel-wise Voting Network (PVNet)[8], EfficientPose[2] and Augmented Autoencoder (AAE)[12]. These three models covers different kind of learning methods, where PVNet and EfficientPose are supervised learning and AAE is self-supervised learning. PVNet and AAE are two-stage detection while EfficientPose is one-stage methods. We want to know the limitations and advantages using these entirely different type of models. Thus, we evaluate the models with ADD(-S) metrics proposed by [6, 17] on LineMOD[5] and Occlusion LineMOD[1] to compare the accuracy. We will also evaluate the performance on noisy input since robustness on different conditions are unavoidable in most application. Finally, we will also compare the runtime for each models on the same machine.

In the approach section, we will first briefly introduce the three methods and their main contributions. Then, we will explain the experiments and show the results that we did on those models. In the conclusion section, we will conclude the results and discuss about the pros and cons for each models.

2. Related Work

On the research field of 6D pose estimation, the deep learning based approaches can mostly be assigned to one of the following two categories: one-stage detection or two-stage detection.

2.1. One-stage detection

The methods in this category directly estimate the 6D pose from end to end. PoseCNN[17] follows this strategy to estimate the 3D translation of an object by localizing its center in the image and predict its distance from the camera. The 3D rotation of the object is estimated by regressing to a quaternion representation. They also introduce a novel loss function that enables PoseCNN to handle symmetric objects.

EfficientPose[2] is also a method that falls into this category. They base their work on an 2D object detection approach and extend it with the ability to also predict the 6D pose of objects in an intuitive and efficient way to maintain the advantages of the base network and to keep the additional computational costs low performing 6D objecct pose estimation of multiple objects. The runtime of their method is nealy independent from the number of objects which makes it suitable for real world scenarios.

2.2. Two-stage detection

In another category, they first detect 2D targets of the object in the given image and subsequently solve a Perspective-n-Point problem for their 6D pose. We can go a step further and divide this method into two category: keypoint-based and dense 2D-3D correspondence methods. PVNet[8] uses pixel-wise voting for keypoint localization and the uncertainty-driven PnP for final pose estimation, and shows that predicting the vector fields followed by RANSAC-based voting for keypoint localization gained a superior performance.

The dense 2D-3D correspondence methods predict the corresponding 3D model point for each 2D pixel of the object. DROD[11] estimates dense multi-class 2D-3D correspondence maps between an input image and available 3D mod- els. Given the correspondences, a 6DoF pose is computed via PnP and RANSAC. An additional RGB pose refinement of the initial pose estimates is performed using a custom deep learning-based refinement scheme.

Another two-stage method is SSD (Single Shot MultiBox Detector)[7]+ AAE (Augmented Autoencoders)[12]. SSD discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. AAE is a real-time pipeline based on a variant of the denoising autoencoder that is trained on simulated views of a 3D model using domain randomization.

3. Approach

We choose to compare one of the one-stage detection approachs and two of the two-stage detection approachs, which is PVNet[8], EfficientPose[2], and Augmented Autoencoders (AAE)[12]. They are all RGB image based model, where both PVNet and EfficientPose are supervised learning, while AAE is self-supervised learning. From the papers of these methods, PVNet aims that using keypoint voting and unit vectors make it robust for occluded or truncated objects, EfficientPose aim to be efficient, scalable, and accurate using single shot image to detect multiple objects, and finally, AAE aims that it doesn't need pose annotation data to achieve robustness to pose ambiguity. Hence, we want to go through these methods, and explore the performance of them.

3.1. PVNet

This paper introduce a Pixel-wise Voting Network (PVNet)[8] to regress pixel-wise unit vectors pointing to the keypoints and use these vectors to vote for keypoint locations using RANSAC. Unlike other recent methods, PVNet can address the problem of being sensitive to occlusion and truncation. They created a flexible representation for localizing occluded or truncated keypoints. They estimated the object pose using a two-stage pipeline: first detect 2D object keypoints using CNNs, and then compute 6D pose parameters using the PnP algorithm.

- Voting-based key point localization Given an RGB image, PVNet predicts pixel-wise object labels and unit vector $\mathbf{v}_k(\mathbf{p})$ that represent the direction from every pixel to every keypoint.
  The unit vector $\mathbf{v}_k(\mathbf{p})$ is defined as

$$\mathbf{v}_k(\mathbf{p}) = \frac{\mathbf{x}_k - \mathbf{p}}{\|\mathbf{x}_k - \mathbf{p}\|_2}$$

  Once the directions to a certain object keypoint from all pixels belonging to that project are given, they generate hypotheses of 2D for that keypoint. They randomly choose two pixels and take the intersection of their vectors as a hypothesis $\mathbf{h}_{k,i}$ for the keypoint $\mathbf{x}_k$. Repeated N times, they can get a set of hypotheses $\mathbf{h}_{k,i}|i = 1, 2, \ldots, N$ representing possible keypoint locations.
  Finally, they can calculate the voting scores $w_{k,i}$ through RANSAC-based voting.

$$w_{k,i} = \sum_{\mathbf{p}\in O} \mathbb{I}(\frac{(\mathbf{h}_{k,i} - \mathbf{p})^T}{\|\mathbf{h}_{k,i} - \mathbf{p}\|}\mathbf{v}_k(\mathbf{p}) \geq \theta)$$

  The resulting hypotheses characterize the spatial probability distribution of a keypoint in the image. The mean μ_k and the covariance σ_k for a keypoint $\mathbf{x}_k$ are estimated by:

$$\mu_k = \frac{\sum_{i=1}^{N} w_{k,i}\mathbf{h}_{k,i}}{\sum_{i=1}^{N} w_{k,i}},$$

$$\Sigma_k = \frac{\sum_{i=1}^{N} w_{k,i}(\mathbf{h}_{k,i} - \mu_k)(\mathbf{h}_{k,i} - \mu_k)^T}{\sum_{i=1}^{N} w_{k,i}}$$

In this paper, the keypoints are selected on the object surface, and these key points should spread out on the object to make the PnP algorithm more stable. They selected K key points using the farthest point sampling (FPS) algorithm.

- Uncertainty-driven PnP Given 2D key points locations for each object, its 6D pose can be computed by solving the PnP problem. Given the estimated mean μ_k and covariance matrix Σ_k for $k = 1, \ldots, K$, they can compute the 6D pose (R, t) by minimizing the Mahalanobis distance. They directly minimize the reprojection errors using the Levenberg-Marquardt algorithm.

They used the smooth l_i loss proposed in [9] for learning unit vectors, which id defined as

$$l(\mathbf{w}) = \sum_{k=1}^{K}\sum_{\mathbf{p}} l_1(\triangle\mathbf{v}_k(\mathbf{p};\mathbf{w})|_x) + l_1(\triangle\mathbf{v}_k(\mathbf{p};\mathbf{w})|_y),$$

$$\mathbf{v}_k(\mathbf{p};\mathbf{w}) = \tilde{\mathbf{v}_k}(\mathbf{p};\mathbf{w}) - \mathbf{v}_k(\mathbf{p})$$

3.2. EfficientPose

EfficientPose[2] is a highly accurate, scalable, and efficient 6D object pose estimation model that can detect the 2D bounding box of multiple objects and estimate their full 6D poses in a single shot. This model is extended from the state-of-the-art 2D object detection EfficientDet[13] and adds two extra subnetworks to predict the translation and rotation of objects, similar to the classification and bounding box regression subnetworks. The authors further proposed a new 6D augmentation technique that uses image rotation and scaling transformation to generate more data, which boosts the performance of their model on small datasets like LineMOD[6] and improve generalization. This approach achieves a new state-of-the-art accuracy of 97.35% in terms of the ADD(-S) metric on LineMOD, while still running end-to-end at over 27 FPS. It runs even with multiple objects (eight) end-to-end at over 26 FPS.

The goal of EfficientPose is to extend the EfficientDet architecture in an intuitive way and keep the computational overhead rather small. The authors integrated the task of 6D pose estimation via two subnetworks to predict the translation and rotation of objects and used the anchor box mapping and non-maximum-suppression (NMS) of the bas architecture to filter out background and multiple detections, which created an architecture that can detect: Class, 2D bounding box, Rotation, and Translation, of one or more object instances and categories for a given RGB image in a single shot. The extra subnetworks are described as follow:

- Rotation Network: The subnetwork predicts one rotation vector $\mathbf{r} \in \mathbb{R}^3$ for each anchor box. The authors added an iterative refinement module on the output $\mathbf{r}_{init}$. The module takes the current rotation $\mathbf{r}_{init}$ and the output of the last convolution layer prior to the initial regression layer to output $\Delta\mathbf{r}$, and iteratively compute N_{iter} times to get the final rotation vector:

$$N_{iter}(\phi) = 1 + \lfloor\phi/3\rfloor$$

$$\mathbf{r} = \mathbf{r}_{init} + \Delta\mathbf{r}$$

Each intermediate iteration step $\mathbf{r}$ is set to $\mathbf{r}_{init}$ for the next step. The authors also replace batch normalization with group normalization to reduce the minimum needed batch size during training.

- Translation Network: Similar to the rotation network described above with the difference of outputting a translation $\mathbf{t} \in \mathbb{R}^3$ for each anchor box. The authors adopt the approach of PoseCNN[17] and split the task into predicting the 2D center point $\mathbf{c} = (c_x, c_y)^\top$ of the object in pixel coordinates and the distance t_z separately. It is equal to predicting the relative offset to the center point $\mathbf{c}$ at each point in the given feature map.

This model is scalable using a coefficient ϕ, which is inherent from EfficientDet [13] that could change the number of layers in the model.

The loss function adopted the PoseLoss and ShapeMatch-Loss from PoseCNN[17]. For asymmetric and symmetric objects is defined as follows:

$$L_{asym} = \frac{1}{m}\sum_{\mathbf{x}\in M} \|\mathrm{Rot}(\tilde{\mathbf{r}}, \mathbf{x}) + \tilde{t}) - \mathrm{Rot}(\mathbf{r}, \mathbf{x}) + t)\|_2$$

$$L_{sym} = \frac{1}{m}\sum_{\mathbf{x_1}\in M} \min_{\mathbf{x_2}\in M} \|\mathrm{Rot}(\tilde{\mathbf{r}}, \mathbf{x_1}) + \tilde{t}) - \mathrm{Rot}(\mathbf{r}, \mathbf{x_2}) + t)\|_2$$

$\mathrm{Rot}(\tilde{\mathbf{r}}, \mathbf{x})$ and $\mathrm{Rot}(\mathbf{r}, \mathbf{x})$ represents the rotation of $\mathbf{x}$ with the ground truth $\mathbf{r}$ and the estimated rotation $\tilde{\mathbf{r}}$ by applying the Rodrigues' rotation formula. Also, m is the number of points and M represents the set of the object's 3D points. It calculates the mean point distances between the transformed model points and directly optimized on the ADD metric. For symmetric objects, the minimal distance for each point to any point in the other transformed point set is taken into account.

The transformation loss L_{trans} is defined as follows:

$$L_{trans} = \begin{cases} L_{sym} & \text{if symmetric,} \\ L_{asym} & \text{if asymmetric.} \end{cases}$$

3.3. Augmented Autoencoders

Augmented Autoencoders (AAE)[12] is a real-time, RGB-based pipeline for object detection and 6D pose estimation. It is a self-supervised method which doesn't require true, pose-annotated data. Instead of learning the explicit representation and mapping the image to the pose, it learns an implicit pose orientation by encoding the image into the latent space. The approach has many advantages: (1) it could handle with pose ambiguity, (2) it is robustness to occlusion and (3) it is generalized to different test sensor and environment.

- Augmented Autoencoders: The autoencoders is constructed based on [10], which is trained to encode an input with $X \in R^D$ to a latent representation $z \in R^n$ where $n << D$, which could be written as:

  $$\hat{x} = (\Psi \circ \Phi)(x) = \Psi(z)$$

  In [14], the input image is added with artificial random noise to train the Denoising Dutoencoder, which shows that latent representations are invariant to noise. In this paper, it applied random augmentations $f_{augm}(\cdot)$, than the target for the AAE. Thus, the target for the AAE becomes:

  $$\hat{x} = (\Psi \circ \Phi \circ f_{augm})(x) = (\Psi \circ \Phi)(x') = \Psi(z')$$

  and it is showed that representing the orientation based on the appearance is efficient to avoid pose ambiguities.

- Synthetic Object Views for Training: To train the autoencoder, the paper proposed a Domain Randomization (DR) technique that makes encoding invariant to environment and sensor variation. It applied random augmentation to the training view, including rendering random light position, diffuse, reflection, inserting background from Pascal VOC dataset[3]. Also, it modified the image contrast, brightness, Gaussion blur, distortion and applied random masks and black squares to generate occlusions.

- Network Architecture: The encode-decoder's architecture is a CNN-based model, which is constructed with 2D convolution layers, ReLU layers with fully-connect layers. Bootstrapped pixel-wise L2 loss[16] is applied to avoid converging to local minima that generated black images. [16].

- Codebook Creation: Once the training is done, the AAE is able to encode 3D objects in latent representation. Then, to determine object orientation, it creates a codebook. First, the object is rendered at nearly equidisitant viewpoints using refined icosahedron[4]. Then, the view is applied in-place rotation. Finally, all resulting images is used to generate latent code in $z \in R^128$ and assigns rotation $R_{cam2obj} \in R^{3x3}$ by the codebook. Thus, in testing, the detected and cropped object could be encode to a $z_{test} \in R^{128}$. Then, cosine similarity between z_{test} and all $z_i \in R^{128}$ in the codebook.

  $$cos_i = \frac{z_i z_{test}}{\|z_i\| \, \|z_{test}\|}$$

  Finally, the 3D object orientation could detemined by finding the k-Nearest-Neighbor with $k = 1$.

4. Experiments

4.1. Dataset

To compare the performance between these different approaches, we selected two benchmark datasets: LineMOD [5] and Occlusion LineMOD Dataset[1].

- **LineMOD**[5] contains 15 texture-less objects which are placed in different cluttered scenes and provides annotations for around 1,000 RGB images for each of the 15 objects. The images in each scene contain multiple objects but only one object is annotated with the ground-truth class label, bounding box, and 6D pose. Most papers only use 13 objects in the dataset to test their approach, we also use the same settings.

- **Occlusion LineMOD**[1] is a subset of LineMOD dataset, which consists of a single scene of LineMOD where additional ground-truth annotations for eight objects in that scene are provided. This introduces more challenging test cases with various levels of occlusion.

4.2. Evaluation Metrics

- ADD [6]: Given the ground truth rotation $\mathbf{R}$, translation $\mathbf{T}$, and the estimated rotation $\tilde{\mathbf{R}}$, translation $\tilde{\mathbf{T}}$. And M denotes the set of 3D model points and m is the number of points.

  $$ADD = \frac{1}{m} \sum_{x \in M} \|(\mathbf{R}\mathbf{x} + \mathbf{T}) - \tilde{\mathbf{R}}\mathbf{x} + \tilde{\mathbf{T}}\|$$

  The average distance computes the mean of the pairwise distances between the 3D model points transformed.

- ADD-S [17]: For symmetric objects, the mean distance is computed based on the closest point distance.

  $$ADD-S = \frac{1}{m} \sum_{x_1 \in M} \min_{x_2 \in M} \|(\mathbf{R}\mathbf{x_1} + \mathbf{T}) - \tilde{\mathbf{R}}\mathbf{x_2} + \tilde{\mathbf{T}}\|$$

The 6D pose is considered to be correct if the average distance (ADD or ADD-S) is smaller than a predefined threshold.

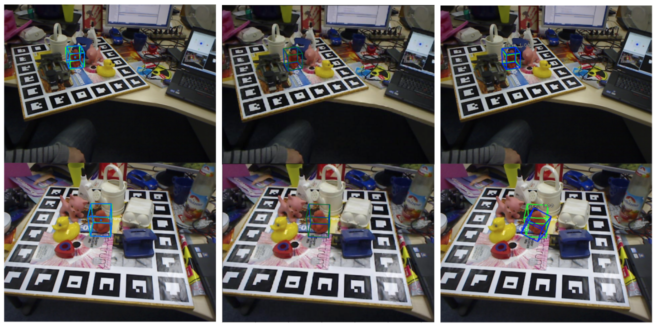

(a) EfficientPose($\phi = 0$) (b) PVNet (c) AAE

Figure 1: Sample output images of three models.

4.3. Experimental details

For experiment, we remain the parameters' settings and evaluate the best model each paper reported on a Google Cloud Platform's virtual machine (n1-standard-4 CPU and Nvidia Tesla K80 GPU). For AAE[12], we couldn't find the precomputed bounding box of the objects, hence, we use the ground-truth bounding box as one of the input. Also, since we couldn't find the weights of AAE model on Occlusion LineMOD, we only evaluate the performance of EfficientPose[2] and PVNet[8] on Occlusion LineMOD dataset.

4.4. Results

In Table 1, we compare the results of three models: EfficientPose, PVNet, and Augmented Autoencoders (AAE), on the LineMOD dataset in terms of the ADD(-S) metric. EfficientPose outperforms the other two methods, followed by PVNet, and AAE has the lowest score. Figure 1 shows some sample outputs of the three models.

EfficientPose has the power of detecting and estimating objects with their 6D poses in only one single shot, without any post-processing steps like RANSAC-based voting or PnP, and still achieves the highest score. The authors of EfficientPose stated that the 6D augmentation technique is a crucial part of this approach, which significantly increases the performance on small dataset like LineMOD. They did some ablation studies on their 6D augmentation method, and found that without the data augmentation, the performance on Driller will drop from 99.90 to 72.15. The performance of $\phi = 0$ and $\phi = 3$ models are close despite of their different capacities. This indicates that the capacity of $\phi = 0$ model is sufficient enough to train the small LineMOD dataset and reach the bottleneck performance.

As for AAE, since it is a self-supervised method and the objective is not mapping the image to the ground truth 3D pose orientation explicitly, we believe it is fair that AAE doesn't reach high scores in this experiment.

Method	EfficientPose		PVNet	AAE
	$\phi = 0$	$\phi = 3$		
Ape	87.71	89.62	49.90	0.50
Benchvise	99.61	99.71	99.81	75.00
Camera	97.94	98.53	86.76	36.00
Can	98.52	99.70	95.96	59.50
Cat	98.00	96.21	76.65	38.00
Driller	99.90	99.50	97.13	30.00
Duck	90.99	89.30	56.34	4.00
Eggbox*	100.00	100.00	99.15	18.50
Glue*	100.00	100.00	95.66	28.00
Holepuncher	95.05	95.72	80.88	30.50
Iron	99.69	99.08	98.67	47.50
Lamp	100.00	100.00	99.14	71.00
Phone	97.98	98.46	92.22	5.10
Average	**97.34**	**97.37**	86.79	32.63

Table 1: Quantitative evaluation and comparison on the LineMOD dataset in terms of the ADD(-S) metric. * represents symmetric object.

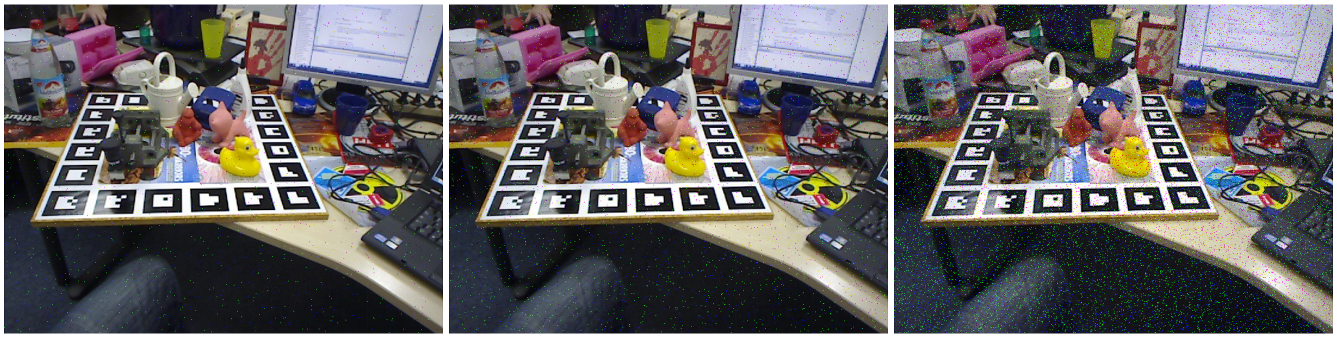
For the Occlusion LineMOD dataset, we compare the results of EfficientPose and PVNet in Table 2. EfficientPose still reaches a higher average score than PVNet, which suggests that enriching the training data on small dataset using data augmentation technique is the keypoint to improve the performance.

Method	EfficientPose		PVNet
	$\phi = 0$	$\phi = 3$	
Ape	56.57	59.29	15.30
Can	91.12	93.37	61.14
Cat	68.58	79.78	19.21
Driller	95.64	97.77	63.76
Duck	65.21	73.23	33.30
Eggbox*	93.46	96.18	50.17
Glue*	85.15	90.80	49.62
Holepuncher	76.43	81.66	39.83
Average	79.02	**84.01**	41.54

Table 2: Quantitative evaluation and comparison on the Occlusion LineMOD dataset in terms of the ADD(-S) metric. * represents symmetric object.

4.5. Noisy Image

In addition to compare the performance, we would like to test the robustness against noises of the three models. We use the LineMOD dataset in this experiment. We generated the noisy images using Salt-and-Pepper noise, an impulse



(a) Amount = 0.004 (b) Amount = 0.01 (c) Amount = 0.05

Figure 2: Example input images after adding Salt-and-Pepper noise with different amount settings

EfficientPose (phi=0)					PVNet				AAE			
S&P Amount	no noise	0.004	0.01	0.05	no noise	0.004	0.01	0.05	no noise	0.004	0.01	0.05
Ape	87.71	84.48	82.67	66.67	49.9	38.54	20.9	0.11	0.5	0.0	0.0	0.0
Benchvise	99.61	99.42	99.32	97.38	99.81	85.93	59.15	0.0	75.0	67.0	65.5	52.0
Camera	97.94	97.75	97.84	96.27	86.76	49.59	10.45	0.0	36.0	24.0	18.0	7.0
Can	98.52	98.18	97.83	95.77	95.96	88.95	10.59	12.13	59.5	48.0	40.0	18.0
Cat	98	97.21	96.11	91.62	76.65	67.17	40.02	4.29	38.0	29.5	17.5	0.5
Driller	99.9	99.5	99.7	98.61	97.13	41.8	4.92	2.08	30	18.5	11.0	4.0
Duck	90.99	90.89	89.3	86.1	56.34	48.07	34.92	10.91	4.0	5.0	5.0	1.5
Eggbox*	100	100	100	99.91	99.15	45.07	35.61	8.49	18.5	15.5	8.0	0.0
Glue*	100	99.9	99.81	99.81	95.66	40.89	33.77	5.52	28.0	22.0	16.5	2.0
Holepuncher	95.05	94.67	94.1	87.82	80.88	61.43	38.91	4.35	30.5	25.5	14.5	0.5
Iron	99.69	99.49	99.39	98.88	98.67	95.19	88.58	4.33	47.5	29.5	13.0	1.0
Lamp	100	99.9	99.81	99.52	99.14	59.5	21.61	1.13	71.0	61.0	57.0	29.0
Phone	97.98	97.6	97.41	95.49	92.22	75.51	56.61	9.11	5.1	44.5	42.0	18.0
Average	97.34	96.85	96.41	93.37	86.79	61.36	35.08	4.8	32.63	26.0	20.53	8.9
Drop		-0.49	-0.93	-3.97		-25.43	-51.71	-81.99		-6.63	-12.1	-23.73

Table 3: Results on different noisy amount images using EfficientPose ($\phi = 0$), PVNet, and AAE.

type of noise in images. We randomly modified each color channel of a pixel with two different values of noise (0 and 255). We changed the noise amount with three settings: 0.004, 0.01, and 0.05. Figure 2 shows some example images of adding Salt-and-Pepper noise in different amount settings. Then, we use the same augmented images as input for the three models and run the evaluation.

Table 3 shows the accuracy drop for three models on different parameter settings. PVNet has the most significant accuracy drop, while EfficientPose only drops slightly and still pertains the highest scores among all the models. As for AAE, we could see that some objects are quite robust to the noise, like Benchvise, while some drops from a high score to nearly zero, like Iron. This suggests that EfficientPose is more robust to noisy input images, while PVNet and AAE is prone to noise. We think that PVNet is unable to detect the keypoints of the object in the noisy image, thus effecting the estimation of 6D pose. Figure 3 shows some sample outputs of three models given amount = 0.05 noisy images.

Network	EfficientPose $\phi = 0$	EfficientPose $\phi = 3$	PVNet	AAE
FPS	8.00	1.94	4.12	15.75
ms	124.96	515.05	242.90	63.49

Table 4: Runtime for network forward propagation.

4.6. Running Time

We use the setting from the original papers that take 480640 RGB image as input. We use n1-standard-4 CPU and one NVIDIA Tesla K80 GPU on the Google Cloud Platform. Table 4 shows the run time for three models on our machine. Due to the Tensorflow version, we run AAE with CPU while the others are run on GPU. The run time is tested on network propagation time. For EfficientPose and PVNet, the time included the time for object detections, while the time for AAE only contains the autoencoder part (pose estimation).

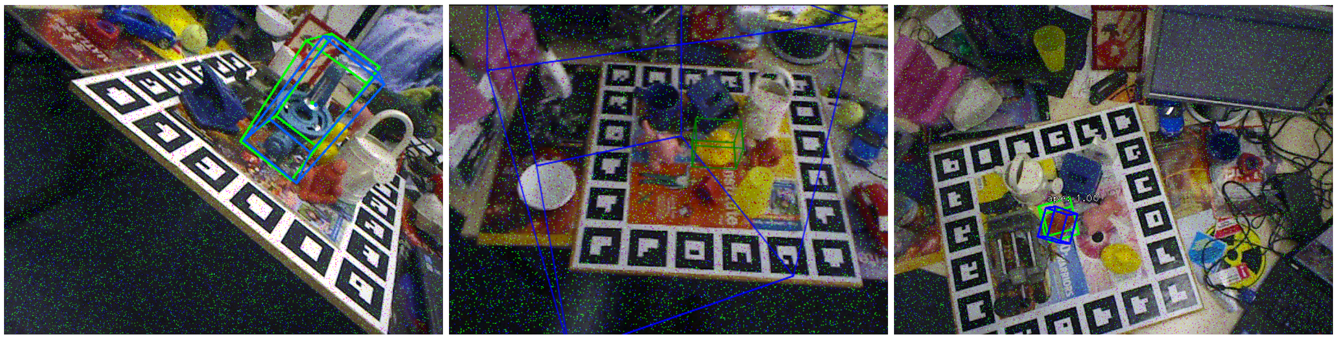

(a) EfficientPose ($\phi = 0$) (b) PVNet (c) AAE

Figure 3: Sample outputs of three models given amount = 0.05 noisy image.

To compare with the run time written on the original papers, Table 5 shows the run time written on the original papers. PVNet [8] runs on Intel i7 3.7GHz CPU and a GTX 1080 Ti GPU. EfficientPose [2] runs on i7-6700K CPU and a GTX 2080 Ti GPU. AAE [12] runs on a GTX 1080 GPU.

Network	EfficientPose $\phi = 0$	EfficientPose $\phi = 3$	PVNet	AAE
FPS	28.18	12.26	25.00	42.00
ms	35.49	81.20	40.10	24.00

Table 5: Runtime end-to-end written on original paper.

We can see from the tables that the run time of AAE are considerably faster than others. EfficientPose are much more slower when setting $\phi = 3$ rather than $\phi = 0$.

5. Conclusion

In all the experiments, we can clearly see that EfficientPose has the most accurate results with a considerable short run time, and it is also robust to noise. EfficientPose also has the advantage that it is scalable through a single parameter ϕ which is convenient to trade off between efficiency and accuracy. PVNet tends to be prone to noise that it would tend to fail when detecting keypoints, but it still has better precision compared to AAE. By constrast, AAE has the lowest ADD comparing to supervised learning methods, but using self-supervised learning would reduce the need of large pose-annotated data, which has great potential for further applications. Also, AAE has the shortest run time and shows fine robustness in the noisy images.

References

[1] 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*, pages 536–551. Springer, 2014. 1, 4

[2] Y. Bukschat and M. Vetter. Efficientpose: An efficient, accurate and scalable end-to-end 6d multi object pose estimation approach. *ArXiv*, abs/2011.04307, 2020. 1, 2, 3, 5, 7

[3] M. Everingham, S. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015. 4

[4] S. Hinterstoisser, S. Benhimane, V. Lepetit, P. Fua, and N. Navab. Simultaneous recognition and homography extraction of local patches with a simple linear classifier. In *BMVC*, pages 1–10, 2008. 4

[5] 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 *2011 international conference on computer vision*, pages 858–865. IEEE, 2011. 1, 4

[6] 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*, pages 548–562. Springer, 2012. 1, 3, 4

[7] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37, 2016. 2

[8] S. Peng, Y. Liu, Q. Huang, H. Bao, and X. Zhou. Pvnet: Pixel-wise voting network for 6dof pose estimation, 2018. 1, 2, 5, 7

[9] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016. 3

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985. 4

[11] S. I. Sergey Zakharov, Ivan Shugurov. Dpod: 6d pose object detector and refiner. *ArXiv*, 2019. 2

[12] M. Sundermeyer, Z.-C. Marton, M. Durner, and R. Triebel. Augmented autoencoders: Implicit 3d orientation learning for 6d object detection. *International Journal of Computer Vision*, 128(3):714–729, 2020. 1, 2, 4, 5, 7

[13] M. Tan, R. Pang, and Q. V. Le. Efficientdet: Scalable and efficient object detection, 2020. 3

[14] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010. 4

[15] C. Wang, D. Xu, Y. Zhu, R. Martin-Martin, C. Lu, L. Fei-Fei, and S. Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1

[16] Z. Wu, C. Shen, and A. v. d. Hengel. Bridging category-level and instance-level semantic image segmentation. *arXiv preprint arXiv:1605.06885*, 2016. 4

[17] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes, 2018. 1, 2, 3, 4