Multi-View Matching Network for 6D Pose Estimation

Daniel Mas Montserrat[1], Jianhang Chen[2], Qian Lin[3], Jan P. Allebach[2], Edward J. Delp[1]
[1]Video and Image Processing Laboratory (VIPER), Purdue University
[2]Electronic Imaging Systems Laboratory (EISL), Purdue University
[3]HP Labs, HP Inc.

Abstract

Applications that interact with the real world such as augmented reality or robot manipulation require a good understanding of the location and pose of the surrounding objects. In this paper, we present a new approach to estimate the 6 Degree of Freedom (DoF) or 6D pose of objects from a single RGB image. Our approach can be paired with an object detection and segmentation method to estimate, refine and track the pose of the objects by matching the input image with rendered images.

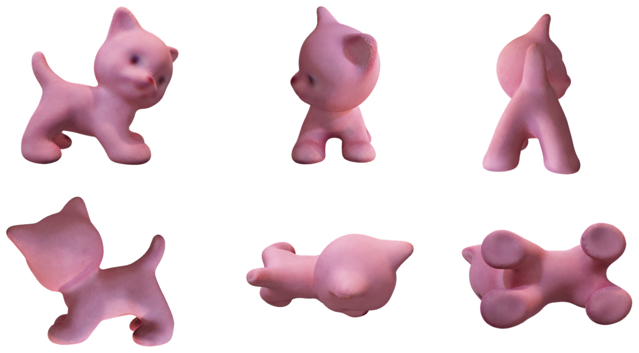


Figure 1. Six rendered views of an object.

1. Introduction

The main challenge when using augmented reality, robot manipulation or autonomous driving is to obtain a clear and detailed understanding of the geometry of the objects appearing in image or video to properly interact with them. In order to understand how the objects in the scene are placed, the pose and location of each element need to be inferred. 6 Degree of Freedom (DoF) or 6D pose estimation techniques can be useful for such tasks. 6D pose estimation consist of inferring the 3D location coordinates and 3D rotation angles of objects in the real world from images, videos or depth information.

Many solutions have been presented for 6D pose estimation working with both RGB-D and RGB images. RGB-D cameras are not highly available (e.g. in smart-phones or laptops) and have many limitations in the depth range, resolution and frame rate making it difficult to detect small, thin or fast moving objects. On the other hand, methods that use RGB-only images can easily fail to correctly estimate the pose of objects with different lighting conditions, occlusions or objects that lack texture or distinctive visual features.

We introduce a new set of neural networks, Multi-View Matching Network (MV-Net) and Single View Matching Network (SV-Net) for 6D pose estimation, refinement, and tracking. These networks are based on DeepIM [8], a neural network that performs pose refinement by estimating the pose difference between pairs of images. MV-Net matches the input image with 6 RGB images containing the object of interest rendered from different views (Figure 1) to obtain an initial estimate of the pose. Then, SV-Net refines the pose estimate in an iterative manner. The same iterative refinement can be used to track the pose in a video sequence.

The main contributions of this paper are as follows. First, we present a new way to extend DeepIM, a pose refinement network, for pose estimation, refinement and tracking, removing the need for an external initial pose estimation method. Second, we show how these networks can be combined with Mask R-CNN to have a complete object detection and pose estimation pipeline. Finally, we evaluate and compare our method with previous pose estimation methods.

2. Related Work

Traditional computer vision methods estimate the 6D pose of an object by matching visual features from the image to a 3D model of the object. The main drawback of this approach is that it fails with textureless objects since only a small number of visual features can be detected on the object. Many recent approaches based on deep learning treat pose estimation as a classification or regression task by extending object classification or detection networks [13, 6]. Although they can handle textureless objects, they are not highly accurate as the pose might be discretized into bins and the methods rely on learning the visual appearance of the objects at different poses. Another approach is to directly regress the 3-dimensional bounding box containing the object with a neural network as in [10].

Recently, methods for pose refinement have been presented showing considerable improvements in accuracy [8]. A common approach is to render an RGB image with an initial pose estimate and then match the rendered image with the input image to obtain a new pose estimate. One of these methods is DeepIM [8], the basis of our work. This method extends FlowNet [2], a neural network trained to estimate the optical flow between consecutive frames, to estimate the pose difference between images.

Multi-view methods have been used in pose related problems. The method presented in [12] uses a multi-view convolutional neural network for shape estimation. The method proposed in [7] uses a multi-view multi-class system for pose estimation.

3. Proposed Method

Our proposed method has multiple stages. First, we use Mask R-CNN [4] to detect and segment the objects of interest in the input image. Next, we estimate the 6D pose of the objects with the Multi-View Matching Network. Finally, the pose estimates are refined with the Single View Matching Network. Figure 2 shows the MV-Net and SV-Net architectures.

3.1. Object Detection And Segmentation

Many object detection and segmentation methods are currently available [11, 4]. In our work, we use Mask R-CNN with ResNet-50 [4] as it provides state-of-the-art results on detecting and segmenting common objects.

We use the bounding boxes estimated by Mask R-CNN for a zoom-in operation to the regions where objects have been detected. The zoom-in operation consists of cropping and resizing to 640×480 the region inside the estimated bounding boxes. Previous to resizing, bounding boxes are expanded to have a 4:3 ratio in order to maintain the aspect ratio. The same zooming in operation is used for the estimated segmentation mask.

3.2. Initial Pose Estimation

Multi-View Matching Network (MV-Net) takes as input 6 rendered images with the detected object in 6 different poses and the zoomed target image to estimate the initial pose of the object.

MV-Net is composed of 6 parallel branches, each of them consisting on the first 10 convolutional layers of FlowNetSimple network followed by one fully-connected layer of dimension 256. The outputs of the fully-connected layers of each 6 branches are concatenated and followed by one fully-connected layer of dimension 512. Finally, two fully-connected output layers of dimension 4 and 3 are added to estimate the relative rotation and translation parameters respectively. The weights of every layer in the 6 branches are shared among them and with SV-Net.

Each branch has as input an 8 channel tensor composed by on the RGB zoomed target image, its segmentation mask (obtained from Mask R-CNN) and a rendered RGB image with its mask. This 8 channel input approach is the same as in [8]. The rendered image used at each branch has a different pose. In our method, we select the 6 equidistant angles in the pitch and yaw dimensions, equivalent to the 6 views of the faces of a cube. Figure 1 shows the rendered images of the 6 views of an object.

The estimated rotation and translation are represented as the relative pose of the target image and the object placed with its frontal face facing the camera (the input of the 1st branch of MV-Net). The initial location of the object is inferred from the center of the bounding box.

The network learns the relative pose with the same representation as in [8], where the relative rotation is expressed with a quaternion and the relative translation is expressed with an untangled representation independent from the coordinate system of the object. Both quaternion and untangled translation parameters are normalized to have zero mean and unit variance.

3.3. Pose Refinement

Single View Matching Network (SV-Net) refines the MV-Net pose estimate. SV-Net consists of one single branch from MV-Net followed by one fully-connected layer of dimension 256. Then 3 fully-connected layers of dimension 4, 3 and 1 are used to estimate the relative rotation, translation and angle distance respectively. Note that without the angle distance estimation output, SV-Net is equivalent to DeepIM[8].

As in MV-Net, the input of the SV-Net consists of one 8 channel tensor composed by the target RGB+mask image, and an RGB+mask rendered image. The rendered image is obtained by rendering the object with the previously estimated pose. The refinement process starts with the initial pose estimate of MV-Net. Then, SV-Net estimates the pose difference between the rendered image and the zoomed input image. The pose difference is used to update the previous pose estimate. A new image is rendered with the updated pose and the refinement process is repeated until the estimated rotation angle, $\hat{\theta}$, between the two images is below a threshold $T_{ref} = 2°$. If after 50 refinement iterations, the estimated rotation angle isn't below T_{ref}, the pose estimate with the lowest estimated rotation angle is selected. This policy to stop the refinement process differs from other methods (e.g. [8]) where a fixed number of refinement iterations are used.

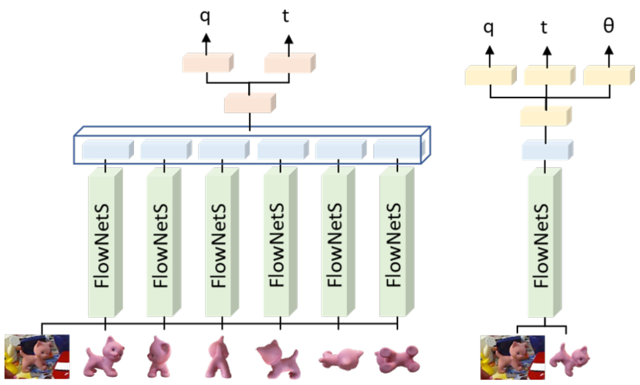


Figure 2. Multi-View Matching Network (left) and Single-View Matching Network (right).

3.4. Pose Tracking

As in DeepIM, SV-Net pose refinement can be easily extended for pose tracking. After the pose is estimated and refined in the first frame, the pose tracking is performed by estimating the pose difference between the current pose estimate and the next frame of the video. Then, the pose estimate is updated, and the process is repeated for the next frame.

The angle distance estimate of the SV-Net can be used to assist the tracking process. If the angle distance estimate is higher than a threshold $T_{high} = 25°$, we restart the tracking process by estimating a new initial pose with MV-Net. If the angle distance estimate is lower than a threshold $T_{low} = 2°$, the pose estimate is not updated.

3.5. Training Process

Our networks are trained using the LINEMOD[5] dataset and synthetic images described in Section 3.6. These datasets include the 3D models of multiple household objects and images containing such objects. We use as ground truth the bounding box surrounding the objects of interest, their segmentation mask, and their pose represented with a rotation quaternion, and the 3D location values.

We fine-tune Mask R-CNN using stochastic gradient descent (SGD) with a learning rate of 0.005 for 10 epochs. We use pre-trained weights from the MS COCO dataset [9].

We train MV-Net and SV-Net using the loss functions in Equation 1 and Equation 2 respectively.

$$L_m(p,\hat{p}) = \frac{1}{N}(\|1-q^T\frac{\hat{q}}{\|\hat{q}\|}\| + \|t-\hat{t}\| + \|1-\|\hat{q}\|\|) \quad (1)$$

$$L_s(p,\hat{p}) = \|1-q^T\frac{\hat{q}}{\|\hat{q}\|}\|+\|t-\hat{t}\|+\|1-\|\hat{q}\|\|+\|\theta-\hat{\theta}\| \quad (2)$$

Where $p = [q|t]$ and $\hat{p} = [\hat{q}|\hat{t}]$ are the ground truth and estimated rotation quaternion and translation parameters respectively. θ and $\hat{\theta}$ are the ground truth and estimated angle distance respectively. N is the number of views. In this work we use $N = 6$.

We follow a simultaneous end-to-end training approach. The training process starts by estimating the initial pose with MV-Net. Then, the initial estimate is used to render a new training sample for SV-Net. We perform a total of 3 refinement iterations during training time. In order to avoid training SV-Net with bad initial estimates, we generate a new sample with a small random rotation if the angle error of the initial estimate is higher than $25°$.

We initialize the FlowNetS convolutional layers of MV-Net and SV-Net with pre-trained weights for optical flow estimation. The fully-connected layers are initialized with random weights as in [8]. We train the networks using SGD with learning rate 0.001 during 48 epochs.

Our simultaneous end-to-end training approach has several advantages. First, we can easily apply weight sharing between MV-Net and SV-Net. Second, we present the training examples to the network in a similar same order as in testing time. Therefore, SV-Net learns to fix the mistakes produced by MV-Net.

3.6. Datasets

We use the LINEMOD (LM) dataset [5] for training and testing. This dataset contains 15 objects but we only use 13 as in previous works [8]. We use the same training and testing split as in [8]. In addition, we use two different rendered datasets for training: A domain randomized dataset and a photorealistic dataset. Figure 3 shows some examples of both rendered datasets.

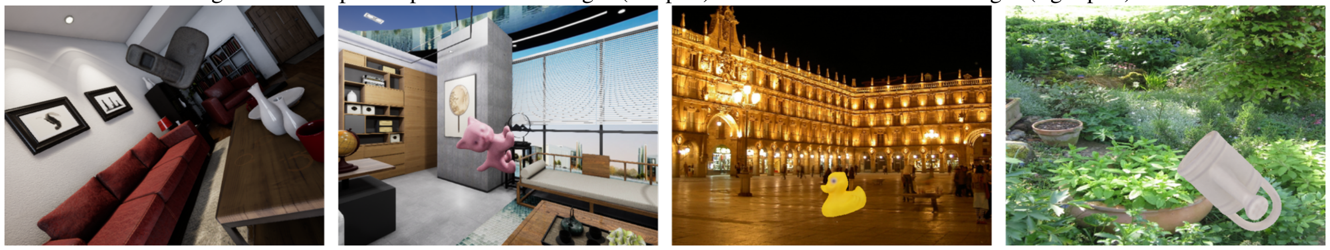
The domain randomized dataset is generated by rendering the objects in random poses and adding background images. The background images are randomly selected from Pascal VOC [3] dataset. Additionally, we apply random blurring and color jittering to the image. The segmentation mask is dilated with a square kernel with a size randomly selected from 0 to 40 in order to mimic errors in the segmentation mask during inference. We render 5,000 training images for each object.

We the 3D models of LM dataset in virtual environments to generate photorealistic images with Unreal Engine 4 (UE4) [1]. This includes a total of 5,000 images per object.

4. Experimental Results

We evaluate our method using the ($n°$, n cm) accuracy metric with the LINEMOD testing set. This metric considers an estimate correct if the error is smaller than $n°$ and n cm and incorrect otherwise. We evaluate our method for $n = 2, 5, 10$. Table 1 presents our accuracy results.

Figure 3. Examples of photorealistic images (left pair) and domain randomized images (right pair).



Method	(n°, n cm)		
	(2, 2)	(5, 5)	(10, 10)
BB8 w/ ref. [10]	-	69.0%	-
PoseCNN [13]	-	19.4%	-
PoseCNN+DeepIM [8]	39.0%	85.2%	97.9%
Ours	24.3%	73.1%	97.5%

Table 1. Comparison with previous methods on LINEMOD dataset.

We can observe that our accuracy results are comparable with previous state-of-the-art RGB-only pose estimation methods such as PoseCNN+DeepIM [8].

Our method shows that pose refinement methods [8] can successfully be extended for initial pose estimation. Therefore, initial pose estimation networks such as PoseCNN [13] can be replaced by highly available object detection networks [4, 11] which can be trained with a large amount of training data.

5. Conclusions

In this paper, we present how a pose refinement network can be extended to perform both pose estimation and refinement. We introduce new research ideas to have a unique network to estimate, refine and track the pose of objects. We show how the network can automatically assist the refinement and tracking process.

References

[1] Unreal Engine 4. URL: www.unrealengine.com.

[2] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. FlowNet: Learning optical flow with convolutional networks. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, December 2015. Las Condes, Chile.

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

[4] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, October 2017. Venice, Italy.

[5] 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. *Proceedings of the Asian Conference on Computer Vision*, pages 548–562, November 2012. Daejeon, Korea.

[6] A. Kundu, Y. Li, and J. M. Rehg. 3D-RCNN: Instance-level 3D object reconstruction via render-and-compare. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3559–3568, June 2018. Salt Lake City, UT.

[7] C. Li, J. Bai, and G. D. Hager. A unified framework for multi-view multi-class object pose estimation. *Proceedings of the European Conference on Computer Vision*, pages 945–953, September 2018. Munich, Germany.

[8] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox. DeepIM: Deep iterative matching for 6D pose estimation. *Proceedings of the European Conference on Computer Vision*, pages 683–698, September 2018. Munich, Germany.

[9] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. *Proceedings of the European Conference on Computer Vision*, pages 740–755, September 2014. Zürich, Switzerland.

[10] 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. *Proceedings of the IEEE International Conference on Computer Vision*, pages 3848–3856, October 2017. Venice, Italy.

[11] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. *arXiv:1612.08242*, 2016.

[12] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. *Proceedings of the IEEE International Conference on Computer Vision*, pages 945–953, December 2015. Santiago, Chile.

[13] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. *Proceedings of Robotics: Science and Systems*, June 2018. Pittsburgh, PA.