# CS 682 Project Report

Arnab Debnath[1], Anowarul Kabir[1], and S M Hasan Mansur[1]

[1]Computer Science, George Mason University

May 14, 2020

**Abstract**

Pose estimation, whether it is object pose estimation or human pose estimation, has always been a challenging and exciting task in the field of computer vision. Lots of progresses have been made in recent years and also there are ample opportunities for improvement. There are two parts of our project. In the first part, we tried to address the problem of 6D object pose estimation especially when the objects are heavily occluded or truncated. We have tried to compare the approaches taken by three recent papers (PoseCnn [1], SegPose [2], PVNet [3]) which have tried to solve this problem and proved to be very efficient and achieved state of the art results. Among the three, we have successfully implemented the latter two using the author's code. Our motivation was to analyze these recent papers so that we can get some insights on how the solution of the problem is formulated, how different approaches have helped and how researchers have tried to improvise on the previous works. In the Second part of our project, we worked with human pose estimation problem. Human Pose estimation is one of the critical tasks that has been around the Computer Vision community for the past few decades. Understanding people's pose in images and videos can unlock many real life application, such as pedestrian's tracking, players pose estimation and so on. DeepPose [4] was the first major paper that applied deep learning to human pose estimation. This studies pose estimation as keypoints regression problem guided by regression loss function. They leverages AlexNet [5] model for building the network. Following their work, we implement the neural network network as regression problem and use PCKh for quality estimation of the model. Finally we compare our model with a existing implementation and show some outputs generated by our model and the ground truth.

## 1 Introduction and Related Work

**6D Pose Estimation** The problem of 6D pose estimation is to detect objects and estimate the translation and orientation (rotation) in 3D. Specifically, for an object in an input image $I$, we want to estimate the 6D pose of that object, that is the translation matrix $T$ and rotation matrix $R$ in 3D, which represent the rigid transformation of the object from the object coordinate system $O$ to the camera coordinate system $C$. There are many applications such as robotics, autonomous driving, augmented reality where estimating the poses of objects are a crucial and essential task. Accurately predicting the poses of objects are a challenging task because of the variety of possible scenarios. Different objects have different shapes and textures. Objects might be partially occluded with other objects in the image. Objects might also be truncated in the image meaning some portion of the object is completely outside the image. There are also issues of variation in lighting and appearance.

Traditional approaches try to estimate the 6D pose of objects by establishing correspondences between 3D models of the object and objects in the images [6, 7, 8, 9, 10]. This approach only works well when the objects have rich textures and shows poor performance in occluded scenes or in case of texture less objects. Some deep learning based approaches try to directly regress images to 6D pose or 3D coordinates [11, 12, 13, 14]. Other approaches first try to directly regress the predefined set of 2D keypoints using neural networks and then estimate the pose using Perspective-n-Point (PnP) algorithm [15, 16]. However, these also can't accurately predict the pose in the case of heavy occlusion, truncation or when symmetric objects are present.

We have selected the following papers for our project that tries to address the above mentioned problems of 6D pose estimation.

1. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes [1].
2. Segmentation-driven 6D Object Pose Estimation [2].
3. PVNet: Pixel-wise Voting Network for 6DoF Pose Estimation [3].

All of these papers have tackled the problem of 6D pose estimation in the case of heavy occlusion. PVNet is the most recent paper which have achieved state-of-the-art results. In the next section, we briefly discuss the technical approaches of each of these papers.

**Human Pose Estimation** is defined as the problem of localization of human joints (also known as keypoints - elbows, wrists, etc) in images or videos. It is also defined as the search for a specific pose in space of all articulated poses. 2D pose estimation - estimate a 2D pose (x,y) coordinates for each joint from a RGB image. 3D pose estimation - estimate a 3D pose (x,y,z) coordinates a RGB image. In this work, we want to understand human pose given a 2D image that contains human.

Understanding people in images and videos is critical for developing many real world applications; for instance, human surveillance in garage or office, detecting pedestrians, human movement detection for self driving cars and so on. Therefore, this problem has been drawn attention of the computer vision community in a great scale. Many of the previous studies focused on graphical model from handmade features. However, by the arrival of deep learning models [5, 17, 18], computer vision community has dived into looking for solutions of many computer vision problems by efficiently designing and optimizing neural network architectures and loss functions. One of the benefits of deep learning approach is that it can extract features guided by the loss function that could be specific to the problems or general from the top view of the solution approach [5, 19].

In this study, we look for an approach for human pose estimation using deep neural networks. The input of the model is an image and the output is keypoint predictions. This study leverages regression problem given some ground truth images and their keypoints set. The dataset used in this study is MPII Human Pose Dataset [20]. MPII Human Pose dataset is a state of the art benchmark for evaluation of articulated human pose estimation. The dataset includes around 25K images containing over 40K people with annotated body joints. The images were systematically collected using an established taxonomy of every day human activities. Overall the dataset covers 410 human activities and each image is provided with an activity label. Each image was extracted from a YouTube video and provided with preceding and following un-annotated frames. In addition, for the test set we obtained richer annotations including body part occlusions and 3D torso and head orientations [20].

This study is basically a reconstruction of a previous work titled as *DeepPose: Human Pose Estimation via Deep Neural Networks* [4]. Deeppose is the first major article that leverages deep learning approach to the problem of 2D human pose estimation and beats existing models. This approach formulates pose estimation as a CNN[5]-based regression problem towards body joints. Finally they use a cascade of such regressors to refine the estimated pose estimation and showed getting better results.

A study titled as *Stacked Hourglass Networks for Human Pose Estimation* [21] is a landmark paper that introduced a novel architecture that beat all previous methods. The naming of stacked hourglass network is as the network consists of steps of pooling and upsampling layers which looks like an hourglass and these are stacked together. The design of the hourglass is motivated by the need to capture information at every scale. While local evidence is essential for identifying features like faces hands, a final pose estimate requires global context. The person's orientation, the arrangement of their limbs, and the relationships of adjacent joints are among the many cues that are best recognized at different scales in the image.

## 2 Technical Approach

For each of these papers below, first we have highlighted the key contribution of the paper, then we have given a brief description about their network architecture and technical approach.

### 2.1 PoseCNN

#### 2.1.1 Key contribution

- Proposed a novel CNN architecture for 6D pose estimation.

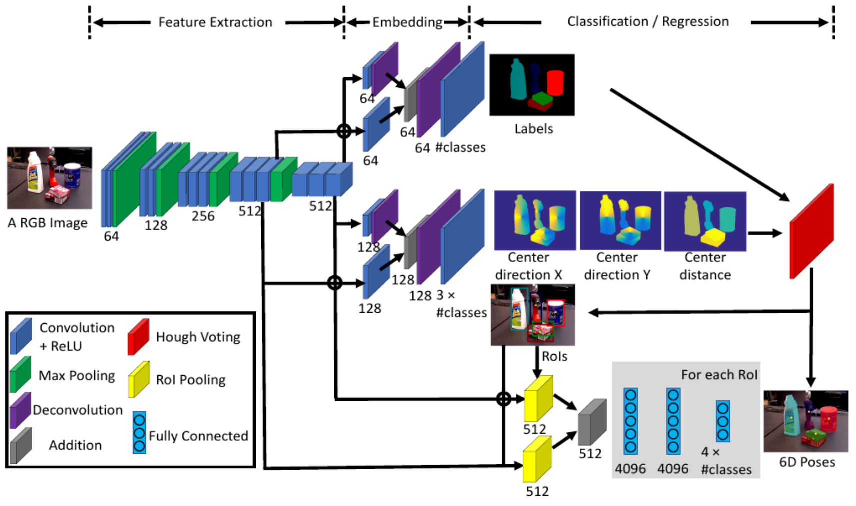


Figure 1: PoseCNN network architecture[1]

- Introduced a new loss function Shape-Match-Loss (SLOSS) for estimating poses of symmetric objects.
- Generated a new RGB-D dataset called YCB-Video Dataset.

### 2.1.2 Proposed Approach

PoseCNN takes an RGB input image I and estimates the 6D poses of some objects from the image. Furthermore, they only use the depth of the objects in the pose refinement stage which shows a much better accuracy than only using the color channels. They require to have 3D object models. First, using a CNN architecture of 13 convolutional and 4 max pooling layers, they extract feature maps from the input image. The network is initialized with VGG16 [22] network trained on ImageNet [23]. These feature maps are shared across the whole network. Then they perform the three following tasks.

**Semantic labelling** Using the feature maps mentioned above, every pixel of the image is semantically labelled. The output of this layer has n channels where n is the number of semantic labels. For training, they use the softmax cross entropy loss.

**3D Translation Estimation** PoseCNN tries to localize the 2D centers $(c_x, c_y)$ of the objects and estimate the distance $T_z$ from the camera. For each pixel, the network regresses the unit vectors to the center direction of the objects.

Specifically, every pixel p(x,y) regress to 3 variables,

$$n_x = \frac{c_x - p}{||c - p||}, n_y = \frac{c_y - p}{||c - p||}, T_z$$

The output of this layer has 3 x n channels, where n is the number of classes. The smoothed L1 loss is used here.

Then, a hough voting layer is used to localize the 2D centers. Hough voting layer takes the semantic labellings and the regression outputs as inputs, and generates some 2D centers based on the voting scores. Basically, every pixels votes for a center location and the location with the highest score is taken as the center of that object. In case of multiple instances in the same image, non maximum suppression is used to the voting scores and centers above a certain threshold are considered. In this way, 2D centers $(c_x, c_y)$ are localized.

Now, the pixels who have voted for the above mentioned centers, are considered to inliers. The distance $T_z$ of the object is taken as the mean of the predicted $T_z$s of the inliers.

From $(c_x, c_y)$ and $T_z$, $T_x$ and $T_y$ are calculated using the following equation.

$$c_x = f_x \frac{T_x}{T_z} + p_x \tag{1}$$

$$c_y = f_y \frac{T_y}{T_z} + p_y \tag{2}$$

Here, $f_x, f_y$ are the focal points and $p_x, p_y$ are the principal points.

The network also generates the bounding boxes the objects which are used for 3D rotation regression.

**3D rotation regression** Using the bounding boxes generated in the previous step, ROI pooling layers are used to get the features maps extracted at the first stage. Then, this feature maps are fed to a FC network. For each class, this networks outputs the 3D rotation represented by a quaternion. The following loss function is used here which can handle the case of symmetric objects.

$$SLOSS(\overline{q}, q) = \frac{1}{2m} \sum_{x_1 \in m} \min_{x_2 \in M} ||R(\overline{q})x_1 - R(q)x_2||^2$$

Here, m is the number of points in the 3D model, M is the set of 3D model points, $R(\overline{q})$ and R(q) are estimated rotation matrix and groundtruth rotation matrix respectively.

**Pose Refinement** If the depth is available, 6D pose estimated from the previous steps can be refined with Iterative Closes Point (ICP) algorithm. After pose refinement, the accuracy increases by a great magnitude but this method takes a lot of time. So, it's not a suitable method in real life scenarios.

## 2.2 SegPose

### 2.2.1 Key contribution

- This paper presents a segmentation-driven approach of 6D pose estimation in a more robust way by combining multiple local predictions instead of a single global one.

### 2.2.2 Proposed Approach

The authors introduce a segmentation-driven 6D pose estimation network in which each visible object patch contributes a pose estimate for the object it belongs to in the form of the predicted 2D projections of predefined 3D keypoints. By using the predicted measure of confidence, the most reliable 2D projections for each 3D keypoint are combined to produce a robust set of 3D-to-2D correspondences. From these correspondences, finally a single reliable pose estimate can be obtained by using a RANSAC-based PnP strategy.

A CNN based two stream network architecture is proposed to jointly perform segmentation by assigning image patches to objects and 2D coordinate regression of predefined 3D keypoints belonging to these objects. To train the model a loss function, which combines a segmentation and a regression term, is defined as follows :

$$L = L_{seg} + L_{reg}$$

The authors take the loss $L_{seg}$ to be the focal loss of [24], a dynamically weighted version of the cross-entropy. The regression loss term $L_{reg}$ is defined as follows where $\beta$ and $\gamma$ modulate the influence of the two terms.

$$L_{reg} = \beta L_{pos} + \gamma L_{conf}$$

The first loss term $L_{pos}$ and the residual used in it are defined as

$$L_{pos} = \sum_{c \in M} \sum_{i=1}^{N} ||\Delta_i(c)||_1$$

$$\Delta_i(c) = c + h_i(c) - g_i$$

where, $c$ is the 2D location of a grid cell center, $g_i$ is the ground-truth 2D location, $h_i(c)$ is an offset such that the resulting location $c + h_i(c)$ is close to the ground-truth $g_i$, $M$ is the foreground segmentation mask and $||.||_1$ denotes the $L^1$ loss function.

The second loss term $L_{conf}$ is defined as follows where $\tau$ is a modulating factor, $s_i(c)$ is a confidence value for each predicted keypoint.

$$L_{conf} = \sum_{c \in M} \sum_{i=1}^{N} ||s_i(c) - exp(-\tau ||\Delta_i(c)||_2)||_1$$

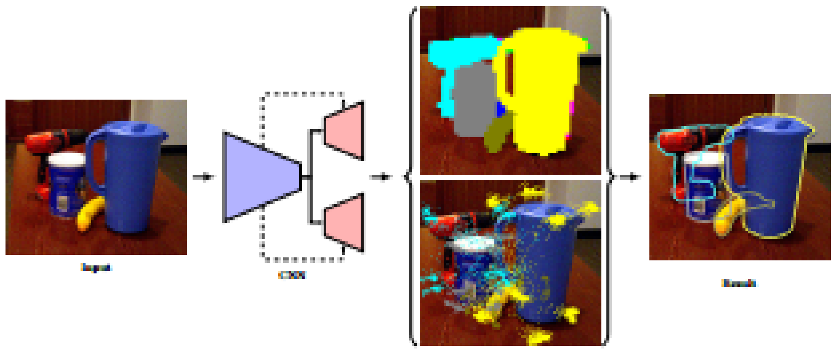


Figure 2: Overall workflow of SegPose. The architecture has two streams: One for object segmentation and the other to regress 2D keypoint locations. These two streams share a common encoder, but the decoders are separate. Each one produces a tensor of a spatial resolution that defines an SxS grid over the image. The segmentation stream predicts the label of the object observed at each grid location. The regression stream predicts the 2D keypoint locations for that object.[25]

Finally, n most confident 2D predictions for each 3D keypoint are combined and the 6D pose of each object is obtained from these filtered 2D-to-3D correspondences by using a RANSAC-based PnP strategy. Figure 3 depict the output of the two stream network and the process of combining pose candidates respectively.

## 2.3 PVNet

### 2.3.1 Key contribution

- Proposed a novel framework which learns a vector field representation for 2D keypoint localization.
- Utilized an uncertainty driven algorithm.
- Created a new dataset Truncated Linemod for evaluation.

### 2.3.2 Proposed Approach

This paper is very similar to the previous two papers. Unlike PoseCNN, this paper only experimented with RGB images.

A pretrained ResNet-18 [26] architecture is used as the backbone network. This network outputs the feature maps which is used for keypoint localization and semantic segmentation. After keypoints are detected, PnP algorithm is used to compute the 6D pose.

The workflow can be divided into the following stages.

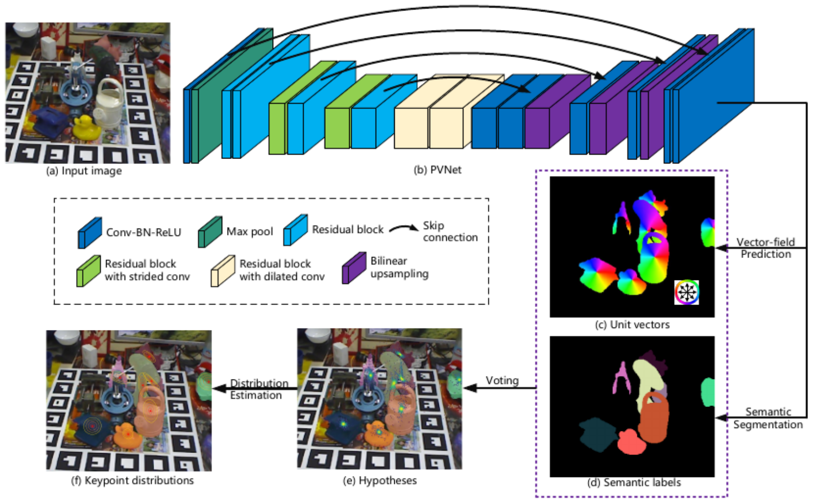


Figure 3: PVNet network architecture[1]

**Semantic segmentation** Given a RGB image I, using the extracted feature maps, for every pixel p, PVNet outputs the semantic label of an object associated with that pixel. Softmax cross entropy loss is used here.

**Keypoint localization** Assuming that there are K keypoints of an object, from every pixel p, PVNet regresses the unit vector $v_k(p)$ which represents the unit vector from pixel p to $x_k$ keypoint of the object.

$$v_k(p) = \frac{x_k - p}{||x_k - p||_2}$$

Smooth l1 loss is used here.

After getting the semantic labels and keypoint estimations from every pixel, a RANSAC based voting scheme is used to select k keypoints. First, two pixels of the target object are selected using the semantic labels, and the intersection of their predecited unit direction is taken as a hypothesis $h_{k,i}$ for the keypoint $x_k$. This is repeated N times to get a set of possible keypoint hypothesis. Then, all pixels of the object votes for these hypotheses and the hypotheses with the highest score is considered as the finalized keypoint.

The resulting hypotheses also gives a spatial probablity distribution of a keypoint in the image. This is used in the next step of PnP algorithm to compute the 6D pose.

**Uncertainity driven PnP** Given the 2D keypoint locations, 6D pose can be computed using a PnP solver like EPnP. The authors of this paper has shown that the uncertainty computed from the keypoint localization step can be used when solving the PnP problem. 6D pose is computed by minimizing the mahalanobis distance,

$$minimize_{R,t} \sum_{k=1}^{K} ((\overline{x_k} - \mu_k)^T \sum_{k=1}^{-1} (\overline{x_k} - \mu_k)^T)$$

$$\overline{x_k} = \pi(RX_k + t)$$

Here, $X_k$ is the 3D coordinate of the keypoint, $\overline{x_k}$ is the 2D projection of $X_k$, and $\pi$ is the perspective projection function. The parameters R and t are initialized by EPnP based on four keypoints, whose covariance matrices have the smallest traces.

## 2.4 Deeppose

Given a pair $(X, Y)$ where $X$ denotes an input image and $Y$ denotes as ground truth pose vector, the network estimates $Y'$ as estimated pose vector. [4] encodes a pose vector as $k$ key body joints where each joint is $x$ and $y$ coordinates. So $Y = (Y_1, Y_2, ......Y_k)$ denotes our pose vector which encodes pose estimation, where $Y_i$ denotes $(x, y)$ coordinates. Then the input pair is normalized with respect to a bounding box $b$. Let, $b$ is defined as box center $b_c$, box height $b_h$ and box width $b_w$. Then the image is cropped sized of the bounding box where the image contains the human on the middle and finally normalizes the image pixel values; this normalizing operation is denoted as $N(X, b)$. Then each key point is normalized as translated as the box center and scaled by the box size which is denoted in the article as $N(Y_i, b)$. The following equation shows the normalization process:

$$N(Y_i, b) = \begin{bmatrix} 1/b_w & 0 \\ 0 & 1/b_h \end{bmatrix} (Y_i - b_c)$$

The article [4] defined the function as $\psi(X, \theta)$ where $X$ is the input image, $\theta$ is the learned parameters and $\psi$ regresses $k$ joints. So in addition to the boxed normalization, the prediction of the points on the absolution image will be as follows:

$$Y^* = N^{-1}(\psi(N(X, b), \theta))$$

where $Y^*$ denotes the predicted joint points and $N^{-1}$ represents the opposite operation of $N(X, b)$ or $N(Y_i, b)$.

The function $\psi$ is denoted as deep learning neural network inspired from AlexNet [5]. This is a convolutional neural network with several layers. Each layers consist of linear transformation followed by non-linear layers excepts for the last three layers. Last three layers are fully connected layers with 50% dropout layer. Dropout layers have advantages, for instance general regularization, encourage all neural nodes to learn something and each dropout of some node generate a fully different model than others which indirectly encourages to work as ensemble methods where multiple models do same thing and final output is the average or maximum of them. For elaborate discussion the reader may refer to AlexNet [5] original paper. The general motivation of using such deep neural network comes from the successful evidence of DNN in the field of classification and object localization problems.

## 3 Results and Discussion

### 3.1 Comparison of PoseCNN vs SegPose vs PVNet

In this section, we try to enlist the differences of their technical approaches to solve the 6D pose estimation problem.

- For the backbone network, PoseCNN used 13 convolutional layers and 4 max pooling layers which were initialized with VGG-16. SegPose used DarkNet-53 architecture of YOLO-v3 and PVNet used a pretrained ResNet-18 as their backbone network.
- For segmentation, a softmax cross entropy Loss is used in PoseCNN and PVNet. A dynamically weighted version of cross entropy (Focal Loss) is used in SegPose.
- PoseCNN separately estimates R and T. It's not a keypoint based approach. The model tries to localize the center of the object. SegPose and PVNet both are keypoint based methods where for each object, N keypoints are detected and 6D pose is then estimated using a PnP algorithm.
- In PoseCNN, From every pixel of the object, an unit vector from that pixel to the center location of the object is estimated. Every pixel also estimates the depth, $T_z$. Then using eqn. 1 and 2, $T_x$ and $T_y$ are calculated. 3D translation stage also outputs the bounding boxes of the objects which is used in the 3D rotation regression stage to regress the 3D rotation.
  In SegPose, 8 corners of the bounding box are taken as keypoints. In PVNet, 8 most distant points on the object surface from the center are taken as keypoints. The CNN architectures of the models try to estimate these set of keypoints.
- PVNet uses the uncertainties of the keypoints in solving the PnP problem.
- The authors of PoseCNN has also showed that depth value can be used to refine the poses. But this stage takes a lot of time. SegPose and PVNet both have only evaluated their model on RGB images. In general, PoseCNN is much slower than the other two approaches.

Some test results of SegPose on occludedLinemod dataset and PVNet on Linemod dataset are shown in fig. 4 and 5 respectively.

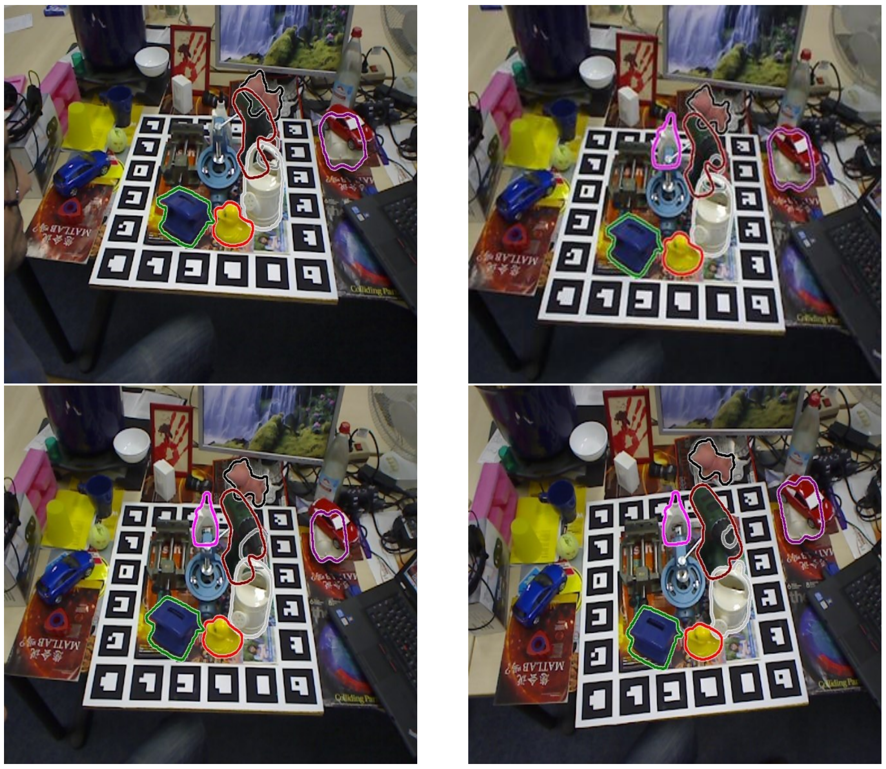


Figure 4: Test results of SegPose on OccludedLineMod dataset

### 3.2 Evaluation on OcclusionLinemod dataset

Two evalutation metric (2D projection error and ADD-s) are used here.

**2D Projection metric** This metric computes the mean distance between the projections of 3D model points given the estimated and the ground truth pose. A pose is considered as correct if the distance is less than 5 pixels.

**ADD-S metric** This metric computes the mean distance between the transformed data points using the estimated and the groundtruth poses. For symmetric objects, the mean is computed based on the closest point distance.

Evalutation results on OcclusionLinemod dataset are shown in table 1.

| 2D projection metric | | | ADD-s metric | | |
|---|---|---|---|---|---|
| PoseCNN | SegPose | PVNet | PoseCNN | SegPose | PVNet |
| 17.2 | 44.9 | **61.06** | 24.9 | 27 | **40.77** |

Table 1: Evalutation on OcclusionLinemod dataset

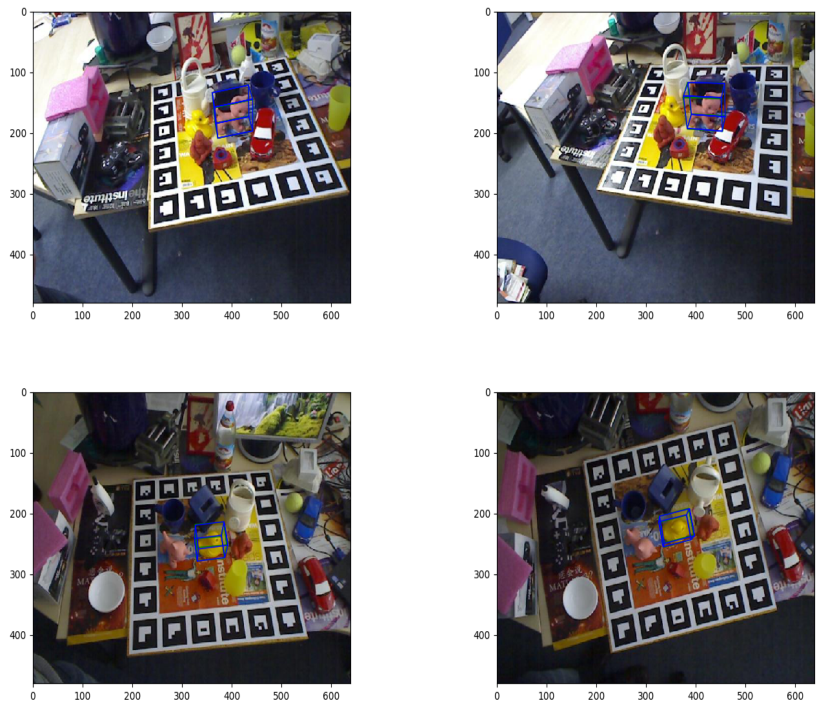


Figure 5: Test results of PVNet on LineMod dataset

### 3.3 Deeppose

Before going into result, lets look at the trained model's final loss and learning rate in Table 2. Learning rate with 1e-6 works better with our model with the same number of epochs and mini-batch size. Throughout the whole training for different hyperparameters, we leveraged adam optimizer provided by the pytorch framework. Note that, the model id indicates nothing specific except for indicating for some visualization works.

We run our models on MPII [20] dataset and optimize the learnable parameter for different hyperparameters and optimize the model using adam optimizer. Figure 6 shows train and validation losses and we see Model30 dominates in both cases. For measuring the quality of out models we used Percentage of Correct Key-points (PCKh) at 50% threshold of the head bone link. A detected joint is considered correct if the distance between the predicted and the true joint is within a certain threshold [27]. The main deeppose article [4] did not evaluate their method on MPII [20] dataset. So we overview some codes in github and found [28] which is mentioned to achieve 54.20% PCKh. On the other hand, none of our implementation could not achieve this far. Our best PCKh is 51.13% for Model30 and other models are far behind that. For looking the reasons be-

| Model id | Learning rate | #epochs | Batch size | Final train loss |
|---|---|---|---|---|
| Model21 | 1e-3 | 40 | 30 | 0.2515 |
| Model22 | 1e-4 | 40 | 30 | 0.2471 |
| Model23 | 1e-5 | 40 | 30 | 0.0260 |
| Model30 | 1e-6 | 40 | 30 | 0.0202 |
| Model31 | 1e-7 | 40 | 30 | 0.0235 |
| Model32 | 1e-8 | 40 | 30 | 0.0308 |

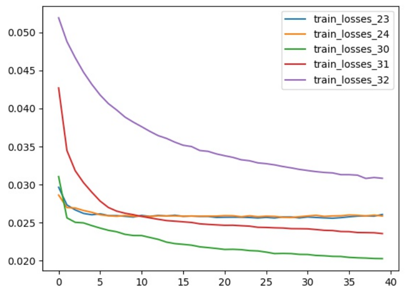
Table 2: Model with parameters

hind that, we finally see we used AlexNet [5] architecture for our model whereas [28] used ResNet [18].

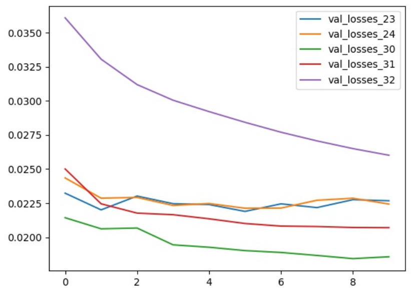
Finally, we like to look at some predicted vs ground truth keypoints. Figure 7 shows some images that are cropped, clipped or rotated with the ground truth krypoints. Figure 8 shows predicted keypoints for each images. From predicted keypoints, we can see that they are almost making a straight line. In very few cases, the model is able to generate keypoints that did not follow straight line.

## 4 Contribution

**Arnab Debnath** I have contributed in the comparison part of our project. I have read the mentioned 3 papers thoroughly and enlisted the differences among them. I have used the authors implementation of PVNet which can be found here and tested on LineMod Dataset on various objects. I have tried to implement PoseCNN from the author's github repository. But it didn't work. I have tried enough to fix it. I guess the problem was with my Cuda version.I have also read another paper [29] which is about 6D object pose estimation



(a) Train losses



(b) Validation losses

Figure 6: Losses for different models

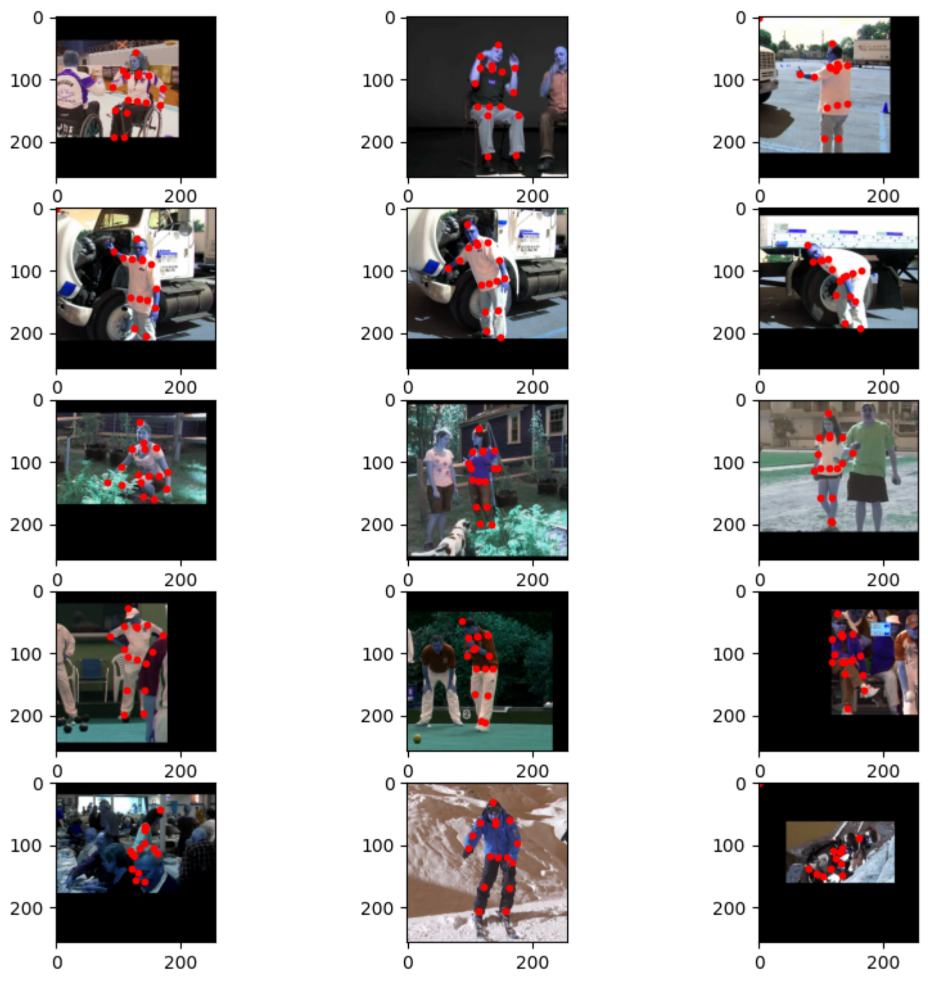


Figure 7: Ground truth Keypoints

using RGB-D images and wanted to include this too. But there are was not enough time to summarize and try to implement this one.

| Model | PCKh |
|---|---|
| Model23 | 36.07 |
| Model24 | 34.25 |
| Model30 | 51.13 |
| Model31 | 36.42 |
| Model32 | 27.96 |
| Deeppose [28] | 54.20 |

Table 3: Evaluation of the models

**Anowarul Kabir** worked with human pose estimation. This study involves implementing the existing method described in the Deeppose [4] article. It also requires to evaluate the implemented method with some existing implementation. The implemented model was then run into gmu argo clusters. And finally the models were evaluated using PCKh evaluation metric and the report was written using the models implemented and the base paper [4]. The code is publicly available in github and it has the instructions about how to run and project directories.

**S M Hasan Mansur** I contributed in literature review mostly in the area of object pose estimation from RGB image and executed the corresponding publicly available code of related papers to evaluate on standard dataset. Though I read several related papers, more specifically, I focused on the papers PoseCNN [1] and SegPose[25]. For the SegPose paper, I faced a problem due to partially available YCB video dataset and I had to write custom functions to handle the issue. I Personally I was interested to review some papers from SMPL based pose estimation. Particularly I tried the code of the paper SMPL eXpressive [30]. Though I rigorously worked to setup the long pipeline of required projects to make it executable, at the end it was getting segmentation fault in runtime, which was a bit frustrating. So I had to skip reviewing that paper.

## 5 Personal Experience

**Arnab Debnath:** I have learnt a lot about 6D pose estimation from this project. From the recent papers, I have learnt how different approaches can be used to solve this problem and how the authors of each paper have tried to overcome the problems of the previous approaches. This gave me some insights on how to think and where to look for, to improve the current accuracy. I have successfully implemented the PVNet paper using the authors code but implementation of PoseCNN from the authors code didn't work. The only problem I see with selecting the 6D object pose estimation problem is that, all of the approaches are required to have 3D models of the objects and are trained on these specific objects. So, this models can't be tested with any random images. It's difficult to find images containing these specific objects, even testing on

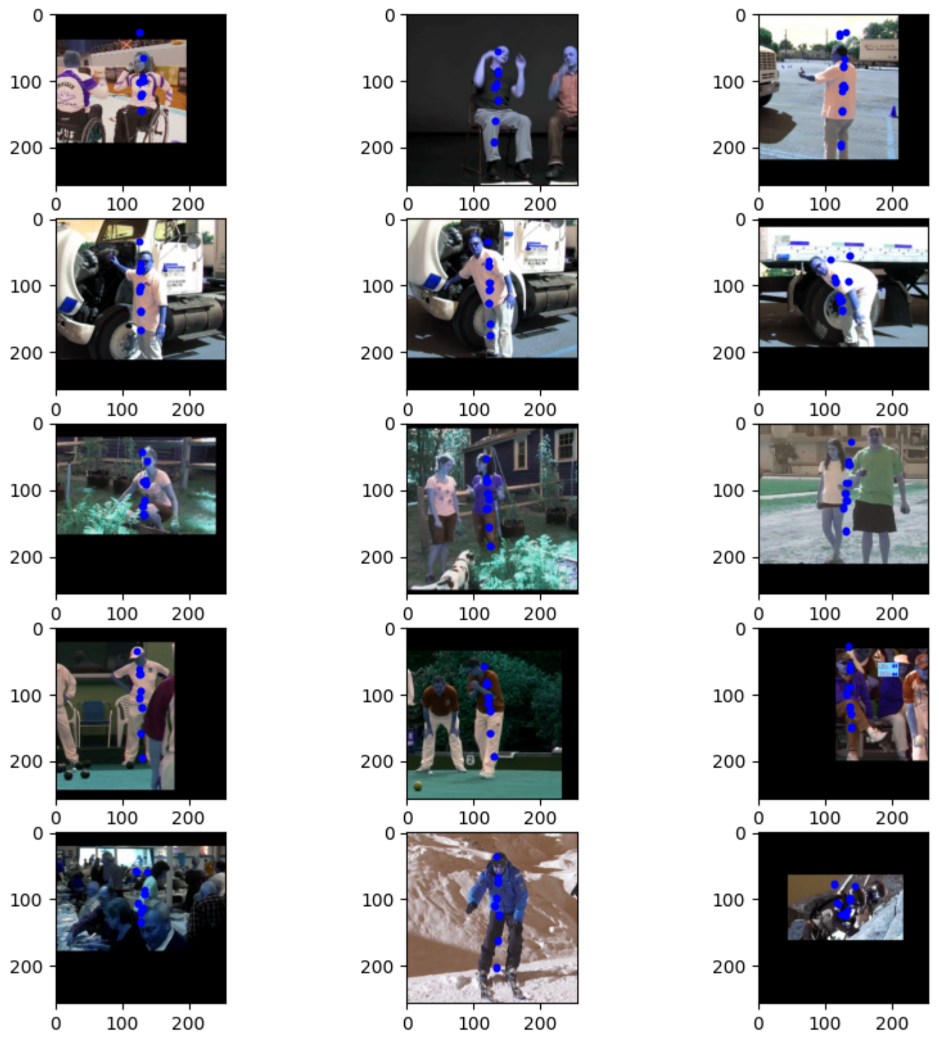


Figure 8: Predicted Keypoints

some image containing similar objects doesn't work properly. I wanted to test on YCB Video dataset, but this dataset is huge. I have tried to download it at least 5 times. But every time it stopped after downloading around 170GB out of 256GB. Currently, I am thinking of gathering more information and read more papers about this 6D object pose estimation to try to increase the accuracy of one of the datasets.

**Anowarul Kabir:** Overall the project goes well. However, these is something to share. When we start our project after the midterm, it was about one month at hand. Within that time period, we had to submit project proposal and work on the project, evaluate and so on. At first it was a bit of hard to come up with a doable project within that time period which could be at the same time exciting and challenging. However, we soon caught our flow. Another big challenge was to run the project with such a big amount of data and wait for at least five hours to go for the next run. Finally, when we run the model for all the combinations of hyperparameters it took almost two and a half days. However, the work is not completely finished. The base article [4] uses a cascade of two such networks. We only tested with one such network. The reason behind that, the use of dropout layers in the fully connected layers. Dropout layers impose some restrictions by dropping nodes so that they can not learn which eventually imposes other nodes to learn. By doing so, the whole network will be an ensemble of neural networks which is able to generate good models. Deeppose [4] was a significant step of estimating human pose estimation, however it a bit of old now. The state of the art works are more comprehensive and rigorous. It could be a good future direction to implement a current state of the art human pose estimation paper.

**S M Hasan Mansur:** As in recent days, I have got much interest in the field of Computer Vision, this particular project was a quite a good opportunity for me to explore a specific research domain, which is Pose Estimation. By reviewing several recent papers, mostly related to object pose estimation, I have got some kind of insight of how the authors identified and approached a problem. As we know, different papers come up with implementation based on different versions of frameworks and packages, sometimes it was quite challenging to make the publicly available code executable in my own machine setup. For example, in case of SegPose paper [25], I faced two particular problem to make it run and evaluate on YCB video dataset. As the YCB video dataset is around 265GB in size and sometimes the corresponding repository restricts the users to download it, I had to look for a partial part of it and to add my own custom function in the existing code base to make it usable. Another thing I noticed that the papers we reviewed don't work properly with random RGB images containing similar objects. Finally, in my personal opinion, it would be better if we could get some project ideas during the first few weeks in the semester rather than after the midterm. Because due to the time constraint, it was quite challenging to come up with a doable project idea and finally accomplish it within one month.

## 6 Conclusion

6D object pose estimation is an interesting problem. Deep learning based methods definitely are a better option to approach this problem. Also, keypoint based methods might be better than trying to localize the center location. For the future work, one can try to divide tasks of the problems in separate sections. As all of the approaches mentioned in the comparison part of our project has used semantic segmentation, using the state of the art semantic segmentation might be useful. We have also seen that the various types of keypoint selection is also an important point to consider.

Human pose estimation is a challenging task. Deep learning imposes end-to-end learning whereas classical methods uses handmade features. DeepPose was the first major paper that applied Deep Learning to Human pose estimation and beats the existing models. Even though the result is not quite expected, however CNN models proves to the removal of handmade features and able to extract the required features guided by the loss function.

## References

[1] 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.

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

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

[4] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR '14. USA: IEEE Computer Society, 2014, p. 1653–1660. [Online]. Available: https://doi.org/10.1109/CVPR.2014.214

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[6] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2. Ieee, 1999, pp. 1150–1157.

[7] 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.

[8] A. Collet, M. Martinez, and S. S. Srinivasa, “The moped framework: Object recognition and pose estimation for manipulation,” *The international journal of robotics research*, vol. 30, no. 10, pp. 1284–1306, 2011.

[9] 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.

[10] 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.

[11] 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.

[12] E. Brachmann, F. Michel, A. Krull, M. Ying Yang, S. Gumhold *et al.*, “Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3364–3372.

[13] 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 *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2686–2694.

[14] 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 *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1521–1529.

[15] 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 *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3828–3836.

[16] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis, “6-dof object pose from semantic keypoints,” in *2017 IEEE international conference on robotics and automation (ICRA).* IEEE, 2017, pp. 2011–2018.

[17] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *International Conference on Learning Representations*, 2015.

[18] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[19] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[20] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, “2d human pose estimation: New benchmark and state of the art analysis,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[21] A. Newell, K. Yang, and J. Deng, “Stacked hourglass networks for human pose estimation,” *CoRR*, vol. abs/1603.06937, 2016. [Online]. Available: http://arxiv.org/abs/1603.06937

[22] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[23] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[24] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, 2020.

[25] Y. Hu, J. Hugonot, P. Fua, and M. Salzmann, “Segmentation-driven 6d object pose estimation,” in *CVPR*, 2019.

[26] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[27] S. C. Babu, “guide to Human Pose Estimation with Deep Learning,” https://nanonets.com/blog/human-pose-estimation-2d-guide/#CPM, 2020, [Online; accessed 15-April-2020].

[28] Naman-ntc, “Deeppose github implementation,” https://github.com/Naman-ntc/Pytorch-Human-Pose-Estimation, 2020, [Online; accessed 15-April-2020].

[29] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3343–3352.

[30] G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. A. A. Osman, D. Tzionas, and M. J. Black, "Expressive body capture: 3d hands, face, and body from a single image," in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.