Edge attention aided 6D pose estimation using Normalised Object Coordinate Space on Customized Dataset

A report submitted

in partial fulfillment for the Degree of

Bachelor of Technology

in

Electronics and Communication Engineering(Avionics)

by

Anju S

(SC16B066)

to

DEPARTMENT OF AVIONICS

INDIAN INSTITUTE OF SPACE SCIENCE AND TECHNOLOGY

THIRUVANANTHAPURAM

July 2020

CERTIFICATE

This is to certify that the report titled **'Edge attention aided 6D pose estimation using Normalised Object Coordinate Space on Customized Dataset'**, submitted by **Anju S**, to the Indian Institute of Space Science and Technology, Thiruvananthapuram, for the degree of **Bachelor of Technology in Electronics and Communication Engineering(Avionics)**, is a bonafide record of the research work done by her under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr Sheeba Rani J
Associate Professor
Department of Avionics

Dr Deepak Mishra
Associate Professor and Head
Department of Avionics

Place: IIST, Thiruvanathapuram
July 2020

Declaration

I declare that this thesis titled **"Edge attention aided 6D pose estimation using Normalised Object Coordinate Space on Customized Dataset"** submitted in fulfillment of the Degree of **Btech in Electronics and Communication Engineering (Avionics)** is a record of the original work carried out by me under the guidance of **Dr Deepak Mishra**, Associate Professor and Head, Department of Avionics, IIST and **Dr. Sheeba Rani J**, Associate Professor, Department of Avionics, IIST, has not formed the basis for any other similar internship training course, diploma, degree, associateship , fellowship or any other titles in IIST or any other university or Institutes of higher learning. In keeping with the ethical practice in reporting scientific information, due acknowledgements have been made wherever the findings of others have been cited.

Anju S
SC16B066

Place: IIST, Thiruvanathapuram
July 2020

Acknowledgements

I sincerely thank my professor and guide **Dr. Deepak Mishra**, Associate Professor and Head, Department of Avionics, IIST for introducing me to the topic of Computer Vision and guiding me through the course of this project, which has helped me in gaining a lot of knowledge in various topics in machine learning. I also thank him for encouraging me at times when I felt stuck while doing the project. I would like to thank my co-guide **Dr. Sheeba Rani**, Associate Professor, Department of Avionics, IIST for her advice and suggestions in the successful completion of my project.

My earnest thanks to the following members of IISU, Durairaj R, Sangeetha GR, Narayanan Namboodiripad M, Sasikumar S, Jayachandran M S and Sam Dayala Dev D for their helpful insights and suggestions for improving my project.

I want to thank my friends for clearing my doubts and motivating me to stay positive. A big thank you to my parents for their unfailing support and encouragement.

Abstract

Computer Vision is the field of machine learning that deals with computers gaining knowledge from digital images/videos and performing tasks that human vision is capable of doing. It finds its applications in a wide range of fields from medical to service sectors like non-invasive medical procedures, virtual shopping, self-driving cars, etc. It is also used in the field of robotics for designing guidance systems where objects in the robot's field of view are identified and located.

Our research work is an application-specific project enabling a half-humanoid to find the 6D pose and bounding boxes of its hand and other objects within its field of view. A Convolutional Neural Network model called the NOCS model has been used for this purpose. We created a new synthetic dataset of the humanoid hand using a software called blender and modified the NOCS model in order to incorporate an extra class prediction. The NOCS model was then trained on this dataset for the 6D pose estimation of the robotic hand. We have added a parallel edge-agreement-loss function that increases the attention to the edges and helps in improving the accuracy of prediction of the instance masks.

Table of contents

List of figures **xiii**

Nomenclature **xvii**

1 Introduction **1**
1.1 Deep Learning in Computer Vision . . . 2
1.1.1 Object detection . . . 2
1.1.2 Instance segmentation . . . 3
1.1.3 Pose estimation . . . 3
1.2 Key contributions of this research work . . . 4
1.3 Chapter Summary . . . 5

2 Literature Survey **7**
2.1 Deep learning in robotics . . . 7
2.1.1 Pose estimation using fiducial markers . . . 8
2.1.2 Lookup table based pose estimation . . . 9
2.2 Computer Vision based pose estimation . . . 10
2.2.1 6D pose estimation models . . . 10
2.3 MaskRCNN model . . . 11
2.4 Edge Agreement Loss . . . 12
2.5 Chapter Summary . . . 14

3 Project idea and implementation **15**
3.1 Requirements and specifications of this research work . . . 15
3.2 CNN model used for pose estimation-NOCS . . . 16
3.2.1 Datasets used for training . . . 17
3.2.2 Loss functions used in NOCS model . . . 18
3.3 Research work and implementation . . . 20
3.3.1 Dataset generation . . . 20
3.3.2 Transfer Learning . . . 22
3.3.3 Training . . . 24

3.4 Chapter Summary 25

4 Results and Discussion **27**
4.1 Testing results of hand dataset 27
4.1.1 Varying orientations 28
4.1.2 Varying lighting 36
4.1.3 Varying scales 42
4.2 Edge-Agreement-Loss 49
4.2.1 Edge errors in NOCS map 53
4.3 Summary of results obtained from our research work 54

5 Conclusions **55**

References **57**

Appendix A Introduction to Neural Networks used **59**
A.1 FPN 59
A.2 ResNet50 60

Appendix B Metrics **61**
B.1 Argmin 61
B.2 MSE 61
B.3 IoU 61

List of figures

1.1 Computer Vision applications 1
1.2 Usage of Deep Learning models in computer vision tasks over the past few years. 2
1.3 The working of an RCNN network 3
1.4 Instance segmentation in a crowded classroom 3
1.5 Different methods of human pose estimation 4

2.1 Deep learning in robotics 7
2.2 Commonly used fiducial markers in computer vision 8
2.3 In this figure, p_is are the 4 corner points of the marker and m_is are their projections on the image plane. β is the complementary of α, which is the angle between p_1p_2 and the optical axis Z'_c.[4] 9
2.4 The NOCS model as a combination of the Mask-RCNN model and an additional convolutional block for NOCS map prediction 10
2.5 The convolutional layers performing various functions in the MaskRCNN model. 11
2.6 MaskRCNN predictions in an image 12
2.7 A comparison between the mask predictions by the baseline model vs Mask-RCNN + Edge Agreement Head after 160k epochs. 13

3.1 A flowchart portraying the navigation of the humanoid arm to the target 15
3.2 NOCS model head architectures added for NOCS map prediction 16
3.3 A 3D camera placed within a unit cube, colour coded according to each pixel's position within the cube. 17
3.4 Example images belonging to CAMERA and REAL dataset 18
3.5 A flowchart of the process followed during our research split into 5 stages starting from hand dataset generation, to transfer learning for weight initialization in the model, including edge attention loss and then training of the model using the created hand dataset. 20
3.6 $1^{st}, 2^{nd}, 3^{rd} and 4^{th}$ columns show sample RGB image, NOCS map, depth image and instance mask respectively from REAL dataset showing a set of 7 table top hand-scale objects.[10]. The depth images shown include the alpha channel. . . 21

3.7 RGB, NOCS map, Depth map, and instance mask images from our hand dataset 22
3.8 Layers of the NOCS model in which changes have been made 23
3.9 Layers of the NOCS model in which changes have been made 24

4.1 Ground truth 6D pose of hand images and 6D pose predicted by our model. . . 28
4.2 The ground truth and predicted NOCS maps of hand in 4 different orientations . 29
4.3 Translation error along the x-axis(Blue axis) in mms vs 125 testing images . . . 30
4.4 Translation error along the y-axis(Green axis) in mms vs 125 testing images . . 30
4.5 Translation error along the z-axis(Red axis) in mms vs 125 testing images . . . 31
4.6 Rotation error along the x-axis(Blue axis) in degrees vs 125 testing images . . . 31
4.7 Rotation error along the y-axis(Green axis) in degrees vs 125 testing images . . 32
4.8 Rotation error along the z-axis(Red axis) in degrees vs 125 testing images . . . 32
4.9 Predicted 6D poses with corresponding NOCS maps explaining the error anomalies. 33
4.10 The figure shows 3D IoU between predicted 3D ground truth and predicted bounding boxes vs testing images. We obtain a mean IoU of 0.5, which is considered to be an acceptable prediction for bounding boxes. 34
4.11 The plots above show ground truth(gt) and predicted(pr) 6D pose of the same hand in 5 different lighting setups 36
4.12 The ground truth and predicted NOCS maps of hand in 4 different lighting conditions corresponding to (c),(g),(k),(s) hands Fig. 4.6. The more is the error in the NOCS map, more is the error in estimated pose. 37
4.13 Translation error along the x-axis(Blue axis) in mms vs 125 testing images. . . 38
4.14 Translation error along the y-axis(Green axis) in mms vs 125 testing images. . . 38
4.15 These pictures demonstrate the error in the prediction of rotation about the x-axis(blue) when the x-axis is parallel to the camera view. 39
4.16 The rotation error about the x-axis(Blue axis) in degrees vs 125 testing images. 39
4.17 The rotation error about the y-axis(Green axis) in degrees vs 125 testing images. 40
4.18 The rotation error about the z-axis(Red axis) in degrees vs 125 testing images. . 40
4.19 The figure shows 3D IoU between predicted 3D ground truth and predicted bounding boxes vs testing images for images in different lighting setups. We obtain a mean IoU of 0.518, which is considered to be an acceptable prediction for bounding boxes and is the same as that for the initial lighting condition. . . 41
4.20 The plots above show ground truth(gt) and predicted(pr) 6D pose of the same hand image at 3 different distances from the camera 42
4.21 The ground truth and predicted NOCS maps of hand at 3 different distances from the camera 43
4.22 Translation error along the x-axis(Blue axis) in mms vs 261 testing images. . . 44
4.23 Translation error along the y-axis(Green axis) in mms vs 261 testing images. . . 44

4.24 Translation error along the x-axis(Blue axis) in mms vs 261 testing images. . . 45
4.25 The rotation error about the x-axis(Blue axis) in degrees vs 186 testing images. 45
4.26 The rotation error about the y-axis(Green axis) in degrees vs 186 testing images. 46
4.27 The rotation error about the z-axis(Red axis) in degrees vs 186 testing images. . 46
4.28 The ground truth and predicted NOCS maps of hand at 3 different distances from the camera 47
4.29 The figure shows 3D IoU between predicted 3D ground truth and predicted bounding boxes vs testing images when the hand is farther from the camera. We obtain a mean IoU of 0.52, which is considered to be an acceptable prediction for bounding boxes. 48
4.30 The figure shows 3D IoU between predicted 3D ground truth and predicted bounding boxes vs testing images when the hand is nearer to the camera. We obtain a mean IoU of 0.33, which indicates poor bounding box prediction. The 3D bounding box in our model is calculated from the dimensions of the 3D point cloud generated from the NOCS map and hence a lesser mean IoU is reflective of the NOCS map error. 48
4.31 The figure shows training mask prediction loss for the baseline model and NOCS model + Edge loss head for different edge detection filters over the first 70 epochs. 49
4.32 The figure shows the overall training loss for the baseline model and NOCS model + Edge loss head for different edge detection filters over the first 70 epochs. 50
4.33 3 examples of instance maps predicted by a model trained without the addition of edge loss vs the model trained with the Laplace filter 51
4.34 Mask predictions by model without edge loss and predictions by model with Laplace edge filter. 52
4.35 (a),(c)-ground truth,(b),(d)-predicted NOCS maps. If we observe the edges we can notice that they are a lot more erratic when compared to the rest of the map area. Errors in the predicted masks lead to errors in the 6D pose estimated. . . . 53
4.36 The ground truth(a) and predicted(b) NOCS edges maps extracted from their respective NOCS maps using the canny edge detection filter used 5 times on the map. 54

A.1 The figure shows the Feature Pyramid Network(FPN) block. The bottom-up convolutional layers and top-down pathway generating different feature maps can be seen. The lateral connection between the layers in the top-down path does the upsampling of output from the present layer and adds that to the output of the next layer. 59
A.2 Comparison between the convolutional blocks used in ResNet34, ResNet50 and ResNet101. 60

B.1 IoU metric = intersection of the ground truth and predicted bounding boxes over the union of the ground truth and predicted bounding boxes. 62

Nomenclature

Acronyms / Abbreviations

3D	3 Degrees of Freedom
6D	6 Degrees of Freedom
wbs	weights and biases
AI	Artificial Intelligence
AR	Augmented Reality
CAD	Computer-Aided Design
CAMERA	Context-Aware MixEd Reality
CNN	Convolutional Neural Network
CV	Computer Vision
DNN	Deep Neural Network
DoF	Degree of freedom
FC	Fully Connected
FPN	Feature Pyramid Network
ICP	Iterative closest point
IoU	Intersection Over Union
LUT	LookUp Table
MSE	Mean Square Error
NN	Neural Network
NOCS	Normalised Object Coordinate Space

R-CNN Region-based Convolutional Neural Networks

RGB-D Red Green Blue - Depth

RNN Recurrent Neural Network

ROI Region Of Interest

RPN Region Proposal Network

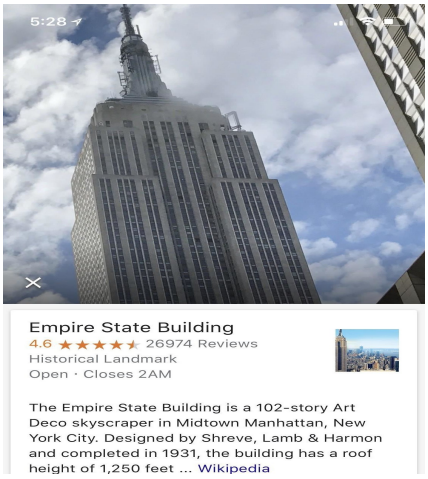
SGD Stochastic Gradient Descent

Chapter 1

Introduction

Computer Vision is a field of study that deals with making computers see and interpret the contents of images/videos, ie. to identify and characterize every section of the image. Biologically speaking, Computer Vision aims to make a computational model that imitates the visual system in a human body. It paves way to a broader problem of Artificial Intelligence.

Computer Vision(CV) has a wide range of real-life applications. A lot of the novel applications launched nowadays such as Amazon Go, Google Lens, Snapchat filters are based on CV ideas. From image search in search engines, face, object and scene recognition in Google Photos, to visual effects in movies and reconstruction algorithms in visual sports replays, CV has reduced human efforts in otherwise laborious tasks to a bare minimum. Advancing research is also happening in medical fields for CAT / MRI reconstruction, assisted diagnosis, automatic pathology, AI-guided surgery, etc.



(a) Google Lens identifying the scanned building.

(b) Image Guided Surgery, visualising internal structures on a patient

Fig. 1.1 Computer Vision applications[1]

1.1 Deep Learning in Computer Vision

Deep Learning models consist of a large number of layers of neurons mimicking the human brain and feature extractor layers similar to the perception of the human eyes. With this assistance, deep learning models achieve state-of-the-art results in computer vision problems. Recently, Deep Learning based computer vision models such as Convolutional Neural Networks(CNNs) and Region-Convolutional Neural Networks(R-CNNs) have taken computer vision tasks to the next level of superiority, even surpassing human vision capacities in certain tasks. But it remains a complex problem since the world we see is dynamic and undergoes constant change in various multitudes.

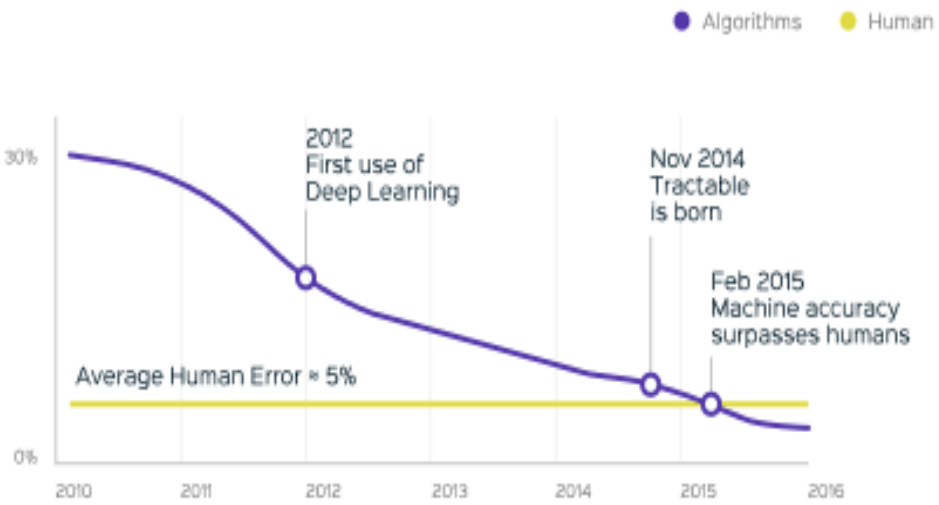


Fig. 1.2 Usage of Deep Learning models in computer vision tasks over the past few years. The accuracy of prediction in vision tasks has improved over the years to the point where it has accuracy better than that by an actual human eye.[2]

Next we will brief a few common CV problems we deal with in this project.

1.1.1 Object detection

Object detection is a prevalent computer vision problem that locates and classifies the objects present in an image. Initially machine learning dealt with images consisting of one single object which had to be classified. A variety of Deep Neural Networks(DNNs) including a variety of CNNs were trained to perform this task. Later on this was extended to classification of objects containing multiple objects. This was done by the addition of a unit called the Region Proposal Network(RPN) which proposed regions consisting of each object in the image, and was then treated as an image classification problem containing only one object.

[1]Google images, http://www.ai.mit.edu/projects/medical-vision/surgery/surgical_navigation.html
[2]https://towardsdatascience.com/computer-vision-an-introduction-bbc81743a2f7

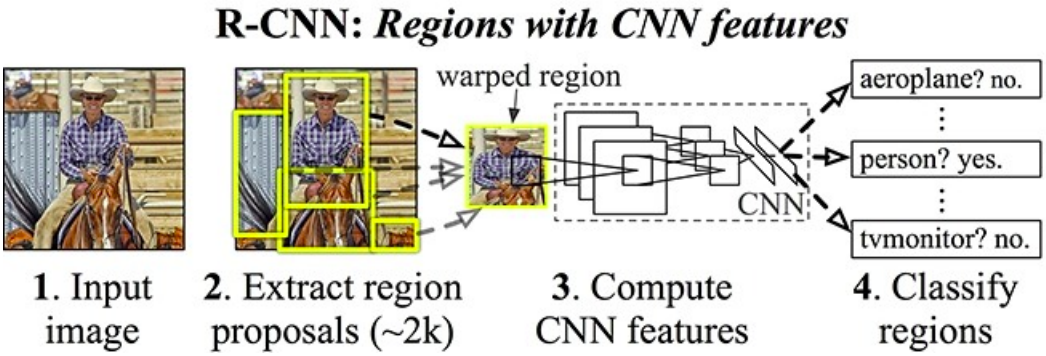


Fig. 1.3 The working of an RCNN network in stages. First stage - In the input image, various regions with the probability of containing objects are predicted, called region proposals are predicted. Second stage - Each region is treated as a single object image and is classified using a CNN.[3]

1.1.2 Instance segmentation

Instance segmentation takes object localisation in an image to the next level by locating exactly which pixels in the image form the object. It gives us the shape of each object and separates it from the background which is a requirement for certain CV tasks like pose estimation. This has proven to work better in cluttered environments prone to occlusion issues.

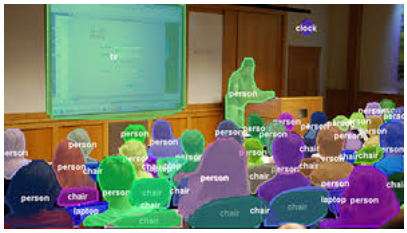


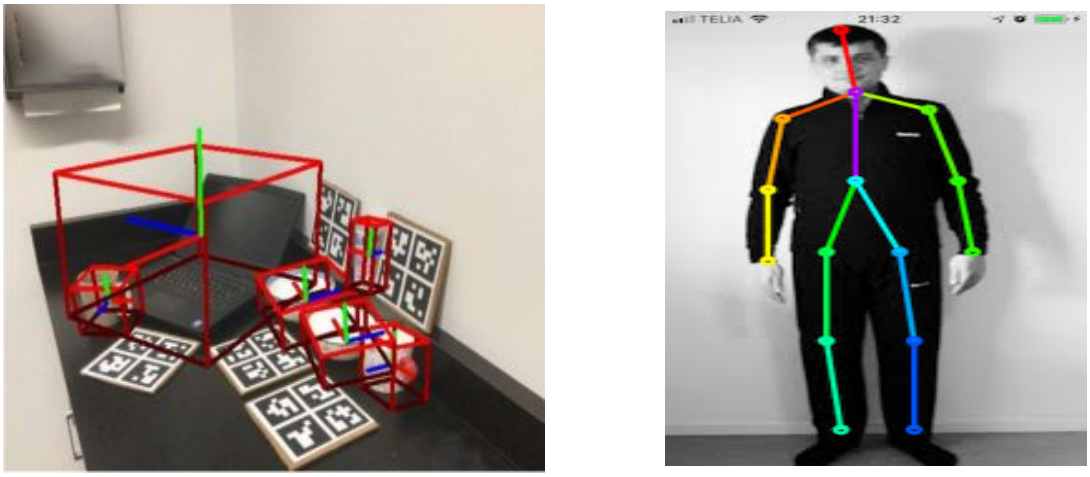
Fig. 1.4 Instance segmentation in a crowded classroom with multiple objects belonging to same or different classes have been separately identified. [5]

1.1.3 Pose estimation

Pose estimation is one of the key areas of research in CV which determines the position and orientation of the objects in an image. It could either be by locating various predefined points on the object in the image and linking them to find the orientation or by simply predicting a rotation vector corresponding to each object(orientation of the whole object). The first approach is used for human pose estimation and the second is used in case of objects. Pose estimation can be done from both RGB and RGB-D(includes depth information) images. We use a dataset of

[3]https://notebooks.azure.com/aribornstein/projects/cvworkshop

RGB-D images for our project since it has proven to provide state-of-the-art results in 6D pose estimation(3 position and 3 orientation DoFs).



(a) (b)

Fig. 1.5 Different methods of human pose estimation. (a) 6D pose estimation of objects by predicting position and rotation vectors [10]. (b) Human Pose estimation by locating positions of key points.[4]

Our work focuses on the 6D pose estimation of objects, which implies predicting the 3 translational and 3 rotational coordinates of every object from an original position.

1.2 Key contributions of this research work

1. Understanding and implemention of the Normalized Object Coordinate Space(NOCS) architecture [10].
2. The creation of a synthetic Hand Dataset using blender software comprising of 1900 training, 300 validation, and 300 testing images, for training the model to recognize and estimate the 6D pose of the hand of a humanoid.
3. Training the NOCS model with the synthetic Hand Dataset using the transfer learning. Transfer learning enables us to train the model on our hand images while retaining the pre-trained information of 6 classes(laptop, mug, bottle, can, camera, bowl).
4. Extensive analysis of the 6D poses obtained from the model using various metrics of evaluation.
5. Addition of the Edge-Agreement-Loss to the loss function to increase the attention on the boundaries of the objects for faster and better training of the instance masks.

[4]https://habr.com/en/post/458000/

6. Training the model with different Edge loss filters and comparing their performances to find the optimal edge filter.

1.3 Chapter Summary

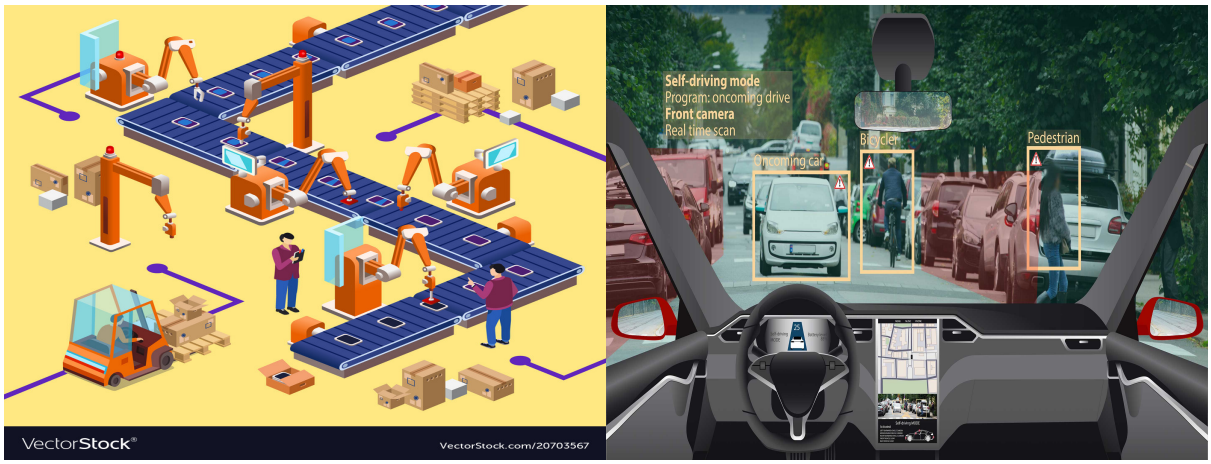
Computer Vision is a field of study that deals with making computers learn the vision tasks performed by humans, ie. to see and interpret the contents of images/videos. The computer learns to identify and characterize every section of the image. It finds its applications in numerous fields such as in the medical industry, Artificial Intelligence(AI), a variety of mobile apps, etc. Object detection, Instance segmentation and Pose and bounding box estimation are the main tasks performed in this research work. Object detection is a basic task in CV that involves identifying the objects present in a given image/video and classifying them to a pre-defined category. Instance segmentation involves finding out the shape or boundary of an object. Pose estimation has two parts, estimating the translation coordinates(x,y,z) and rotation coordinates(ψ,θ,ϕ) of each object with respect to a fixed reference and dimension estimation involves finding out the tight bounding box. Lastly listed are the contributions we made during the course of this research work.

Chapter 2

Literature Survey

2.1 Deep learning in robotics

Designing robots to become human substitutes in mundane day-to-day tasks has always been of human fascination. For eg. working on an assembly line, identifying objects, picking up and placing them back in a different orientation, or in self-driving cars which require identifying all objects in the view and navigating around them, etc. All these applications require the robots to gain knowledge about each object, their position, and orientation. Our research problem is to compute the 6D pose from an RGB-D camera attached to a humanoid, which involves the identification of the orientation of its hand and objects in the view in order to navigate to each object. In this section, we will discuss the various methods that have been previously used along with the method we have used for the 6D pose and bounding box estimation.



(a) Robots in assembly line (b) Self-driving car

Fig. 2.1 (a)Robots performing tasks in an assembly line where picking up of objects is required where computer vision is applied for detecting and identifying the objects.
(b) A self-driving car recognising the objects like cars, trees, street lights etc. and using that information for navigation.

2.1.1 Pose estimation using fiducial markers

Fiducial markers are markers or objects placed in the field of view of imaging systems(cameras) which are used as points of reference or measurements. Computer vision uses fiducial markers that are usually grids with binary patterns printed on them. Once the information about the size and pattern of the grid is known, these markers can be used as anchors of location, orientation, and scale. For pose estimation of an object, fiducial markers of known size and patterns are printed on various faces of the objects and read through cameras. These can estimate the size, orientation, and location of objects by locating keypoints on the objects based on which fiducial is being seen, its orientation, and size.

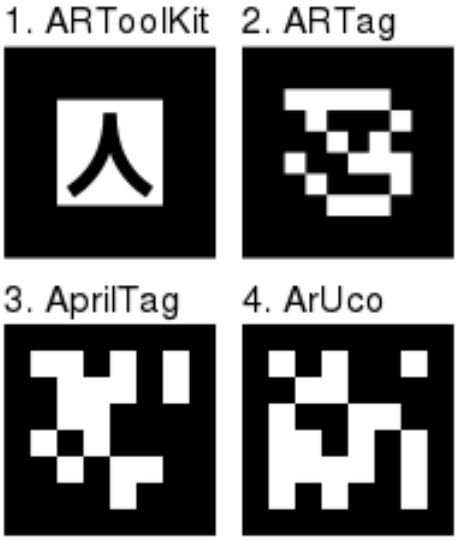


Fig. 2.2 These are some of the commonly used fiducial markers in computer vision[1]

The research work in [1] discusses a simplified camera pose estimation model using fiducial marker recognition. The 2D fiducial markers that are detected are used to match 2D features in the image with their corresponding 3D points. The 2D-3D correspondence is then used to estimate the camera pose. The method proposed proves accurate with varying scale and in poor detection scenarios. Another research paper [13] explores using multiple fiducial markers to improve the accuracy of pose estimation. The camera pose is found out using all the markers which are weighted based on their detection confidence scores. An optimal global pose is obtained from the local transforms of various markers using a proposed algorithm called the coordinate registration algorithm [13].

[1] http://sciencedirect.com/science/article/abs/pii/S0262885619300903

2.1.2 Lookup table based pose estimation

Marker-based methods of pose estimation are usually highly susceptible to noise and are computationally complex. In applications where computational resources are limited, such as mobile AR applications, a trade off between quality and computational speed happens. Methods proposed before the introduction of LookUp Tables (LUTs) involved time-consuming iterative solutions to eliminate the effect of noise in images. With LUTs, a non-iterative, hence less time-consuming, robust and accurate method comes into place for pose estimation for noisy data. In the presence of noise, redundant information can be used for obtaining better accuracy and performance. The research work [4], an extra parameter called the primary rotation angle (β) is extracted from the camera pose and a LUT is created by considering the symmetry of the marker.

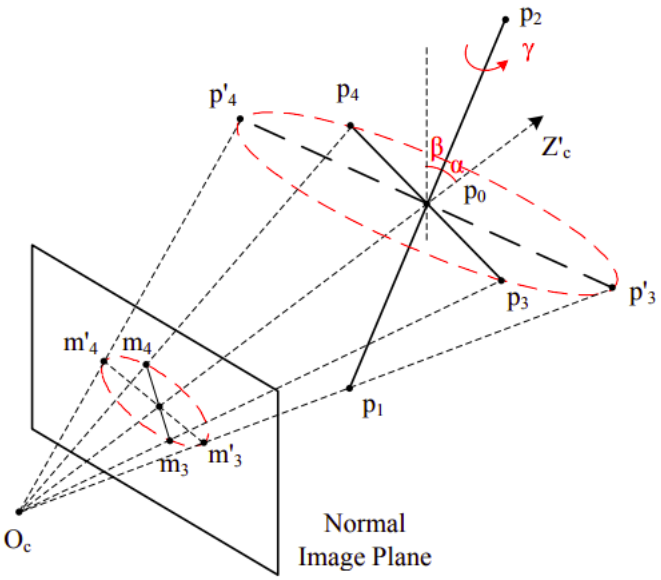


Fig. 2.3 In this figure, p_is are the 4 corner points of the marker and m_is are their projections on the image plane. β is the complementary of α, which is the angle between p_1p_2 and the optical axis Z'_c.[4]

β_i's lookup table is written by varying $\theta = \angle m_1 O_c m_2$, u_i and v_i, which are the normalised image coordinates of m_i. There may also exist a different angle of rotation β' that gives a local minimum. This angle is called the ambiguous pose and can be determined from β, since it will be in the opposite direction to β. Its lookup table is mirror symmetric with that of β's. The camera pose can be determined by first computing the depth's of the 4 corner points of the marker, which can be written as functions of β. Then the camera's position and orientation is are retrieved as the Euclidian motion that align β and β' [4].

2.2 Computer Vision based pose estimation

Learning-based pose estimation has been widely used and has numerous advantages when compared to other conventional methods. These can be used in more dynamic environments where all the objects to be detected aren't previously registered unlike the case in fiducial marker-based systems. Computer Vision also proves useful in poor lighting conditions and noisy environments where fiducials are difficult to recognize and work with. Once trained, a deep learning model can identify any new object belonging to the trained classes. One disadvantage with deep learning methods is the computational complexity and cost of training the model, especially when dealing with large models comprised of hundreds of layers.

2.2.1 6D pose estimation models

6D pose estimation involves locating each object in the image and estimating its orientation. Many different state-of-the-art methods have been proposed over the years to achieve this task. PoseCNN [12] is one such model which finds out the position of the objects, by locating its centre and estimating its distance from the camera and their rotation, by regressing it to a quaternion form. PVNet [7] regresses pixel-wise unit vectors to choose the keypoints using RANSAC and uses them to find the pose using Perspective-n-Point(PnP) algorithm. DPOD [14] also uses PnP and RANSAC algorithms to find pose after estimating the 2D-3D correspondences input images and available 3D models.

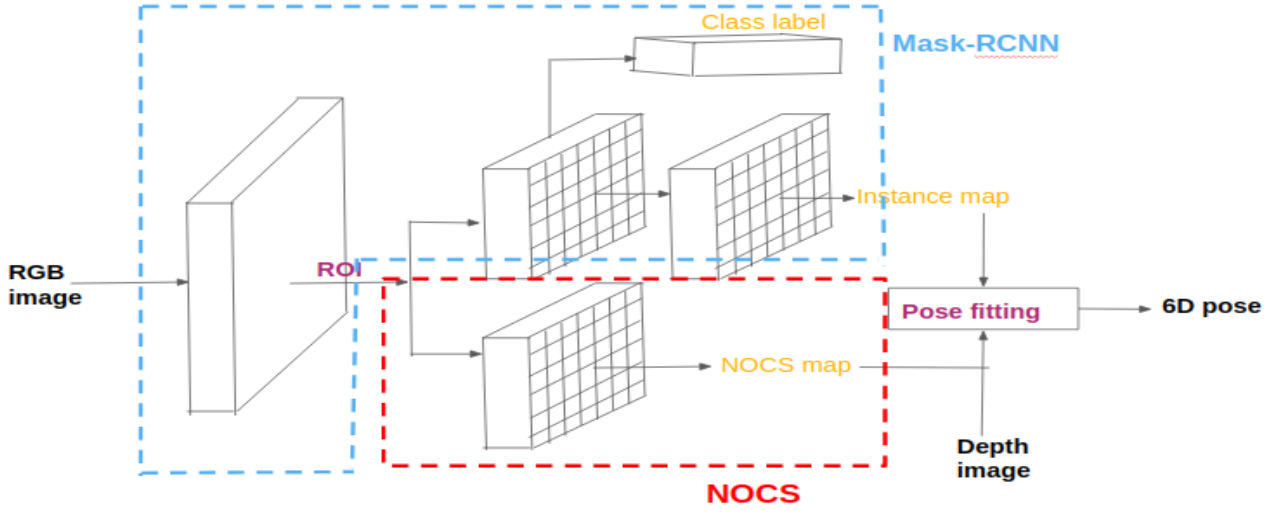


Fig. 2.4 The MaskRCNN predicts class and instance maps for each object and the NOCS convolutional block predicts the NOCS map paralelly from the input RGB image. These maps along with the input depth data are pose fitted to obtain the 6D pose and 3D bounding box of objects [10].

We use the NOCS 6D pose estimation model [10] as our baseline in this project. The NOCS model is based on the already existing MaskRCNN model. The MaskRCNN model [3] classifies and predicts instance maps for each object in the image. The NOCS model predicts an NOCS map (explained in Sec. 2.3) in parallel with the instance segmentation, which helps us in finding the orientation of the object. It has implemented a CNN model that predicts the class label, instance mask, and NOCS map, all of which can be treated as classification problems. In the instance map, each pixel is either given a 0 or 1 value depending on whether it belongs to the object or not. NOCS map prediction involves the prediction of 3 values for each pixel corresponding to the R,G,B colours. Each colour is discretized into 32 values and a pixel is classified into one of the 32 values. Since all predictions are classification problems simple loss functions like softmax are only required. The NOCS model also holds top positions in the leaderboards in contests for state-of-the-art pose estimation methods.

2.3 MaskRCNN model

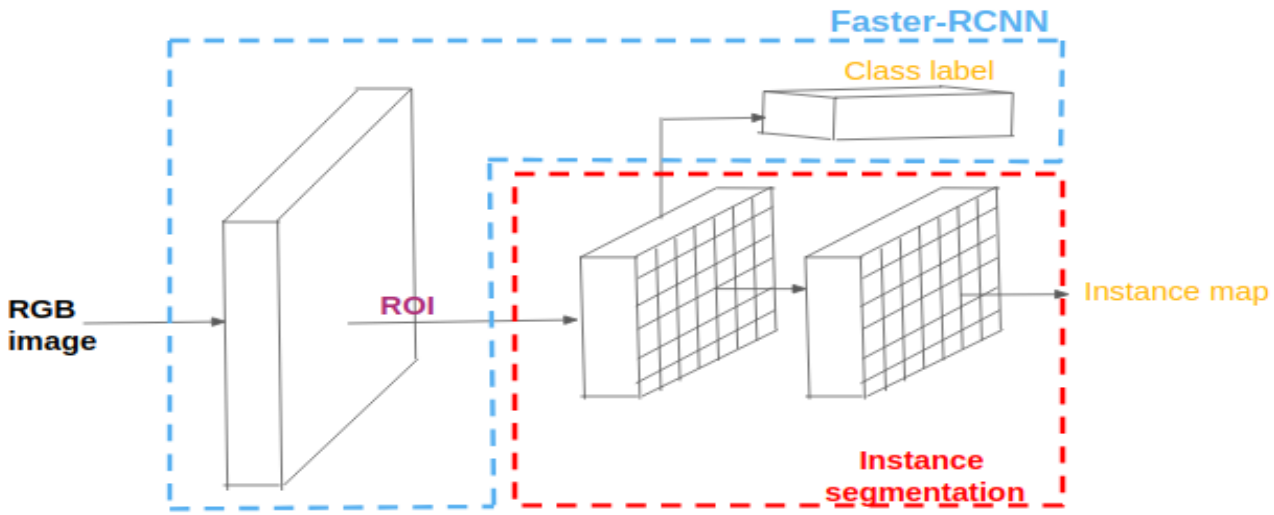


Fig. 2.5 The convolutional layers performing various functions in the MaskRCNN model. The MaskRCNN network adds an instance segmentation block to the Faster-RCNN network and replaced ROI pool operation with ROI align operation. The training time and costs aren't increased much since a parallel head is added.[2]

Faster-RCNN is a well-known state-of-the-art object detection model that predicts class and bounding box around each object in an image. The blue boundary in Fig. 2.4 shows the operation of Faster-RCNN. It has a Region Proposal Network(RPN) which uses anchor boxes for finding the optimal regional proposals or Region Of Interests(ROIs) and a block of fully connected(FC) layers that perform classification and bounding box regression. To the Faster-RCNN network,

[2]https://pythonawesome.com/a-pytorch-implementation-of-the-architecture-of-mask-rcnn/

Mask-RCNN adds a parallel head of mask prediction along with classification and bounding box regression. It also proposes the ROIAlign operation as an alternative to the conventional ROIPool operation used in Faster-RCNN. Pooling operation tends to break the pixel-to-pixel alignment due to quantized stride causing data loss. ROIAlign rectifies this giving us better spatial pixel-to-pixel alignment. Hence ROIAlign rectifies the small errors in bounding box alignment caused because of quantized strides in pooling.

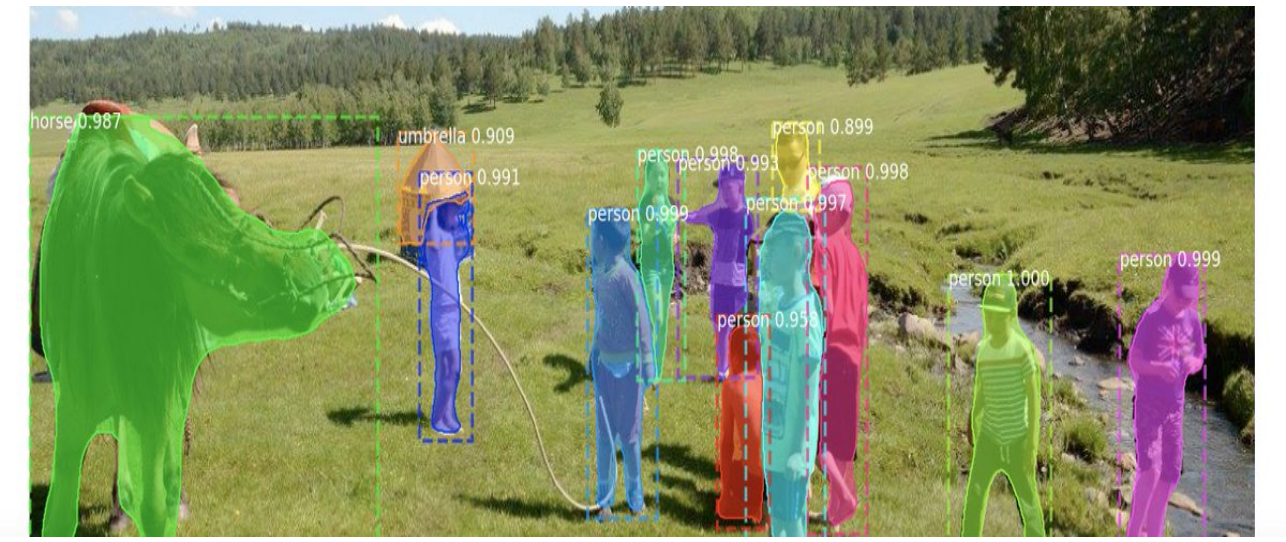


Fig. 2.6 The image shows the output of the MaskRCNN model depicting the classification, bounding box prediction and instance segmentation for each object in the image. It gives confidence values for each prediction and identifies occluded objects separately. [3]

2.4 Edge Agreement Loss

The edge agreement loss, inspired by [15], works towards the better and faster prediction of the instance masks. It is derived from the idea of how humans annotators notice the object boundaries first and then fill the area within to form the shape of the object. Therefore to our model, an auxiliary prediction head is added that predicts the edges of objects present in the image. An edge agreement loss is found out by taking the mean square error between ground truth and predicted edges. This loss function is added to the total mask loss that increases the importance of boundaries of the objects.

The Edge Agreement Loss, L_{Edge} of an image is calculated convoluting an edge loss filter over the image. There are various available edge loss filters such as the Sobel filter, Prewitt filter, Roberts filter, gaussian and laplacian filters, etc. The first three filters are 1^{st} order filters since the output of the convolution calculates the 1^{st} order derivates and the last two are 2^{nd} order filters since the output of their convolution calculates the 2^{nd} order derivates. For example, for the Sobel filter, we have a Sobel-x filter and a Sobel-y filter.

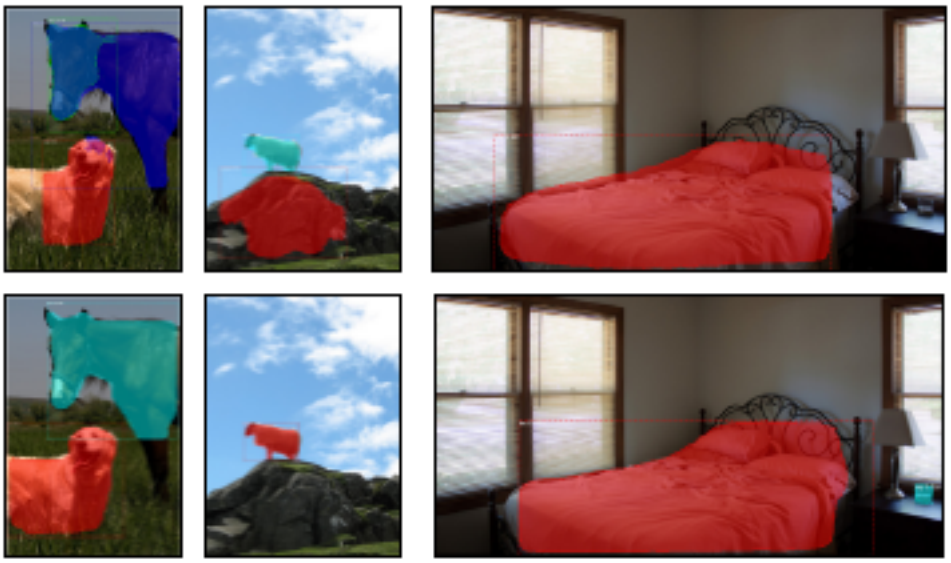


Fig. 2.7 This figure shows a comparison between the mask predictions by the baseline model vs Mask-RCNN + Edge Agreement Head after 160k epochs. We can observe the improvement in mask prediction with the addition of the edge loss. The The masks predicted have better coverage of the object, removes false positives, etc. because of better edge detection. [15]

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \text{ and } \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A} \text{ , where } \mathbf{A} \text{ is the image.}$$

The gradient magnitude at each point in the image is given by

$$G = \sqrt{G_x^2 + G_y^2} \tag{2.1}$$

and the gradient's direction is given by

$$\theta = \arctan \frac{G_y}{G_x} \tag{2.2}$$

$$\therefore L_{edge}^{p}(\tilde{y}, y) = Mean_p(|\tilde{y} - y|^p) \tag{2.3}$$

, where y is the ground truth edge image and is the predicted edge image. We have taken p=2, ie. mean square error in our research work since it gives higher weightage to larger errors. Higher powers aren't used since they might give more importance to outliers which can affect our fit.

So, the final mask loss function is given by,

$$L_{Mask}^{new} = L_{Mask} + L_{Edge} \quad [15] \tag{2.4}$$

2.5 Chapter Summary

The first section discusses the use of deep learning in robotics tasks like self-driving cars, robot navigation, assembly line work in factories, etc. Then we explore a bit on the deep learning methods used for pose estimation like fiducial markers, lookup tables, etc. and refer the research work of 2 papers that use the marker-based system for pose estimation. We then move on to computer vision-based pose estimation, Sec. 2.2, in which we discuss in detail the model we use for 6D pose estimation called the NOCS model. We provide details about the structure of the model and discuss the Mask-RCNN network that is the baseline of the NOCS model. Lastly, we elaborate on the working and usage of the Edge Agreement Loss function. We also discuss the edge detection filters that we have used, the Sobel filters, Gaussian filters, etc.

Chapter 3

Project idea and implementation

This chapter describes the requirements of our application based project, discusses the CNN model that we use, and how we implement our proposed methods in the model.

3.1 Requirements and specifications of this research work

As mentioned in Chapter 2, the research work is an application-based and hence comes with an already existing problem statement. Our research is based on the requirements of a half-humanoid, which is designed to imitate humans. It has an **Intel Real Sense camera** attached to its torso, which can extract RGB and depth images within the available field of view. The humanoid is supposed to identify various aspects of the objects in its view in order to perform crucial functions like picking up objects, placing them elsewhere and toggling switches, etc. Hence a 6D pose and bounding box estimation of its hand and other objects in its view are to be found using Computer Vision. With the help of the translations and orientations obtained from the 6D pose, the source-to-target navigation can be performed by computing the error between source and target can be procured dynamically.

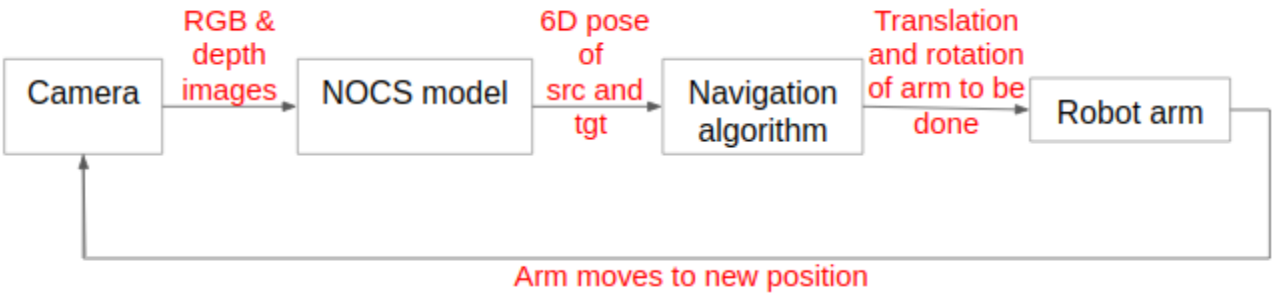


Fig. 3.1 A flowchart portraying the navigation of source(src), which in our case is the humanoid arm, to the target(tgt), which is the object. The Intel Real Sense camera outputs the RGB and depth images, then our NOCS model provides the 6D pose for both the src and tgt. The amount of translation and rotation to be done by the arm is calculated by a navigation algorithm. Then this process is repeated until the error between the 6D poses is less than a pre-decided threshold.

In Fig.3.1, we show the scheme we adopt, where we obtain the RGB and depth images from the camera. Then the 6D pose is estimated by the proposed model from the RGB and depth images. The 6D poses retrieved from the model are used to estimate the error in positions between the hand which is the source(src), and the object that is the target(tgt). This error is then used to compute the translation and rotation needed by the arm of the robot using some navigation algorithm. After the robot arm moves the calculated amount, the steps are repeated until we get and error below a decided threshold.

Technical Specifications of the model:

- Locating multiple objects(upto 5).
- Providing bounding box coordinates for each object.
- Classification and 6D pose(3 translation coordinates and 3 rotation angles) as output.
- Model is immune to different lighting conditions.
- For symmetric objects atleast two degrees of freedom is provided.
- Objects are assumed to stay within a range of 60cm from the camera.

3.2 CNN model used for pose estimation-NOCS

Normalized Object Coordinate Space(NOCS) is a shared canonical representation for all objects belonging to the same category [10] ie. all objects of the same class, for eg. a camera is assumed to be aligned in the same direction inside NOCS and there will have similar NOCS maps. In the case of a camera, the lens of any camera will have a red color, the left part of the camera will have a blue colour, the top will have green, etc. This shared representation proves highly advantageous when it comes to the training of the model.

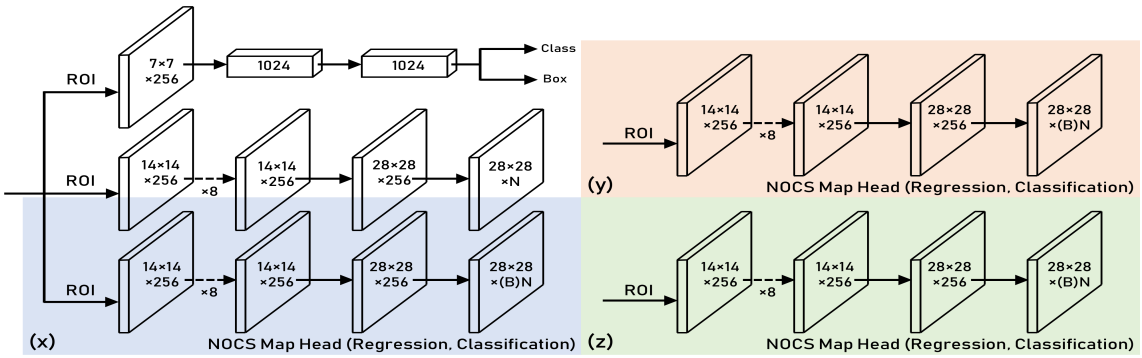


Fig. 3.2 Top left shows the convolutional block that carries out classification, bounding box prediction, and instance mask prediction. The blue, pink, and yellow parts show the convolutional blocks that predict (x,y,z) coordinates of the NOCS map respectively. ReLU activation and 3x3 convolutions have been used. [10]

As mention in Chapter 2, the NOCS model is an extension to the MaskRCNN network to predict an extra NOCS map. Convolutional blocks are added in parallel with mask prediction and classification networks for NOCS predictions(Fig.3.2).

The NOCS map of an object is essentially an RGB colour map corresponding to each pixel of the object. The colour assigned to each pixel is given by (r,g,b) = (x,y,z) of the object's 3D CAD model normalized within a unit cube.

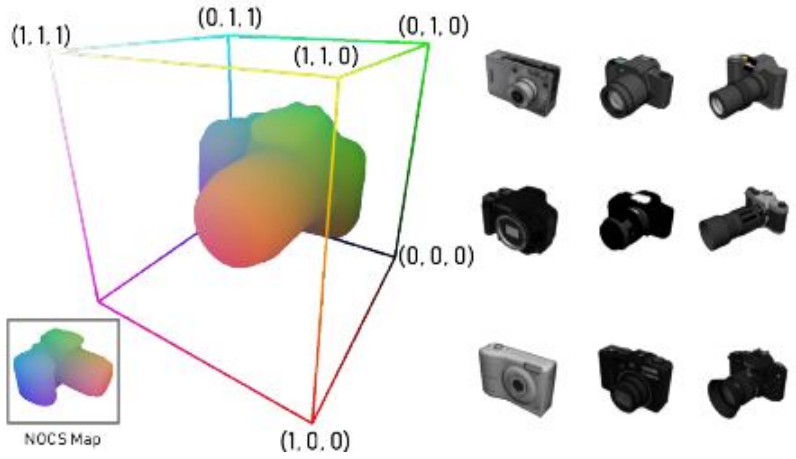


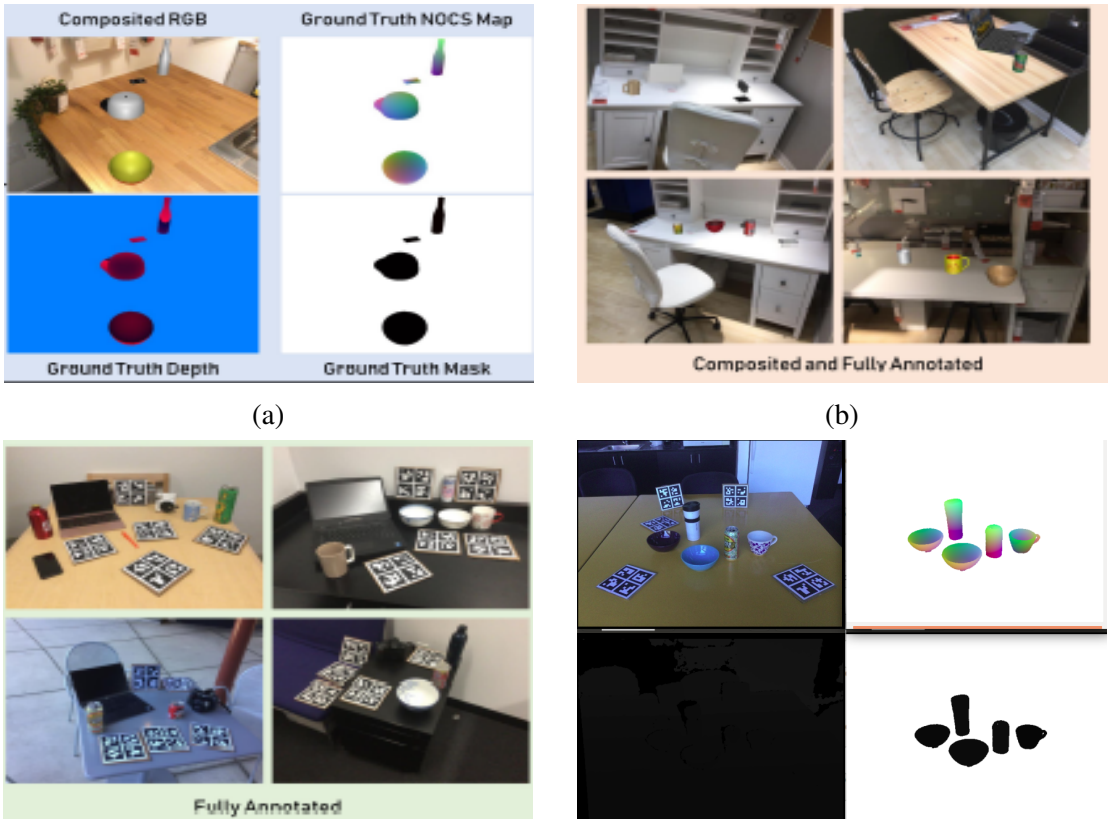
Fig. 3.3 The figure shows a 3D camera placed within a unit cube, colour coded according to each pixel's position within the cube. The image at the bottom left shows a 2D image of the ground-truth NOCS map.

The research work in [10] uses RGB-D (RGB image + Depth image) data to find out the 6D pose of objects. The NOCS model predicts the class, instance map, and the NOCS map of each object in the image. Two point clouds of the object are generated from the NOCS map and the instance mask+depth image. The translation and rotation calculated from the two point clouds using the Umeyama algorithm [9] gives the 6D pose for every object. However, the pose estimation in the research work [10] is dependent on the region proposals and category prediction, an error in which can lead to negative effects on the pose estimated.

3.2.1 Datasets used for training

1. **C**ontext **A**ware **M**ix**E**d **R**e**A**lity(CAMERA) [10]: To facilitate the generation of a large number of images for training and testing purposes, a combination of real background with synthetically rendered objects obtained from 3D CAD models was made in a 3D manipulation software called unity. A context-aware approach was used in the sense that images comprised of various lighting conditions, scale, and possible physical locations. As the name suggests, the mixed-reality approach helped in lower time-consumption and cost-effectiveness. A total of 275K training and 25K testing and 184 object instances for validation.

2. REAL dataset: A dataset containing real scenes was also used for training. All the images are tabletop scenes containing hand-scale objects from 31 widely varying indoor scenes [10]. A total of 553 images in 31 object instances in which 27 were taken for training and 4 for validation.



(a) (b)

(c) (d)

Fig. 3.4 (ab),(b): Context-aware mixed reality(CAMERA) composite images with real backgrounds and synthetic objects. (a) shows RGB images, ground truth NOCS map, depth map and instance mask for the same image.
(c),(d): Images belonging to the REAL dataset in various physical locations with (d) showing RGB image and corresponding ground truth maps generated [10].

3.2.2 Loss functions used in NOCS model

The total loss in the NOCS model is a multi-task loss which is a combination of all the loss functions used for various tasks like classification, bounding box prediction, instance mask, NOCS map prediction, etc. We will be discussing 5 different loss functions used for different tasks performed by the model.

1. **Classification:** The log loss function is used for the classification task. It is defined as the negative log of the true class probability. It takes values between 0 and 1 and the goal is to

minimize the log loss value [2].

$$L_{cls}(p,u) = -logp_u \tag{3.1}$$

, where u is the true class.

2. **Bounding box regression loss:** For bounding box regression, the smooth L1 loss is used, which is the robust L1 loss as it is less sensitive to outliers. Assume t=(x,y,w,h) is the target ground truth bounding box coordinates and p=(x,y,w,h) is the predicted bounding box coordinates where (x,y) is the coordinates of the centre of the bounding box and w and h are the corresponding width and height respectively [2].

$$L_{bbox}(t,p) = \sum_{i \in x,y,w,h} smooth_{L1}(p_i, t_i) \tag{3.2}$$

, where

$$smooth_{L1}(x) = \begin{cases} 0.5x^2, & \text{if } |x|<1 \\ |x| - 0.5, & \text{otherwise} \end{cases} \tag{3.3}$$

3. **Instance mask prediction:** The mask loss function used in the NOCS model the same as the one used for MaskRCNN, which is the average binary cross-entropy loss. The loss function is the average of pixel-wise binary cross-entropy loss applied to the pixels belonging to the ground truth mask. This is commonly used in classification problems where the distance between two probability distributions is minimized [3].

$$L_{mask} = -\frac{1}{m^2} \sum_{1 \leq i,j \leq m} [y_{ij} log \hat{y}^k_{ij} + (1 - y_{ij}) log(1 - \hat{y}^k_{ij})] \tag{3.4}$$

, where y_{ij} and $\hat{y}_{ij}$ are the ground truth and predicted pixel values respectively, m is the pixels belonging to the mask and k represents the true class.

4. **NOCS map:** The NOCS map prediction can be treated either as a classification or a regression problem. In the case of classification, the softmax loss function is used and in case of regression, the smooth L1 loss is used.

$$L_{softmax}(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \tag{3.5}$$

, where i is the index for the true class.

5. **Object Symmetry:** NOCS representation doesn't take into account the symmetry in the objects, like a bottle, which has one axis of symmetry. Therefore, a loss function that is independent of the rotation about the axis of symmetry is introduced. Six ground

truth NOCS maps are found out by rotating the object about its axis of symmetry by six different angles. The new loss is defined as the minimum of six losses found by taking the losses between predicted and six ground truths.

$$L_{symm} = min_{i=1,\ldots,|\theta|} L(\hat{y}_i, y^*) \tag{3.6}$$

, where θ is the angle of rotation and $|\theta|$ is the number of angles. $\hat{y}_i$ is the ground truth NOCS map for i^{th} angle of rotation and y^* is the predicted NOCS map. $|\theta| \leq 6$ was found to be enough for most symmetric objects.

3.3 Research work and implementation

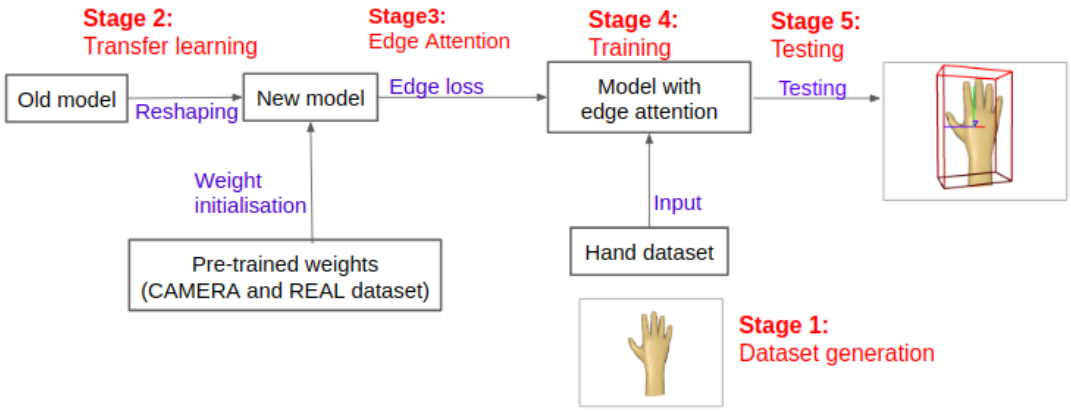


Fig. 3.5 A flowchart of the process followed during our research split into 5 stages starting from hand dataset generation, to transfer learning for weight initialization in the model, including edge attention loss and then training of the model using the created hand dataset.

3.3.1 Dataset generation

The requirement for our project was to identify and compute the 6D pose of human hands in different orientations and lightings. We create a training and validation dataset similar to the CAMERA Dataset in [10].
Contents of the Hand Dataset:

- RGB image
- Depth image
- Ground truth instance map
- Ground truth NOCS map
- A text document consisting of class labels of objects.
- A text file containing dimensions of each object's CAD model.

Software and method used for dataset creation

We use a very popular 3D creation software called Blender for creating the images.

- **RGB images:** We import each hand CAD model into blender and colour it to a random human skin colour and render images from different camera views for obtaining different orientations of the RGB image.
- **Instance masks:** Instance masks are obtained by replacing the skin coloured pixels in RGB images by black colour.
- **NOCS maps:** For creating NOCS maps, we first place the model in a unit box with axis R,G,B colours. The centroid of the model is aligned with the center of the cube and the size of the model is normalized such that the diagonal of its tight 3d bounding box, equals 1 unit. The RGB colour (r,g,b) of each point on the model is equal to (x,y,z) respectively, corresponding to its position in the cube. NOCS maps are now rendered from the coloured model from the same orientations that RGB images were rendered.
- **Depth images:** 16-bit depth images are rendered using a depth utility available in blender.

Using the above-mentioned method we have generated 1900 training, 300 validation, and 300 testing images varying in orientation, lighting, and scale. The Hand Dataset created along with codes is available on this Github page: https://github.com/AnjuVolt/EdgeAttention_6Dpose_estimation A few examples from the training dataset have been shown below along with images from the REAL dataset for comparison.

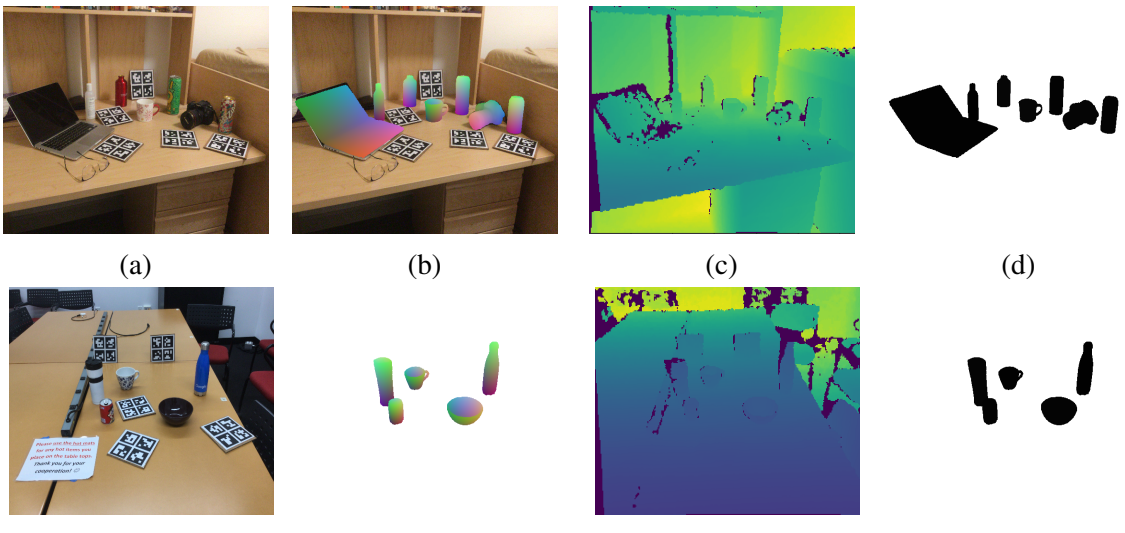


Fig. 3.6 $1^{st}, 2^{nd}, 3^{rd} and 4^{th}$ columns show sample RGB image, NOCS map, depth image and instance mask respectively from REAL dataset showing a set of 7 table top hand-scale objects.[10]. The depth images shown include the alpha channel.

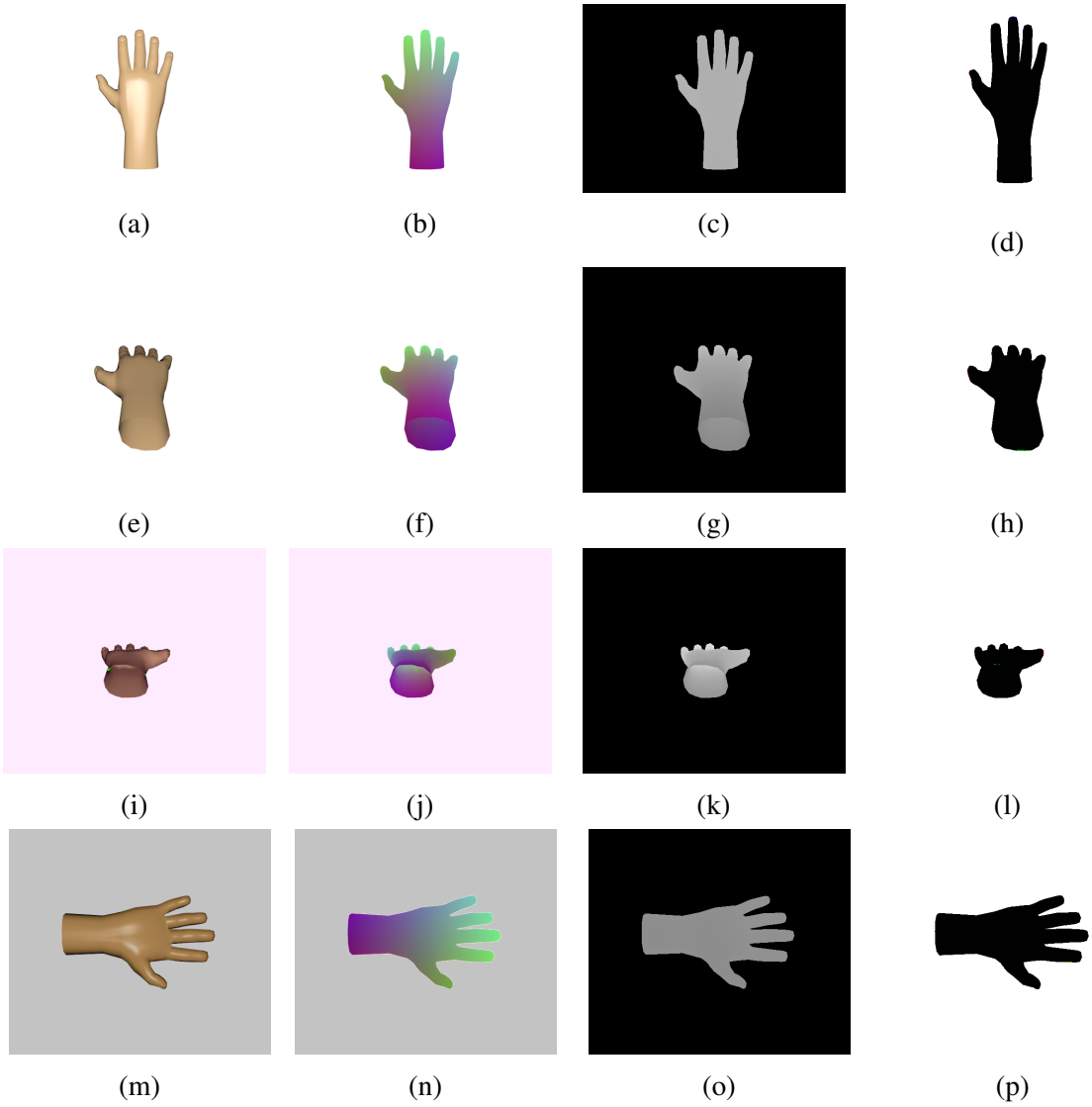


Fig. 3.7 The first two rows show RGB image, NOCS map, Depth map, and instance mask images from our hand dataset in two different hand orientations. The last two rows show the same four sets of images in two different lighting conditions, 3^{rd} row-image has a red undertone, 4^{th} row-illumination is lower which is simulated by decreasing the pixel values if the image.

3.3.2 Transfer Learning

To train a pre-trained model with a new class while still retaining the information about the already trained objects, we use a technique called **transfer learning**. Transfer learning is a machine learning method where a model trained for one task is re-purposed and used on a second task. This technique can also be applied in scenarios where we need to train the model on an additional class while still keeping the information about the previously trained classes.

To implement this, we design a new model in which we make alterations to the head layers alone and retain the backbone of the original model. The top layers of the new model, performing the end tasks like classification, bounding box regression, instance mask, and NOCS map prediction will have to predict

their respective values for one more class. And hence, there will be an increase in the number of weights and biases that need to be trained.

1. Therefore the transfer learning method we follow involves initializing the backbone layers of the new model with the pre-trained weights and biases(**wb**s).

2. Then for the head layers, the **wb** values corresponding to already trained 7 classes are initialized with the pre-trained **wb** values and the extra **wb**s added due to prediction for an extra class, we initialize them with the average of the first 7 pre-trained **wb**s.

Shown below are the changes made to each head architecture and corresponding weights and bias initializations before training for the inclusion of one additional class.

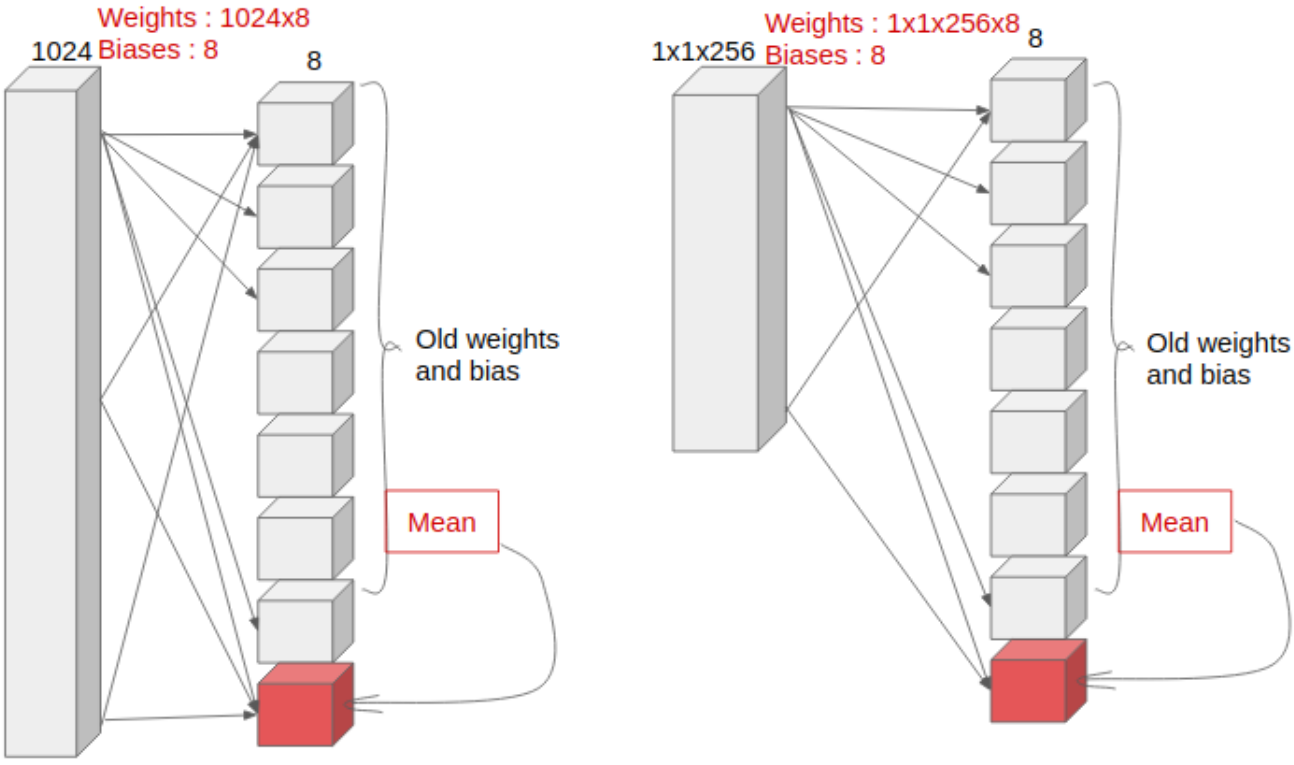


Fig. 3.8 The figure on the left shows the last two layers of the classification head and right shows the last two layers of the mask prediction head of the new model. The red box is the additional classification and mask prediction for the 8^{th} class. The new set of weights and biases are initialized with the mean of the weights and biases for the rest of the classes.

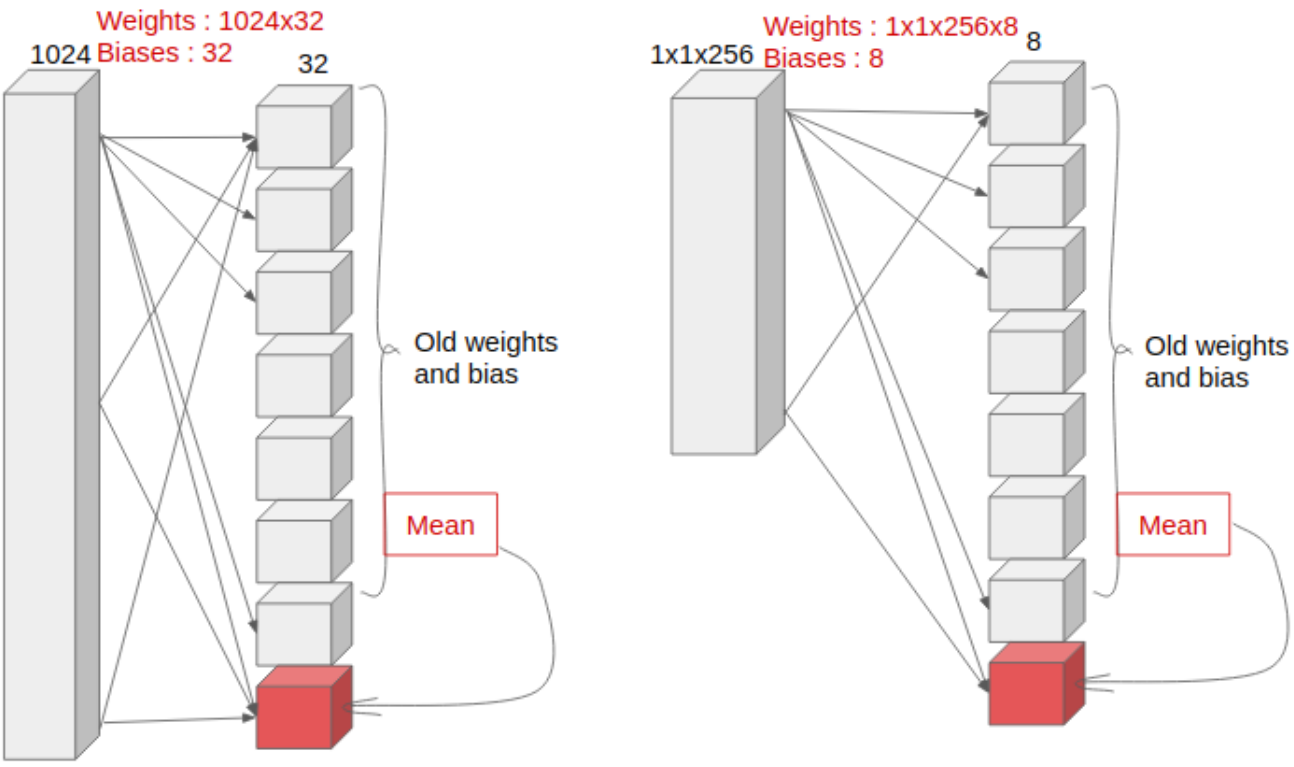


Fig. 3.9 The figure on the left shows the last two layers of the bounding box prediction head and right shows the last two layers of the NOCS prediction head(same for r,g, and b value prediction) of the new model. The red box is the additional classification and mask prediction for the 8^{th} class. The new set of weights and biases are initialized with the mean of the weights and biases for the rest of the classes.

3.3.3 Training

Model hyper-parameters used while training

The NOCS model can be divided into network backbone and head layers. The backbone layers perform the feature extraction tasks, whereas the head layers perform the ultimate tasks of classification, bounding box regression, instance mask prediction, and NOCS map prediction. Since the feature extraction tasks remain independent of the class the object belongs to, our training for the inclusion of one extra class prediction would only require training the head layers.

Therefore, after initializing the model with the pre-trained weights as mentioned in Sec. 3.3.2, we train the head layers alone for 70 epochs, at a learning rate of 0.001, weight decay of 0.0001, and 1000 steps per epoch.

The network is trained on a dataset made with alternating hand images and images from the pre-trained dataset so that the final trained network will not be biased towards the hand dataset.

3.4 Chapter Summary

The chapter starts by specifying the requirements of our project to find the 6D pose of the humanoid hand from the images taken using an Intel Real Sense camera attached to the humanoid's chest. We then mention the characteristics of the model we want to develop such as multiple object detection in various lighting conditions, in conditions of occlusion, etc. Then we explain the Normalised Object Coordinate Space(NOCS) and how our model uses this in addition to the instance map and depth image, to estimate the 6D pose of the objects. We explain the datasets called the CAMERA and REAL datasets that were used in the pre-trained model. The loss functions that are used in the prediction of class, instance mask, NOCS map and the loss function used for symmetric objects are elaborated.

Sec. 3.3 discusses the process we follow during the course of our research which comprises of 5 stages, **1.** Hand dataset generation, **2.** Weight initialization by transfer learning from pre-trained to the new model, **3.** Addition of Edge Agreement Loss to the model, **4.** Training of the model on the hand dataset and **5.** Testing of the model on various hand images. The succeeding subchapters discuss the execution of each stage in detail.

Chapter 4

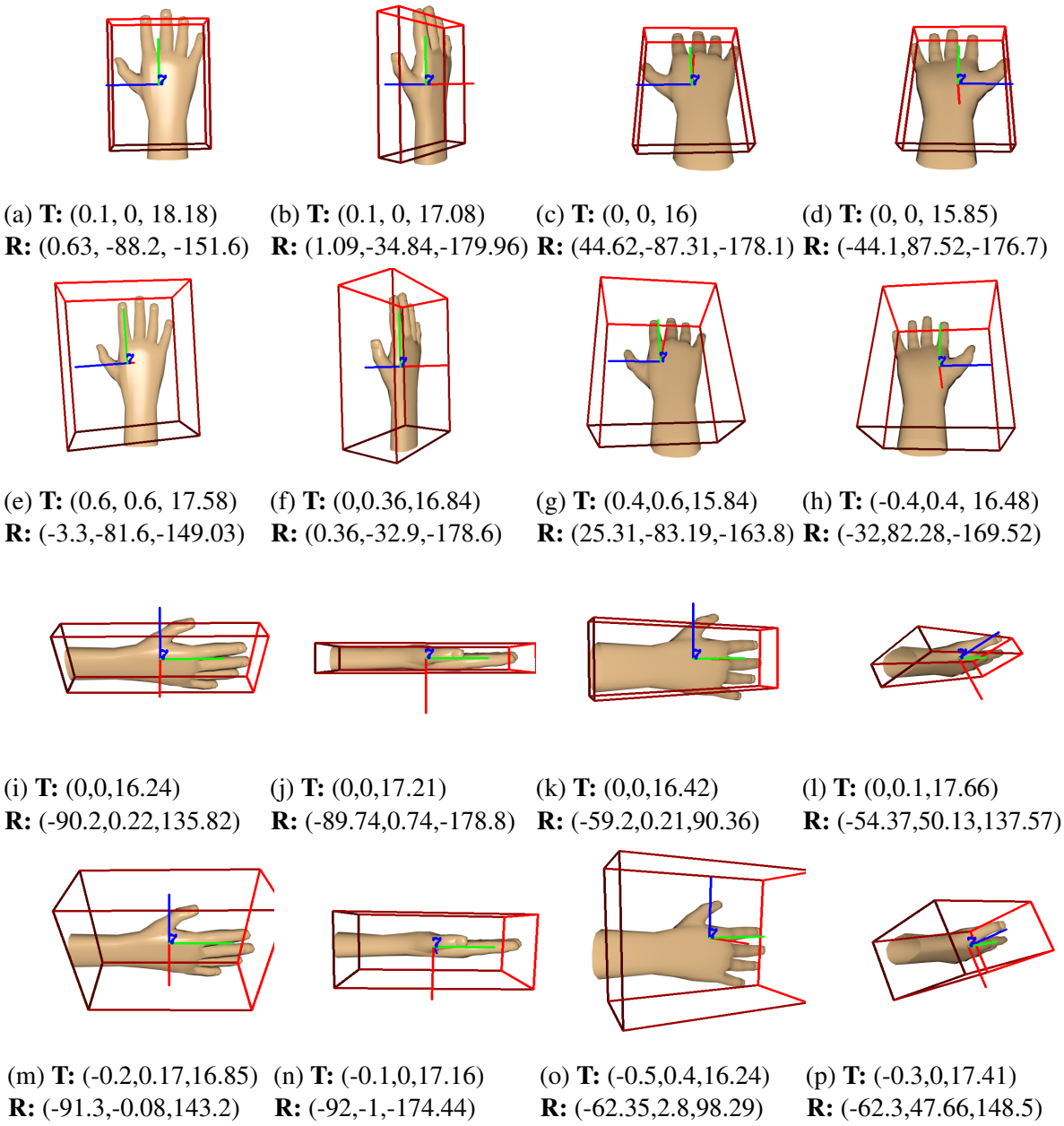
Results and Discussion

We create a new model by replacing the head layers of the NOCS model with an additional prediction class for the recognition of our humanoid hand. We initialize this model with the pre-trained weights and biases for the 6 classes(mug, laptop, can, bottle, bowl and camera) trained on the CAMERA and the REAL dataset. The model can be divided into network backbone and head layers. The backbone layers perform the feature extraction tasks, whereas the head layers perform the ultimate tasks of classification, bounding box regression, instance mask prediction, and NOCS map prediction. The head layers alone are trained on the hand dataset for 70 epochs since all the backbone layers perform feature extraciton,which is independent of the class. A learning rate of 0.001, weight decay of 0.0001, and 1000 steps per epoch are used. The training takes 1min 5secs/epoch in the NVIDIA Tesla V100 system. Testing takes about 5.39s per image.

4.1 Testing results of hand dataset

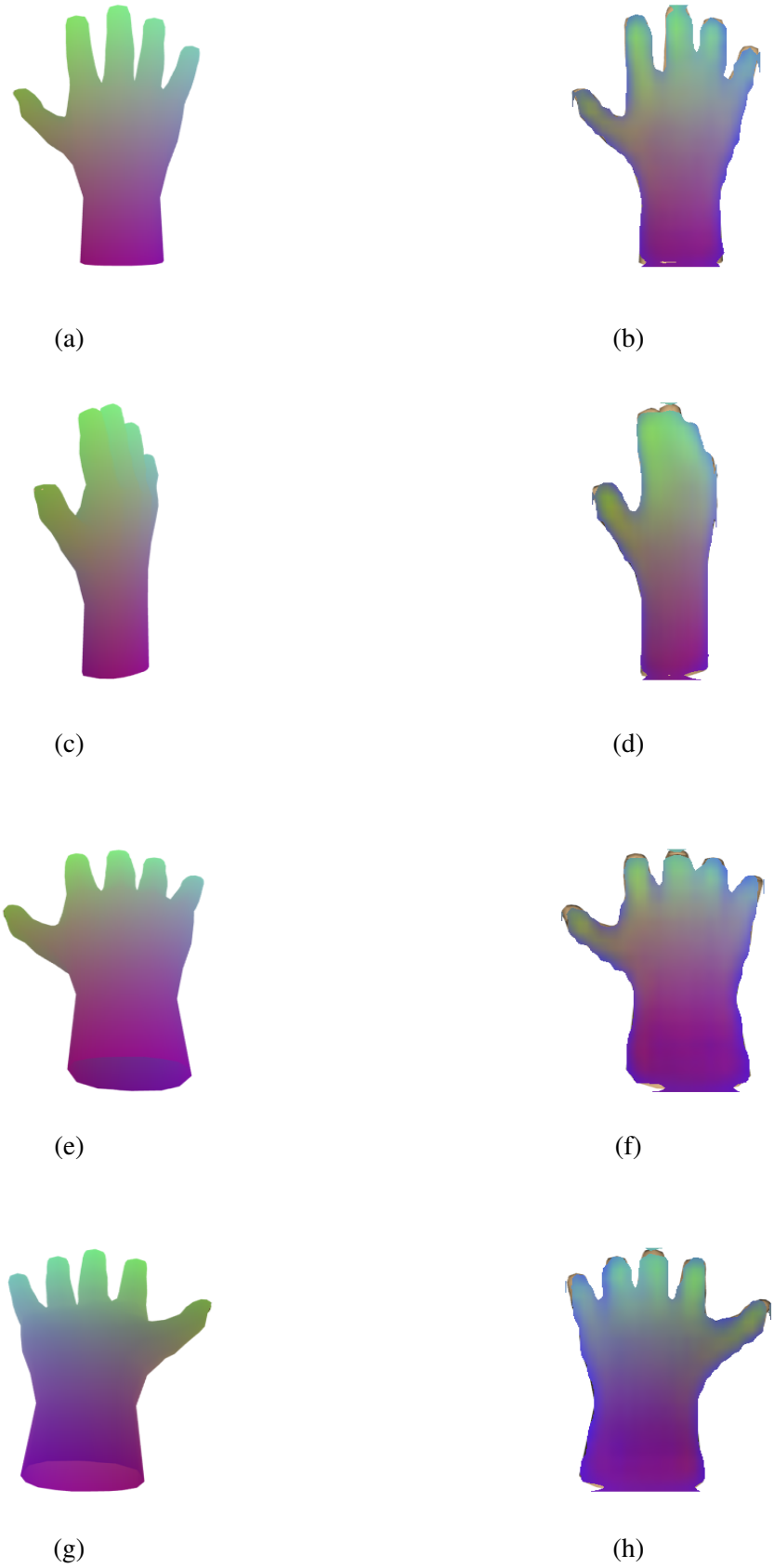
This chapter manifests the testing results for different variations of objects varying in orientation, scale, and lighting. The ground truth and predicted 6D poses and NOCS maps have been plotted for different orientations, lighting intensities and distance of the hand from the camera. The translation and rotation errors obtained from the model's prediction are plotted and compared. Certain anomalies that occur in the images are discussed for every section. The testing results for the edge loss have been portrayed along with the various mask images before and after the addition of edge loss. We also discuss a common error that has been found in the NOCS map prediction where the edges of the maps predicted are erratic. We propose an edge attention loss for the NOCS map, similar to that used for mask prediction. This would improve the prediction at the edges of the map and hence increase the 6D pose estimation accuracy even further.

4.1.1 Varying orientations



(a) **T:** (0.1, 0, 18.18) **R:** (0.63, -88.2, -151.6)
(b) **T:** (0.1, 0, 17.08) **R:** (1.09,-34.84,-179.96)
(c) **T:** (0, 0, 16) **R:** (44.62,-87.31,-178.1)
(d) **T:** (0, 0, 15.85) **R:** (-44.1,87.52,-176.7)
(e) **T:** (0.6, 0.6, 17.58) **R:** (-3.3,-81.6,-149.03)
(f) **T:** (0,0.36,16.84) **R:** (0.36,-32.9,-178.6)
(g) **T:** (0.4,0.6,15.84) **R:** (25.31,-83.19,-163.8)
(h) **T:** (-0.4,0.4, 16.48) **R:** (-32,82.28,-169.52)
(i) **T:** (0,0,16.24) **R:** (-90.2,0.22,135.82)
(j) **T:** (0,0,17.21) **R:** (-89.74,0.74,-178.8)
(k) **T:** (0,0,16.42) **R:** (-59.2,0.21,90.36)
(l) **T:** (0,0.1,17.66) **R:** (-54.37,50.13,137.57)
(m) **T:** (-0.2,0.17,16.85) **R:** (-91.3,-0.08,143.2)
(n) **T:** (-0.1,0,17.16) **R:** (-92,-1,-174.44)
(o) **T:** (-0.5,0.4,16.24) **R:** (-62.35,2.8,98.29)
(p) **T:** (-0.3,0,17.41) **R:** (-62.3,47.66,148.5)

Fig. 4.1 $1^{st}, 3^{rd}$ rows-ground truth 6D pose of hand images, $2^{nd}, 4^{th}$-predicted 6D pose of hand images. The translation(cm) and rotation(deg) triplets have been mentioned as captions for each image. B,G,R(x,y,z) lines in the centre whose origin is the centroid of the hand, and its orientation is in the orientation of the hand. In the middle of each image is displayed the class label. We observe a translation error <1cm and rotation error <8°.



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

Fig. 4.2 Col-1: Ground truth NOCS map, Col-2: Predicted NOCS map.
Shown above are the ground truth and predicted NOCS maps of hand in 4 different orientations corresponding to (a),(b),(c),(d) hands Fig. 4.2 The more is the error in the NOCS map, more is the error in the estimated pose.

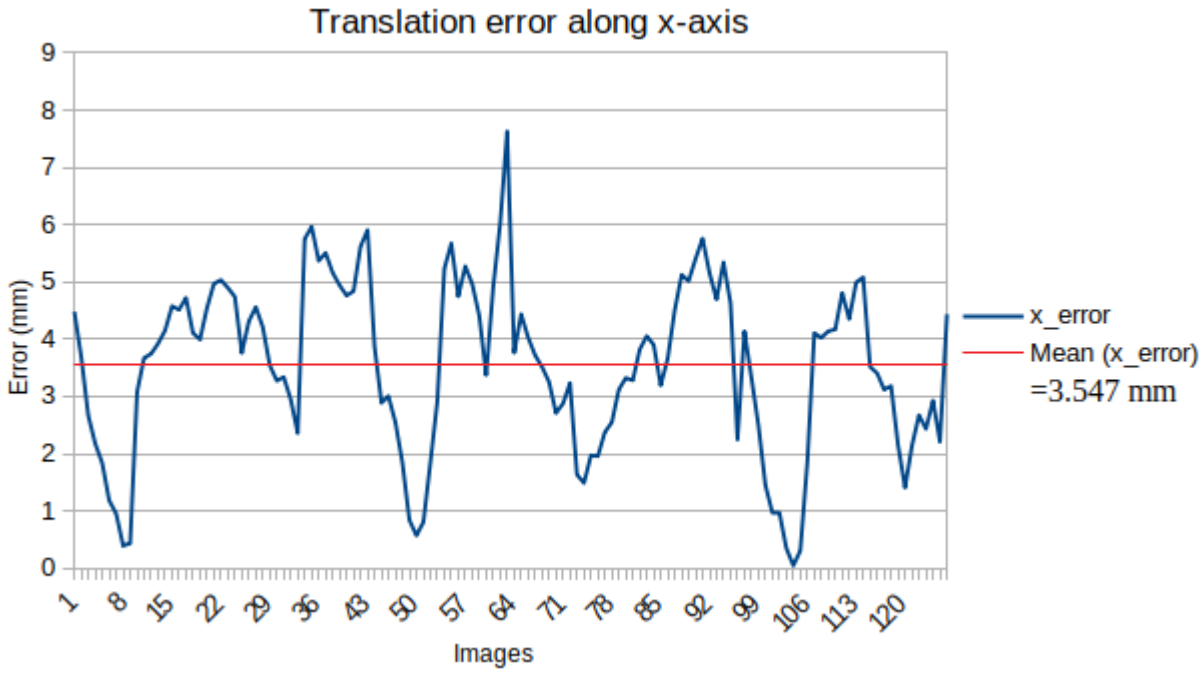


Fig. 4.3 This graph plots Translation error along the x-axis(Blue axis) in mms vs 125 testing images. The minimum loss is found to be 0mm, the maximum loss is 7.63mm, and the mean loss is 3.547mm which is indicated by the red line in the graph.

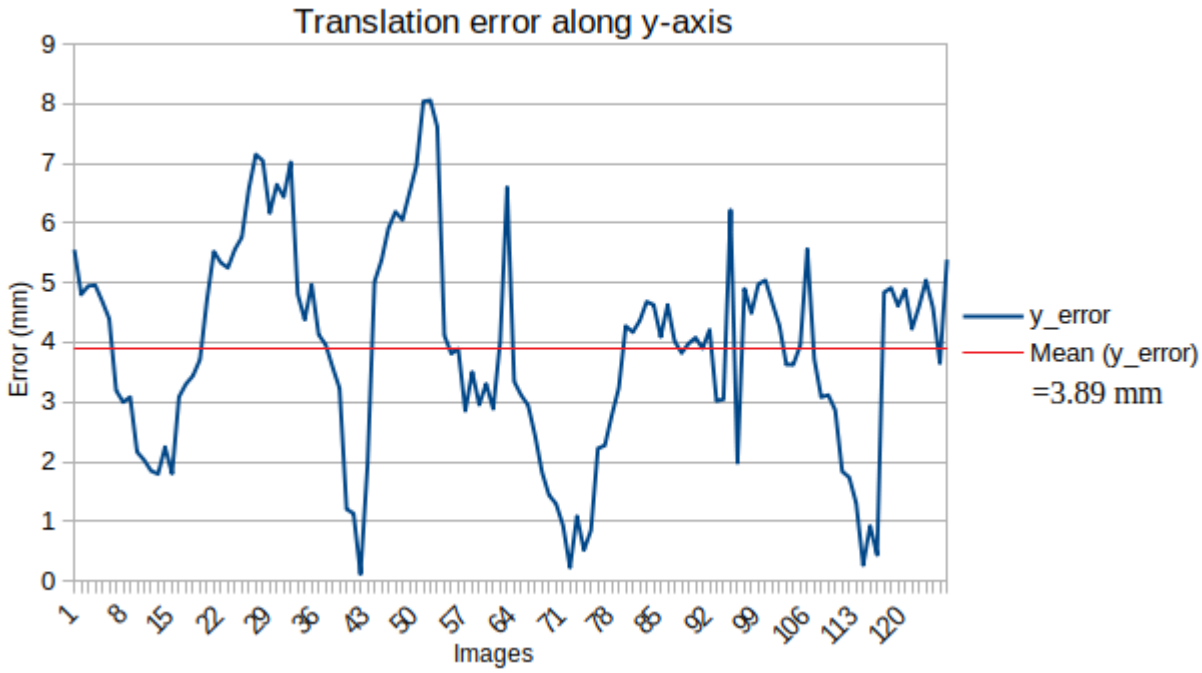


Fig. 4.4 This graph plots Translation error along the y-axis(Green axis) in mms vs 125 testing images. The minimum loss is found to be 0.1mm, the maximum loss is 8mm, and the mean loss is 3.89mm which is indicated by the red line in the graph.

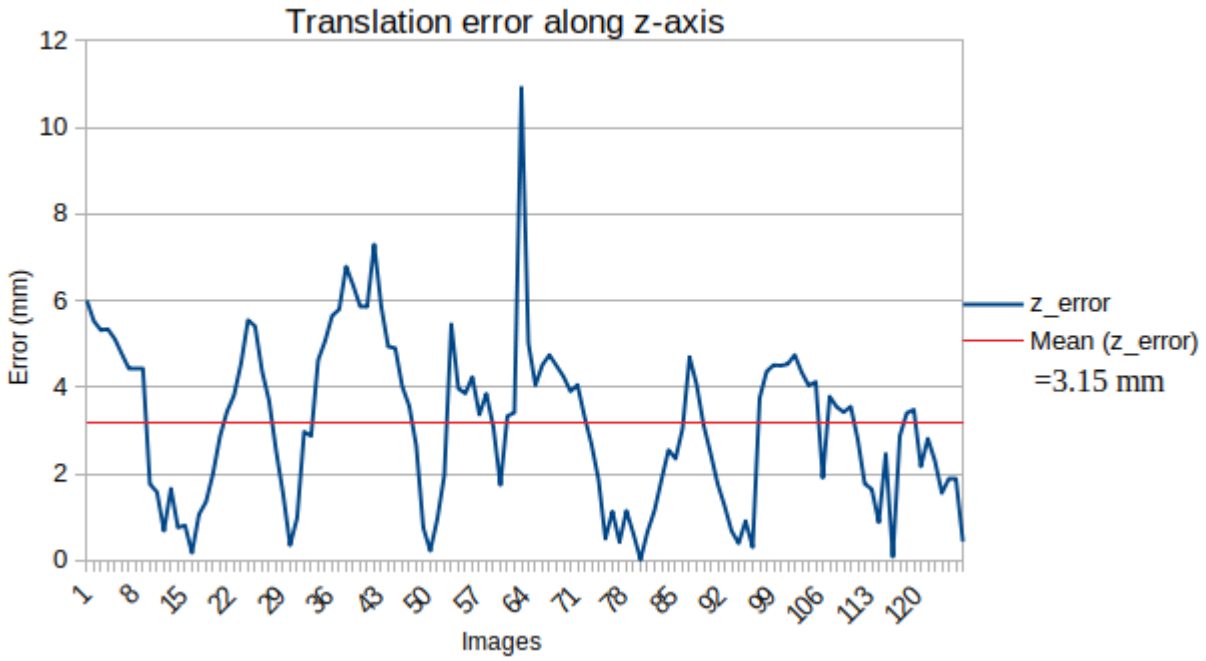


Fig. 4.5 This graph plots Translation error along the z-axis(Red axis) in mms vs 125 testing images. The minimum loss is found to be 0mm, the maximum loss is 10.9mm and the mean loss is 3.15mm which is indicated by the red line in the graph.

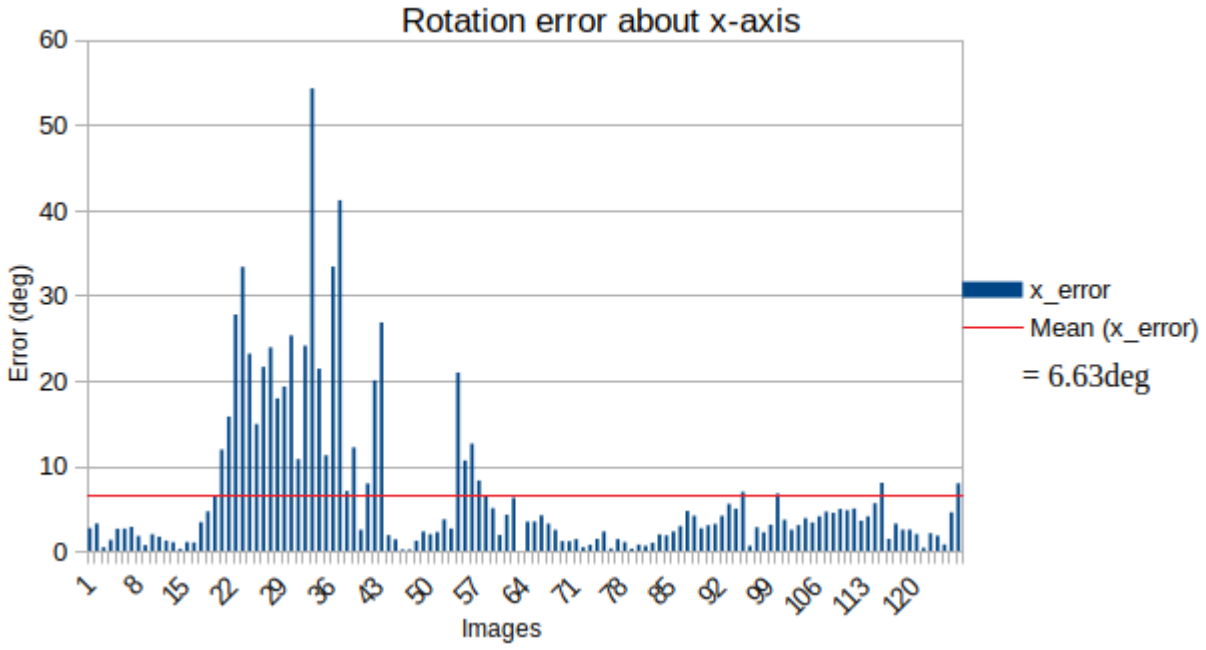


Fig. 4.6 This graph plots Rotation error along the x-axis(Blue axis) in degrees vs 125 testing images. The minimum loss is found to be 0.1°, the maximum loss is 54.3°, and the mean loss is 6.63°which is indicated by the red line in the graph.

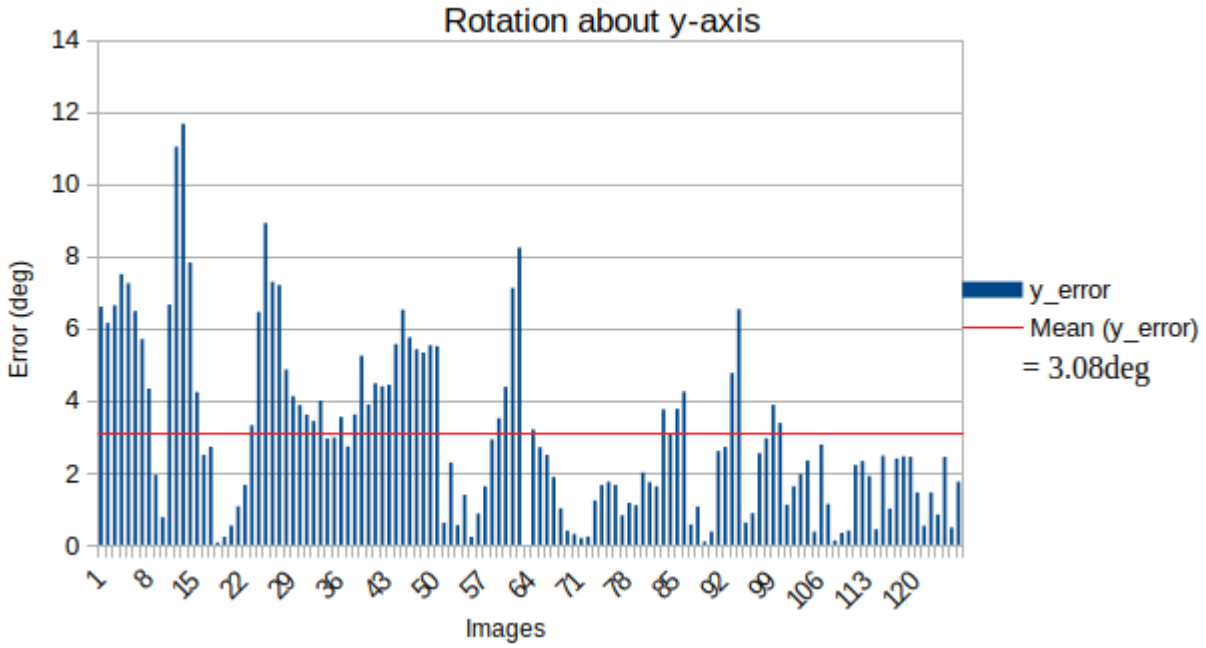


Fig. 4.7 This graph plots Rotation error along the y-axis(Green axis) in degrees vs 125 testing images. The minimum loss is found to be 0°, the maximum loss is 11.6°and the mean loss is 3.08°which is indicated by the red line in the graph.

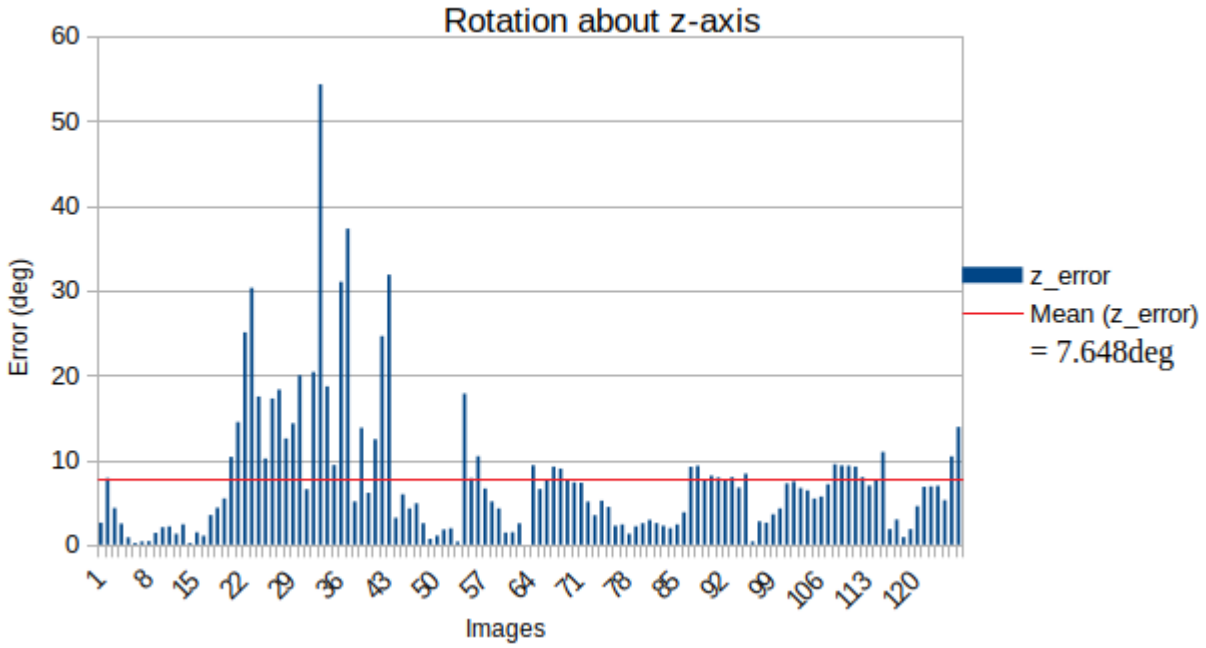
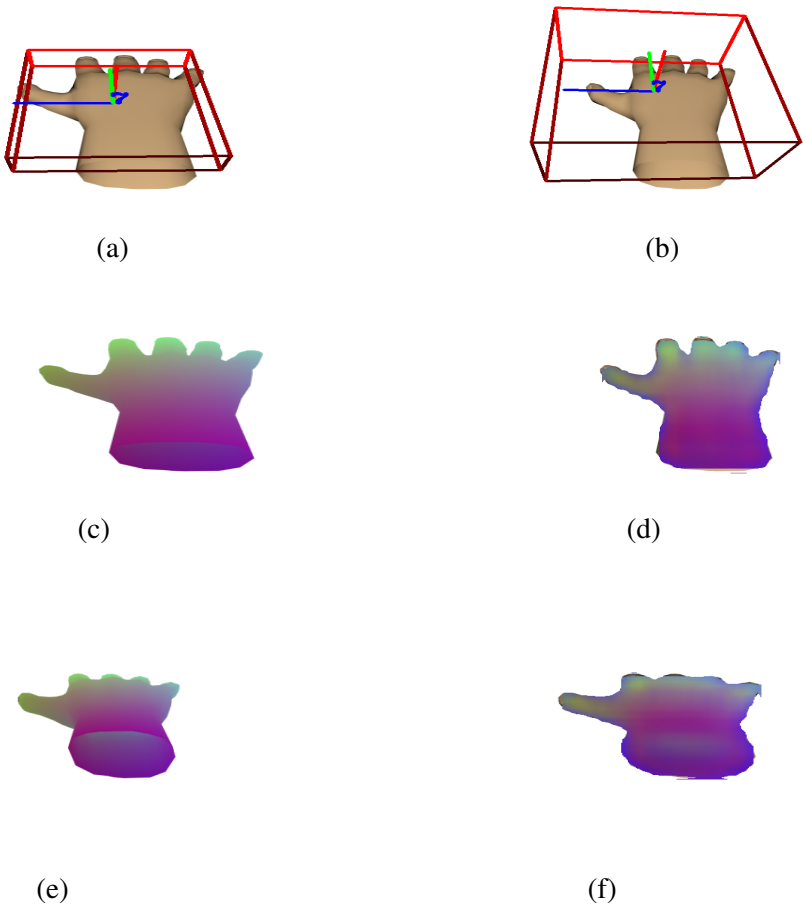


Fig. 4.8 This graph plots Rotation error along the z-axis(Red axis) in degrees vs 125 testing images. The minimum loss is found to be 0°, the maximum loss is 54.2°and the mean loss is 7.64°which is indicated by the red line in the graph.

Anomalies

We can observe from the graphs that the rotation errors about x and z axes are higher for the image numbers in the range of 25-32. Shown below are the ground truth and predicted 6D poses of these images and their corresponding NOCS maps. When the hand is in this particular orientation, we find that the NOCS map that needs to be found has a smaller area. We also observe a common error in all of the predicted NOCS maps, which is that the edge regions of the NOCS maps have incorrect predictions in all of the hand images(discussed in Sec.4.1.4). Since the NOCS maps in this orientation have only the edge portions(only the tips of the fingers), where we obtain incorrect predictions, the rotation errors are also larger in this section.



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

Fig. 4.9 (a),(b)-Image 29 ground truth and predicted 6D poses. (c),(d)-Image 29 ground truth and predicted NOCS maps. (e),(f)- Image 32 ground truth and predicted NOCS maps. With a closer look at the predicted NOCS maps, the prediction errors at the edges of the fingers can be observed. The ground truth has a predominant green colour while the predicted map has blue components at the same position. Since the NOCS maps contain the orientation information, errors in them will affect the estimated 6D poses.

For the evaluation of the predicted bounding boxes, we estimate a metric called Intersection over Union(IoU) between the ground truth and predicted bounding boxes. While a value of IoU=1 is considered ideal, a value ≥ 0.5 is considered to be a good prediction.

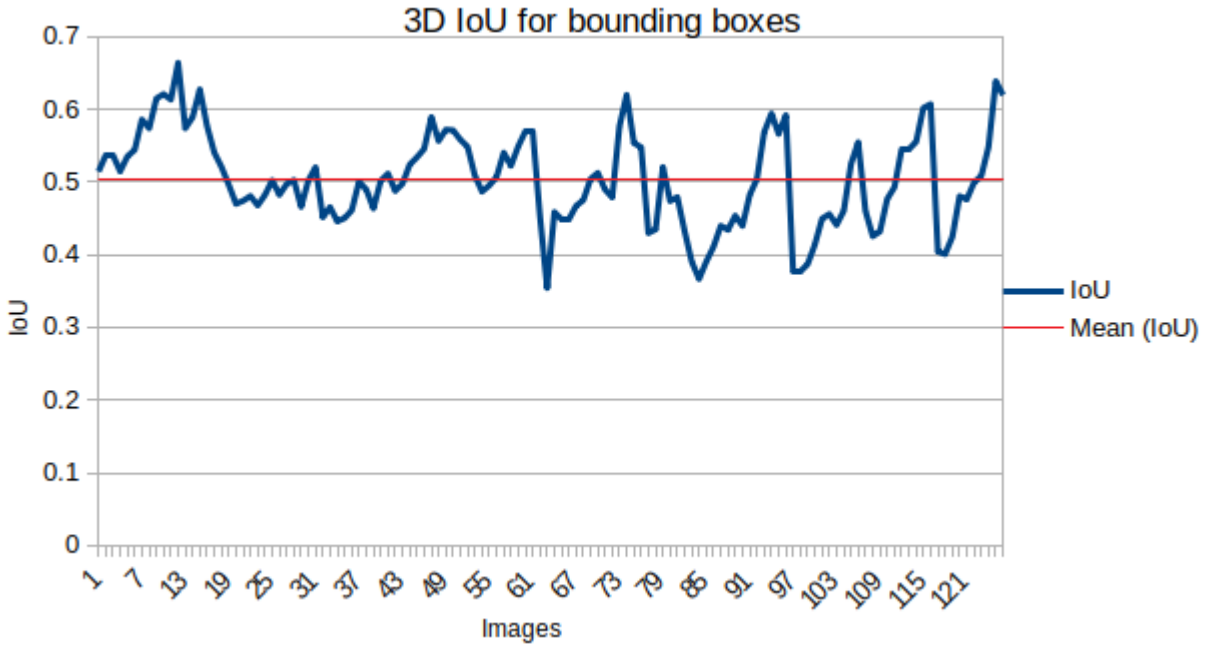
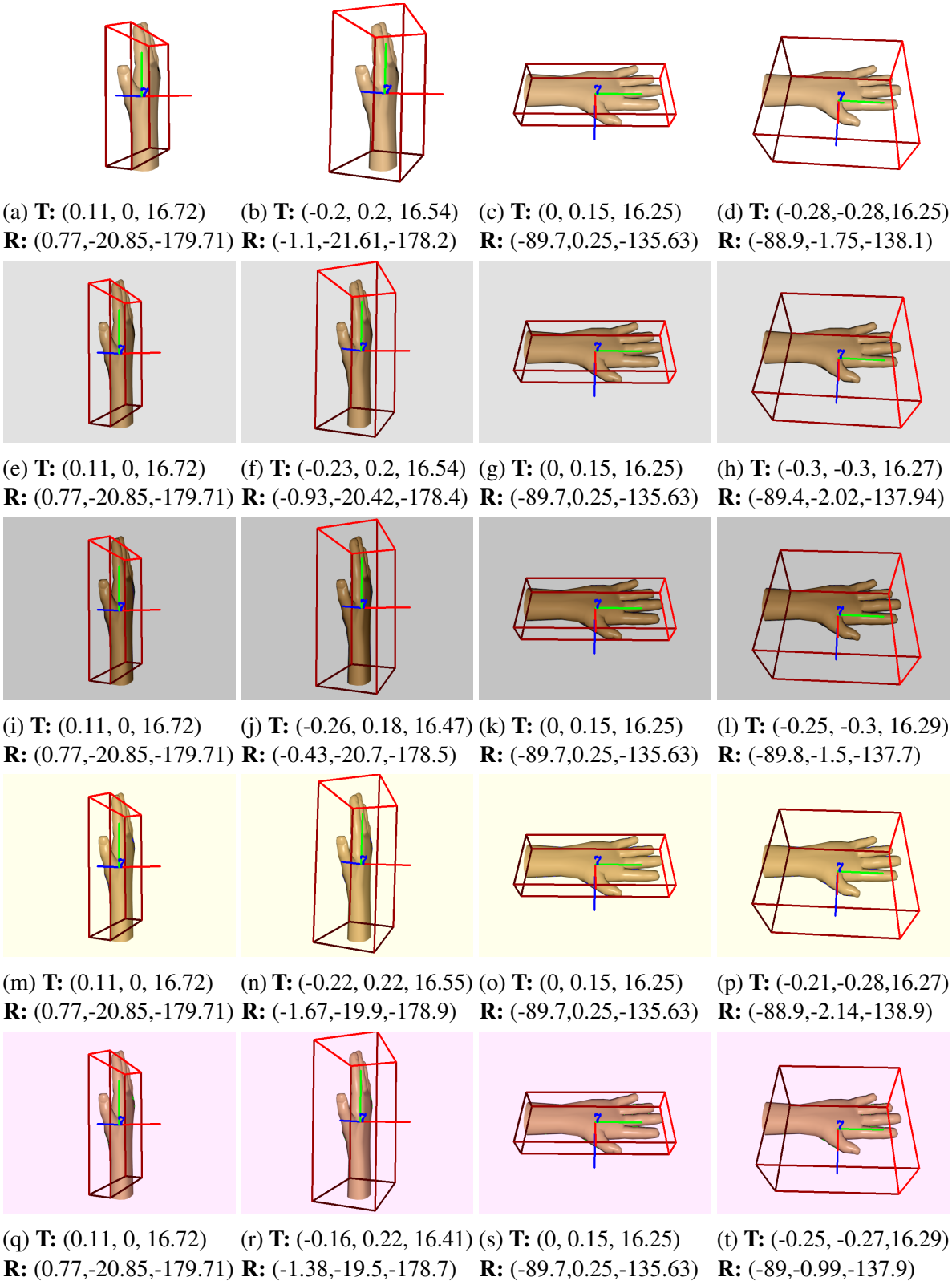


Fig. 4.10 The figure shows 3D IoU between predicted 3D ground truth and predicted bounding boxes vs testing images. We obtain a mean IoU of 0.5, which is considered to be an acceptable prediction for bounding boxes.

4.1.2 Varying lighting



(a) **T:** (0.11, 0, 16.72) **R:** (0.77,-20.85,-179.71)
(b) **T:** (-0.2, 0.2, 16.54) **R:** (-1.1,-21.61,-178.2)
(c) **T:** (0, 0.15, 16.25) **R:** (-89.7,0.25,-135.63)
(d) **T:** (-0.28,-0.28,16.25) **R:** (-88.9,-1.75,-138.1)

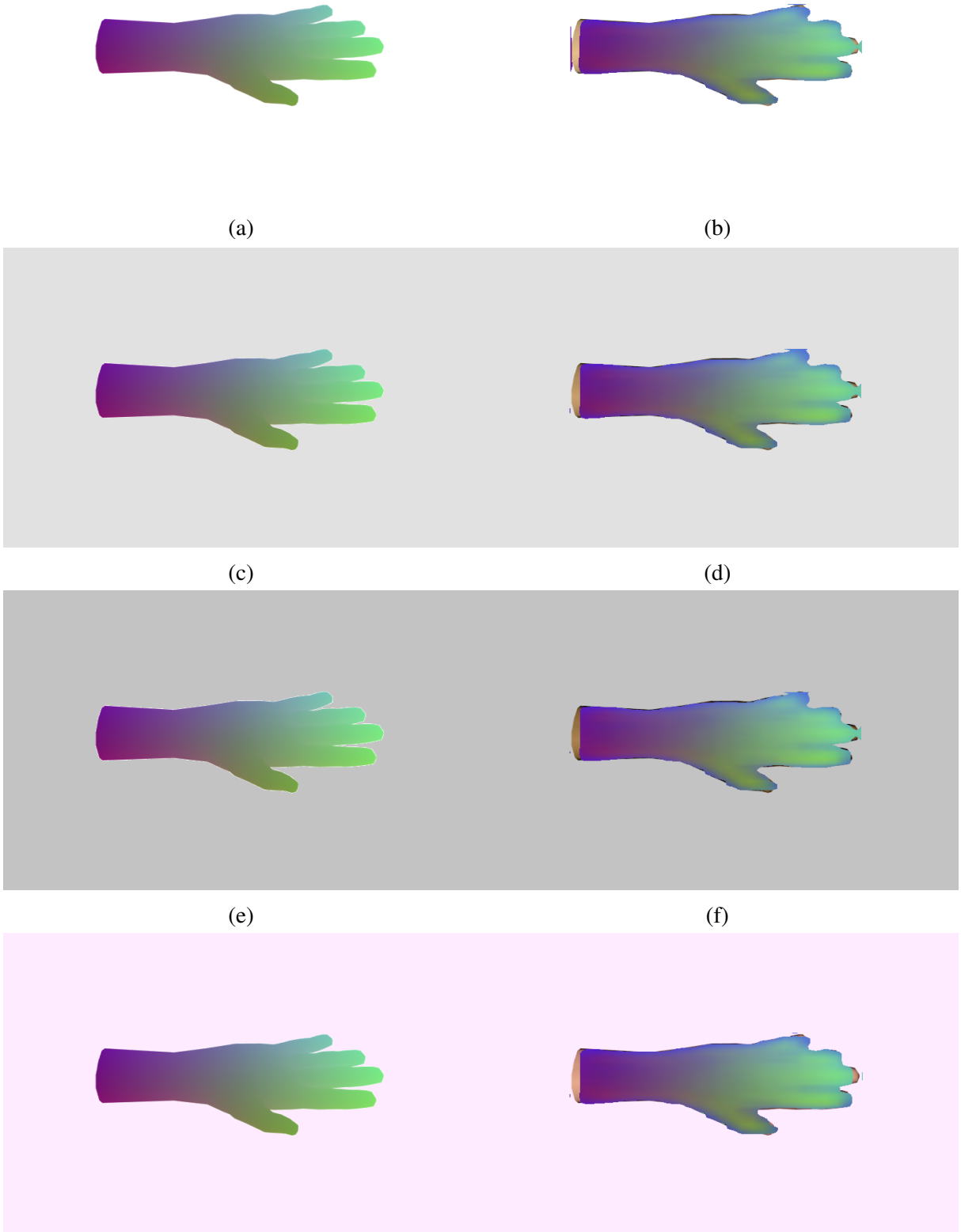
(e) **T:** (0.11, 0, 16.72) **R:** (0.77,-20.85,-179.71)
(f) **T:** (-0.23, 0.2, 16.54) **R:** (-0.93,-20.42,-178.4)
(g) **T:** (0, 0.15, 16.25) **R:** (-89.7,0.25,-135.63)
(h) **T:** (-0.3, -0.3, 16.27) **R:** (-89.4,-2.02,-137.94)

(i) **T:** (0.11, 0, 16.72) **R:** (0.77,-20.85,-179.71)
(j) **T:** (-0.26, 0.18, 16.47) **R:** (-0.43,-20.7,-178.5)
(k) **T:** (0, 0.15, 16.25) **R:** (-89.7,0.25,-135.63)
(l) **T:** (-0.25, -0.3, 16.29) **R:** (-89.8,-1.5,-137.7)

(m) **T:** (0.11, 0, 16.72) **R:** (0.77,-20.85,-179.71)
(n) **T:** (-0.22, 0.22, 16.55) **R:** (-1.67,-19.9,-178.9)
(o) **T:** (0, 0.15, 16.25) **R:** (-89.7,0.25,-135.63)
(p) **T:** (-0.21,-0.28,16.27) **R:** (-88.9,-2.14,-138.9)

(q) **T:** (0.11, 0, 16.72) **R:** (0.77,-20.85,-179.71)
(r) **T:** (-0.16, 0.22, 16.41) **R:** (-1.38,-19.5,-178.7)
(s) **T:** (0, 0.15, 16.25) **R:** (-89.7,0.25,-135.63)
(t) **T:** (-0.25, -0.27,16.29) **R:** (-89,-0.99,-137.9)

Fig. 4.11 The plots above show ground truth(gt) and predicted(pr) 6D pose of the same hand in 5 different lighting setups(1,3 cols-gt;2,4 cols-pr). The camera centered translation(**T** in cms) and rotation(**R** in degrees) triplets have been mentioned in the captions for each image. Similar to Fig.4.1, The red boxes indicate the bounding boxes, the RGB axes give an idea of translation and rotation of the hand.
The pose predictions are very similar for all lighting cases. We can observe < 2deg and < 0.1mm deviation in rotation and translation among hand images in different lightings.



(a) (b)

(c) (d)

(e) (f)

(g) (h)

Fig. 4.12 Col-1: Ground truth NOCS map, Col-2: Predicted NOCS map.
Shown above are the ground truth and predicted NOCS maps of hand in 4 different lighting conditions corresponding to (c),(g),(k),(s) hands Fig. 4.6. The more is the error in the NOCS map, more is the error in estimated pose.

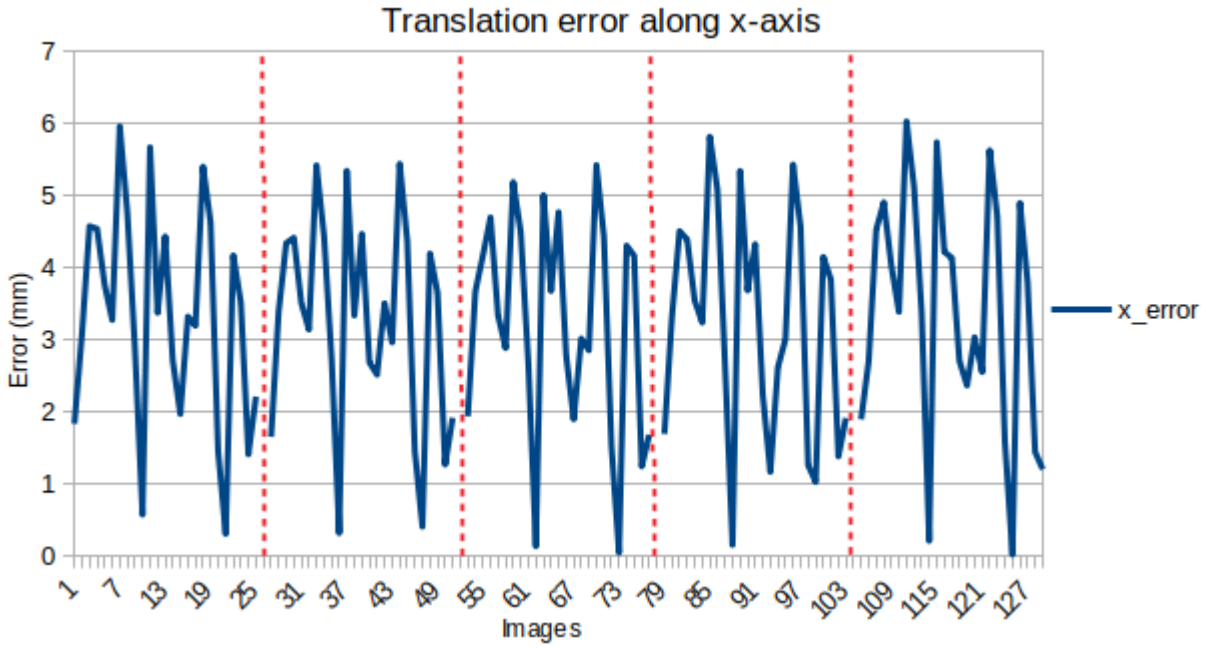


Fig. 4.13 This graph plots Translation error along the x-axis(Blue axis) in mms vs 125 testing images. The 4 dotted red lines divide the plot into 5 sections of different lighting setups, each plotting errors for 25 images. We can observe very less deviation in errors(<1mm deviation) for the same hand images kept in various lighting setups.

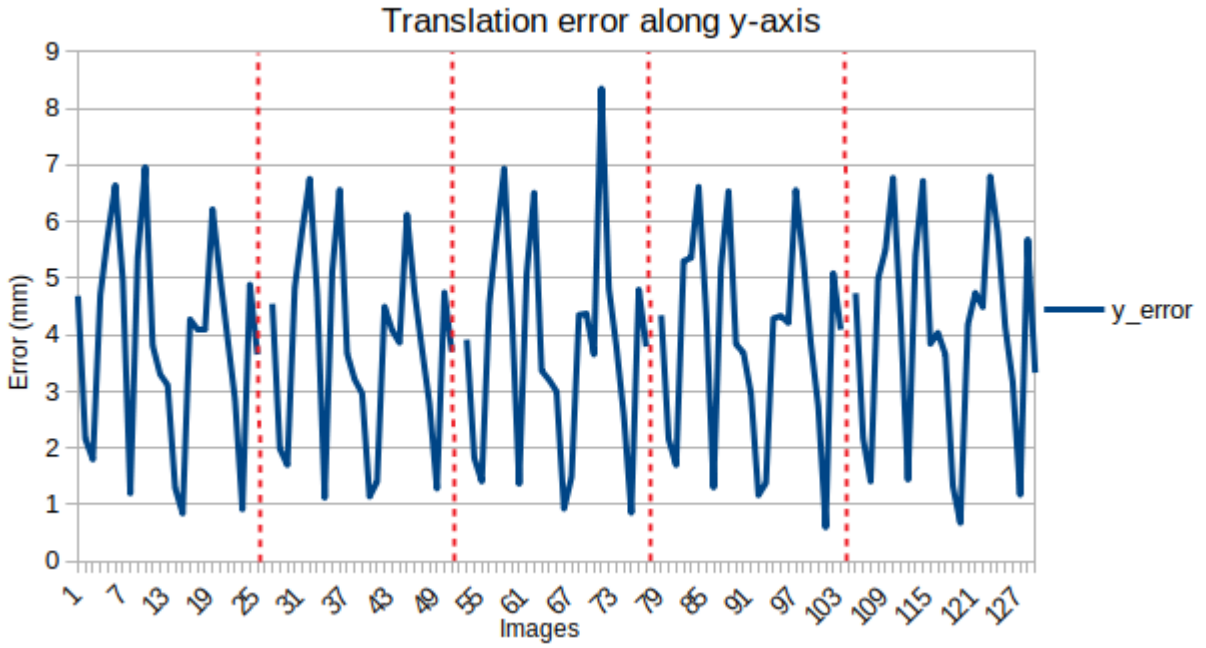


Fig. 4.14 This graph plots Translation error along the y-axis(Green axis) in mms vs 125 testing images. The 4 dotted red lines divide the plot into 5 sections of different lighting setups, each plotting errors for 25 images. We can observe very less deviation in errors(1mm deviation) for the same hand images kept in various lighting setups.

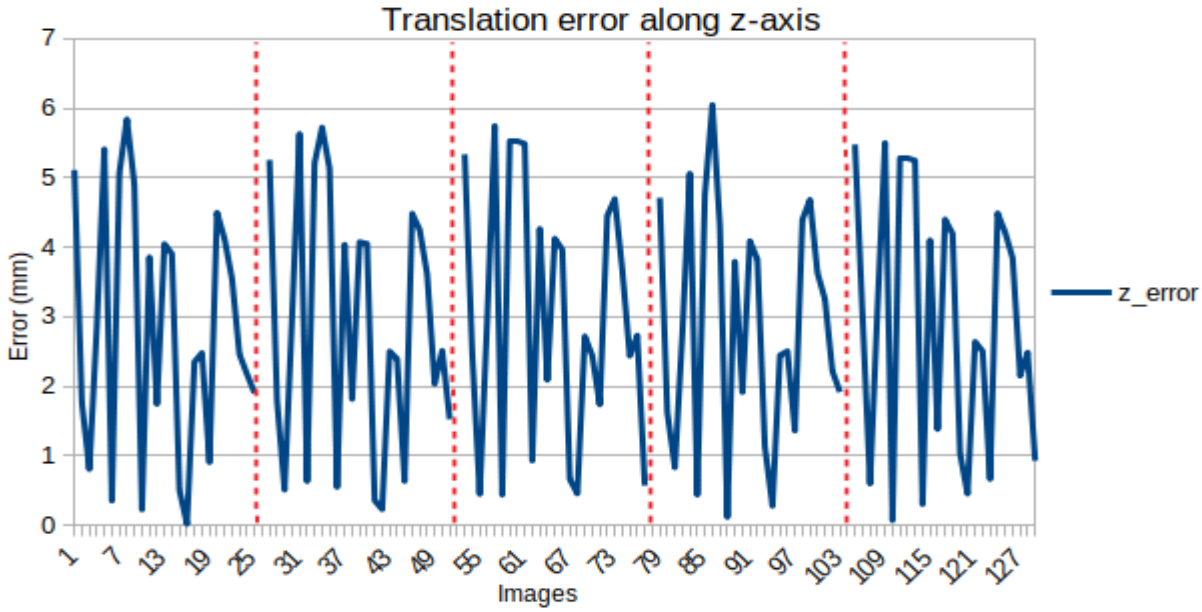


Fig. 4.15 These pictures demonstrate the error in the prediction of rotation about the x-axis(blue) when the x-axis is parallel to the camera view. We observe an error of about 40 deg in these two orientations as opposed to the mean error of 4.2deg. The predicted bounding boxes can be seen with a tilt about the x-axis with respect to the ground truth bounding boxes.

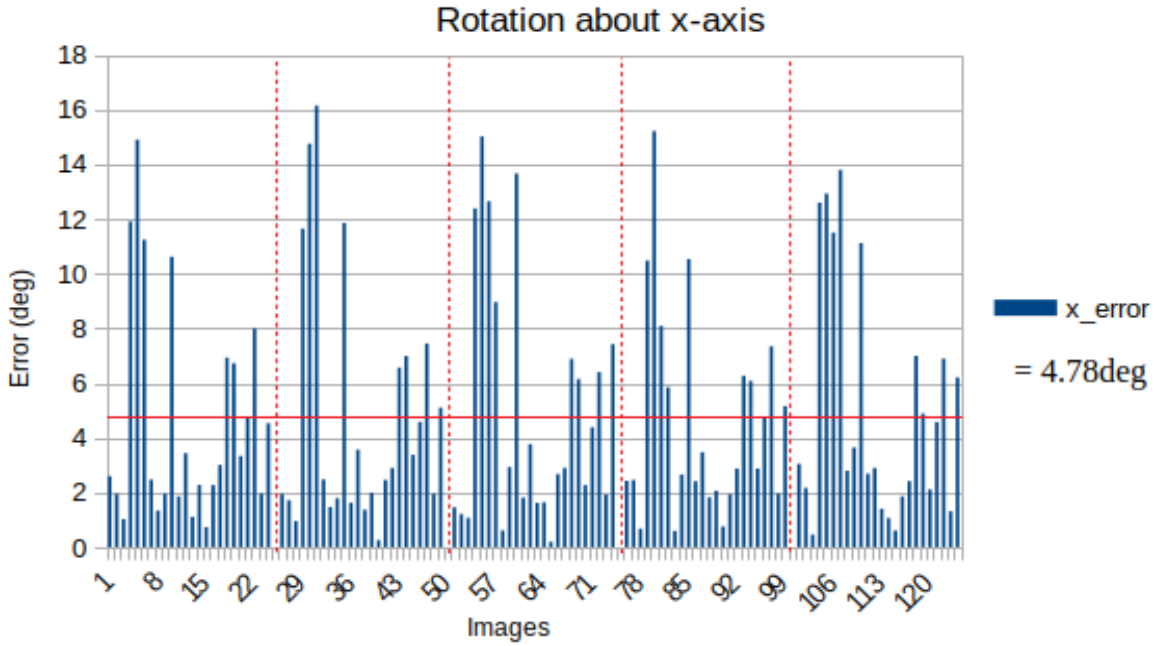


Fig. 4.16 This graph plots rotation error about the x-axis(Blue axis) in degrees vs 125 testing images. The 4 dotted red lines divide the plot into 5 sections of different lighting setups, each plotting errors for 25 images. Mean error for a dataset containing images in varying lighting = 4.78°. We can observe very little deviation in errors($\leq 1°$deviation) for the same hand images kept in various lighting setups.

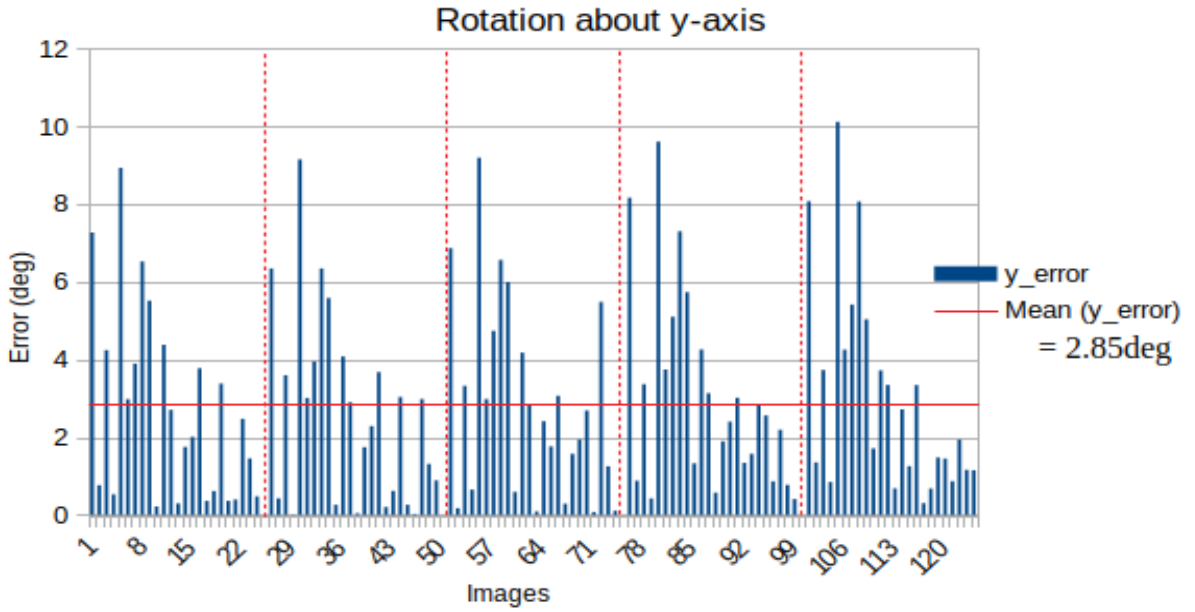


Fig. 4.17 This graph plots rotation error about the y-axis(Green axis) in degrees vs 125 testing images. The 4 dotted red lines divide the plot into 5 sections of different lighting setups, each plotting errors for 25 images. Mean error for a dataset containing images in varying lighting = 2.85°. We can observe very little deviation in errors($\leq$ 1°deviation) for the same hand images kept in various lighting setups.

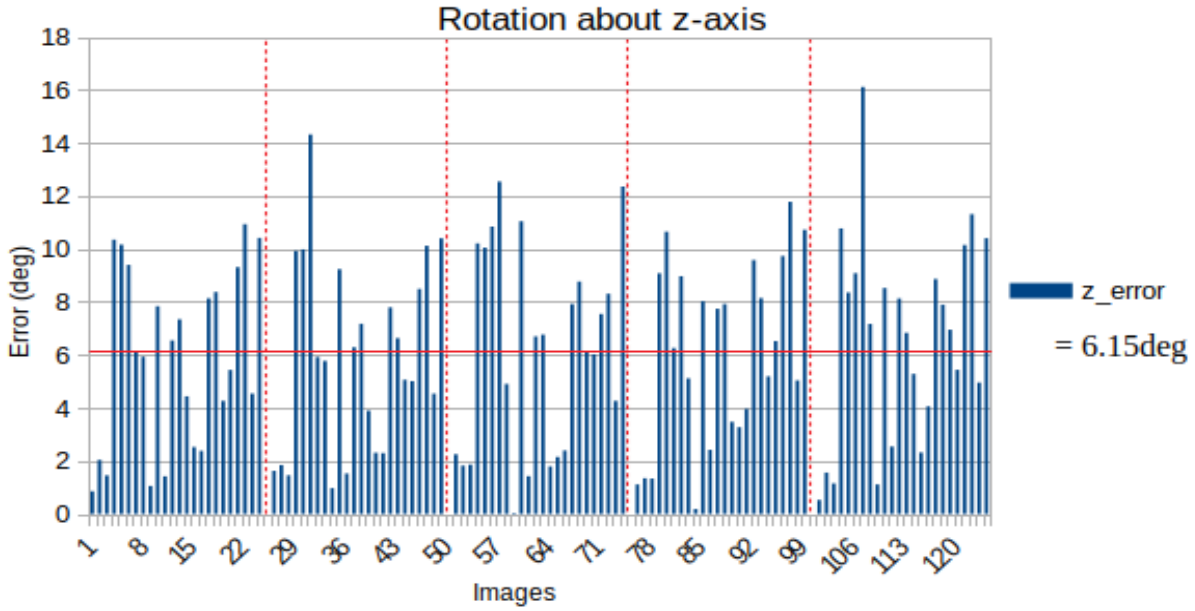


Fig. 4.18 This graph plots rotation error about the z-axis(Red axis) in degrees vs 125 testing images. The 4 dotted red lines divide the plot into 5 sections of different lighting setups, each plotting errors for 25 images. Mean error for a dataset containing images in varying lighting = 6.15°. We can observe very little deviation in errors($\leq$ 1°deviation) for the same hand images kept in various lighting setups.

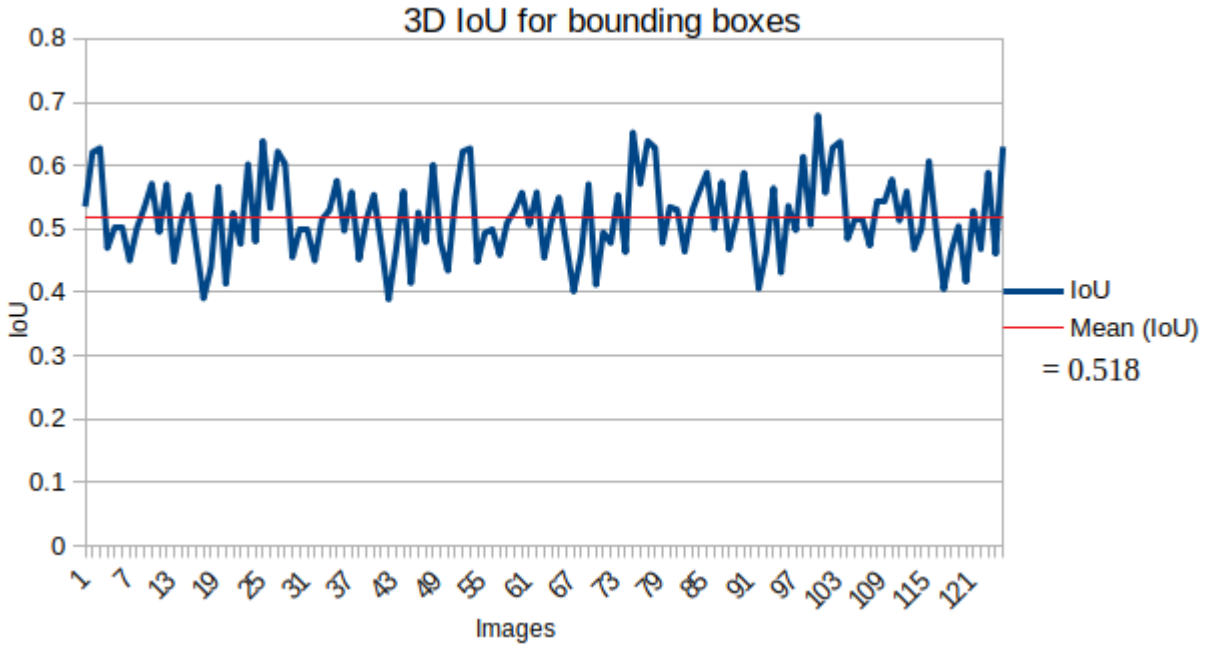
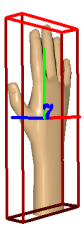
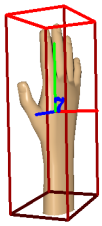


Fig. 4.19 The figure shows 3D IoU between predicted 3D ground truth and predicted bounding boxes vs testing images for images in different lighting setups. We obtain a mean IoU of 0.518, which is considered to be an acceptable prediction for bounding boxes and is the same as that for the initial lighting condition.

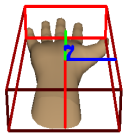
4.1.3 Varying scales



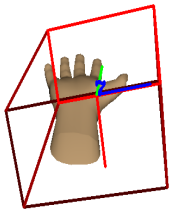
(a) **T:** (0,0,20.18)
R: (0.74,-137.6,0.15)



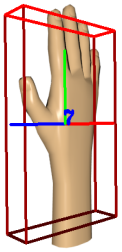
(b) **T:** (-0.1,0.2,20.04)
R: (-4.9,-155.9,-6.2)



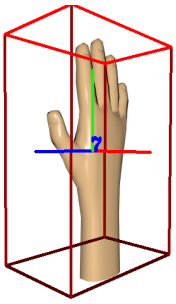
(c) **T:** (0,0,19.7)
R: (-66.6,-90.6,0.05)



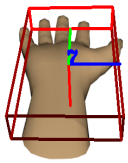
(d) **T:** (-0.4,0,19.9)
R: (-68.7,-101.6,-7.1)



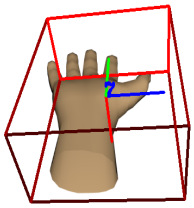
(e) **T:** (0,0,16.99)
R: (-0.32,-137.9,-0.8)



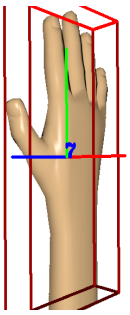
(f) **T:** (0.1,0.3,16.55)
R: (-2.36,-135.9,-1.4)



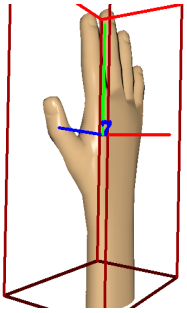
(g) **T:** (0,0.1,16.44)
R: (-66,-92.8,-0.06)



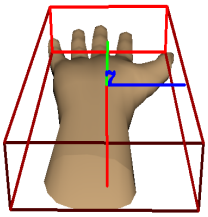
(h) **T:** (-0.3,0.2,17.03)
R: (-50.16,-98.38,-2.97)



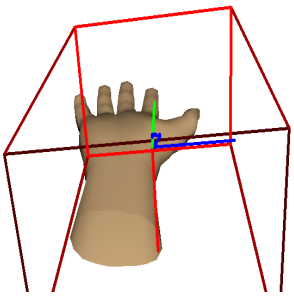
(i) **T:** (0,0,14.1)
R: (0.58,-137,-0.06)



(j) **T:** (0,0.4,13.6)
R: (0.9,-148.7,2.47)

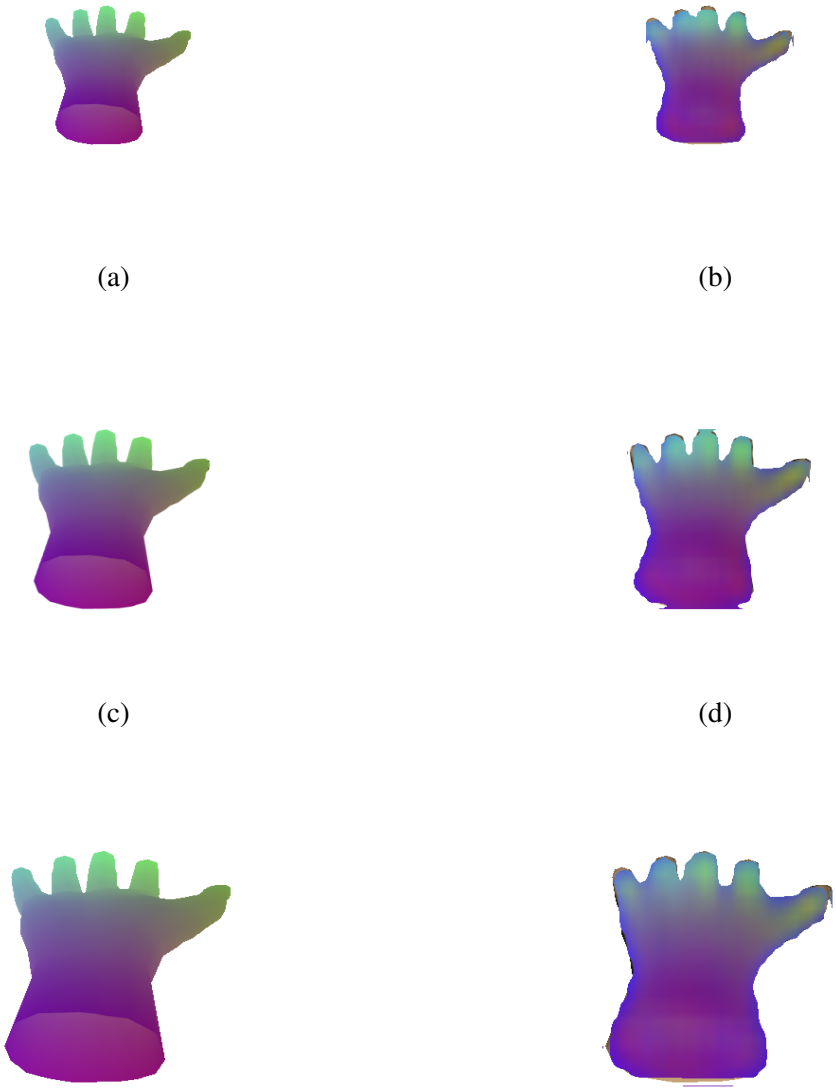


(k) **T:** (0,0,13.91)
R: (-65.5,-91.1,-0.02)



(l) **T:** (-0.6,-0.1,14.6)
R: (-65.6,-95.7,-3.34)

Fig. 4.20 The plots above show ground truth(gt) and predicted(pr) 6D pose of the same hand image at 3 different distances from the camera(1,3 cols-gt;2,4 cols-pr). The translation(**T** in cms) and rotation(**R** in degrees) triplets have been mentioned as captions for each image. Similar to Fig.4.1, The red boxes indicate the bounding boxes, the RGB axes give an idea of translation and rotation of the hand.
The translation errors between ground truth and predicted images are found to be similar for different scales, which is < 1cm. The rotation errors can be observed to be different for different scales.



(a) (b)

(c) (d)

(e) (f)

Fig. 4.21 Col-1: Ground truth NOCS map, Col-2: Predicted NOCS map.
Shown above are the ground truth and predicted NOCS maps of hand at 3 different distances from the camera, corresponding to (c),(g),(k) hands Fig. 4.6. The more is the error in the NOCS map, more is the error in the estimated pose.

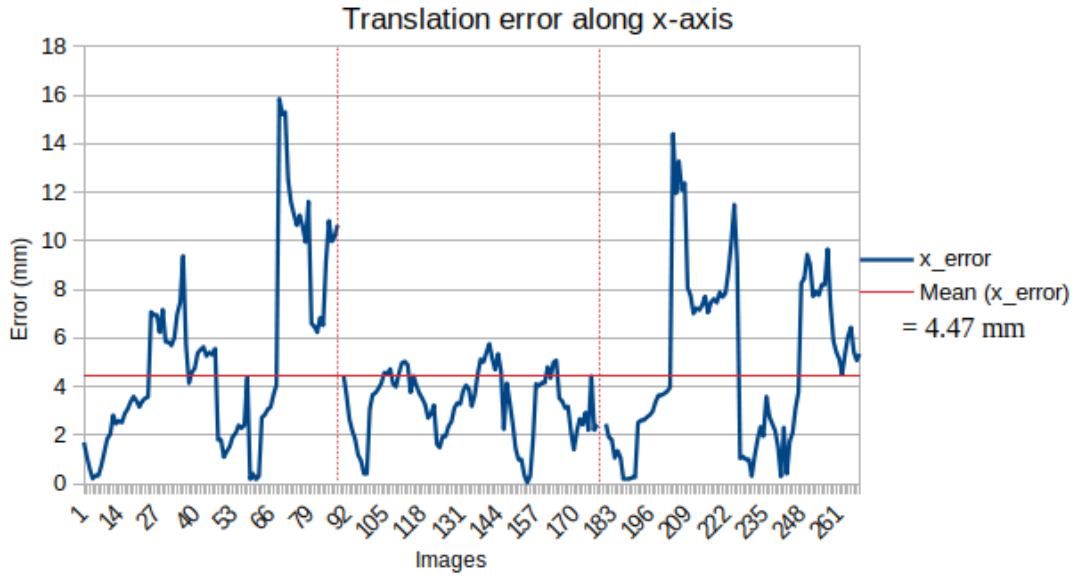


Fig. 4.22 This graph plots Translation error along the x-axis(Blue axis) in mms vs 261 testing images. The red dotted lines divide the plot into 3 sections, 87 images each, decreasing in distance from the camera from left to right. We can observe that along the x-axis the images that are too far or too close to the camera have greater errors. The maximum translation error is 1.6cm, minimum is 0cm, and mean error, shown by the red line, is found to be 4.47mm for a dataset containing images at all distances.

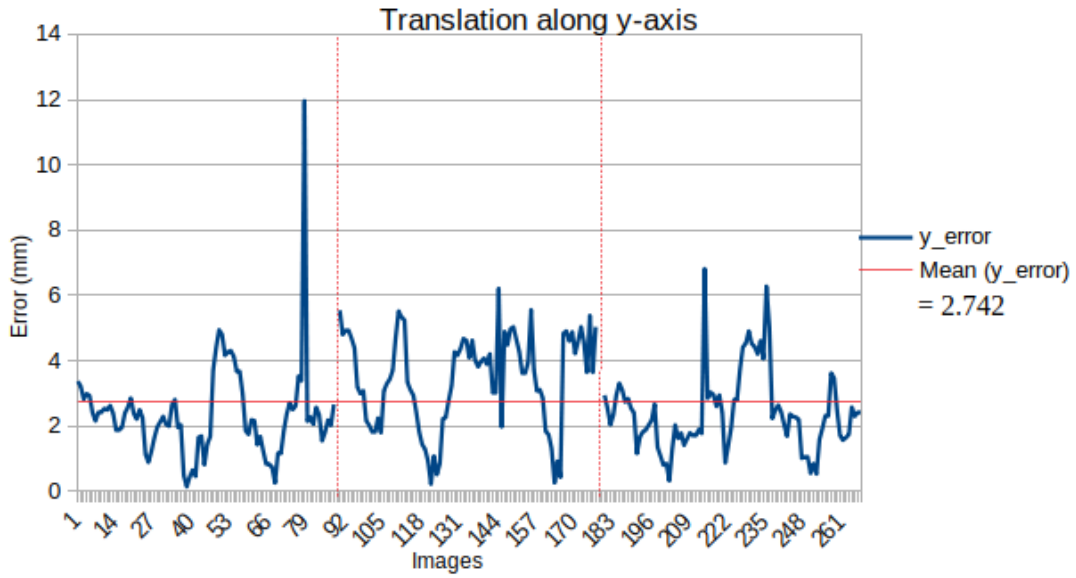


Fig. 4.23 This graph plots Translation error along the y-axis(Green axis) in mms vs 261 testing images. The red dotted lines divide the plot into 3 sections, 87 images each, decreasing in distance from the camera from left to right. We can observe that along the y-axis the images at all distances have found to have similar prediction errors. The maximum translation error is 1.2cm, the minimum is 1mm, and mean error, shown by the red line, is found to be 2.742mm for a dataset containing images at all distances.

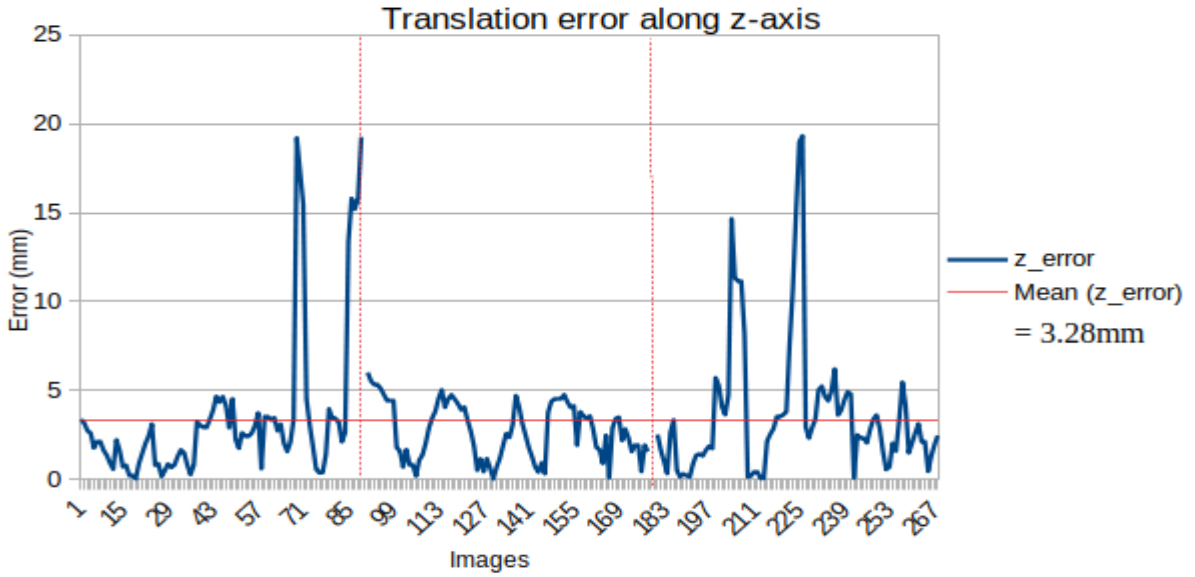


Fig. 4.24 This graph plots Translation error along the x-axis(Blue axis) in mms vs 261 testing images. The red dotted lines divide the plot into 3 sections, 87 images each, decreasing in distance from the camera from left to right. We can observe that along the x-axis the images that are too far or too close to the camera have slightly larger errors. The maximum translation error is 1.8cm, minimum is 0cm, and mean error, shown by the red line, is found to be 3.28mm for a dataset containing images at all distances.

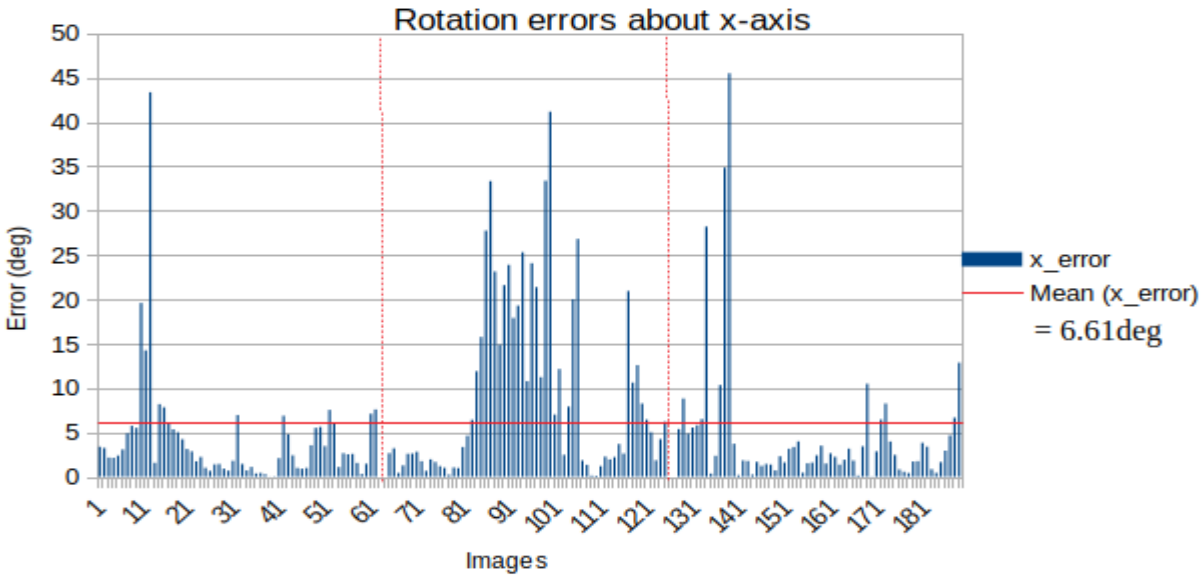


Fig. 4.25 This graph plots rotation error about the x-axis(Blue axis) in degrees vs 186 testing images. The 2 dotted red lines divide the plot into 3 sections of hand with varying distances from the camera. The left section shows hands that are 14cm away from the camera, the middle being 16cm and right being 20cm.

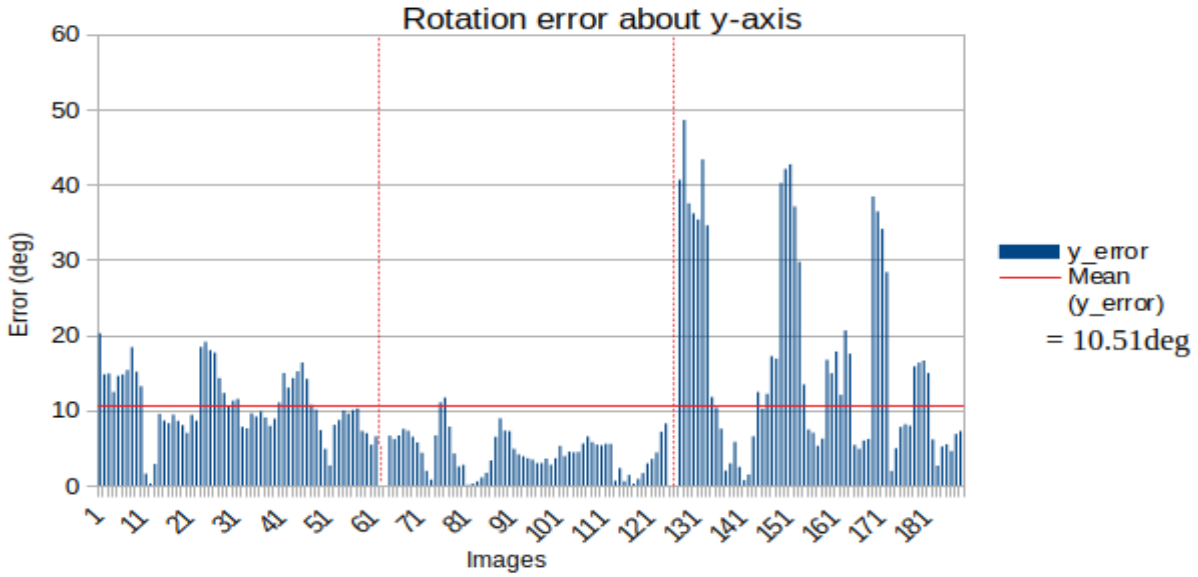


Fig. 4.26 This graph plots rotation error about the y-axis(Green axis) in degrees vs 186 testing images. The 2 dotted red lines divide the plot into 3 sections of hand with varying distances from the camera. The left section shows hands that are 14cm away from the camera, the middle being 16cm and right being 20cm. We observe that images that are close to the camera have significantly higher errors than the other two cases. This trend is reflective of the NOCS maps predicted for the same hand at different distances.

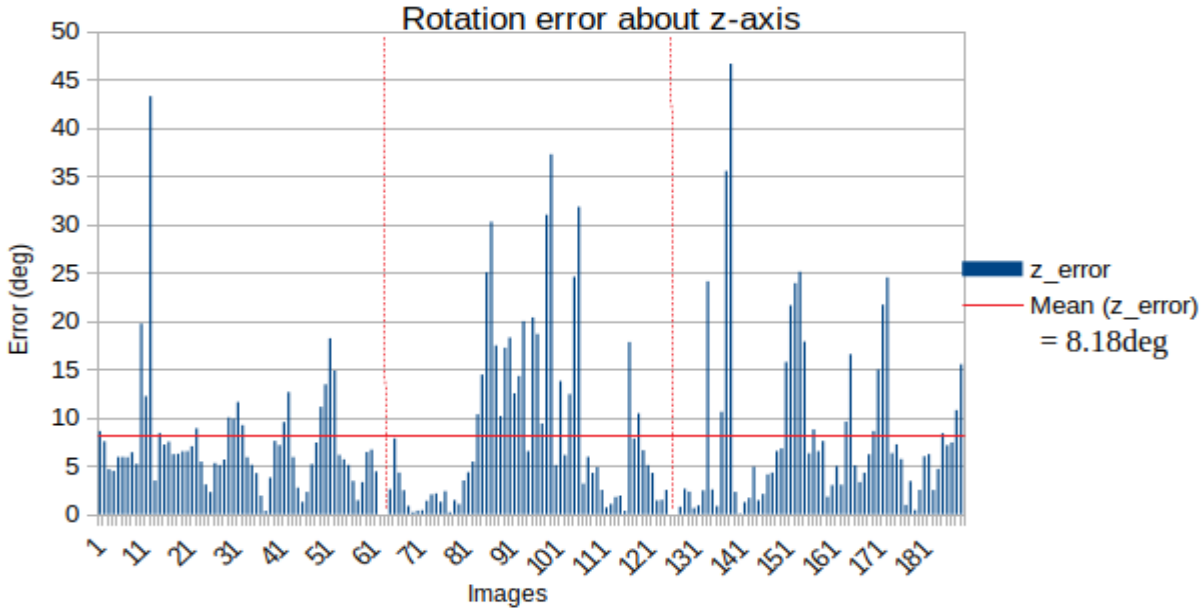
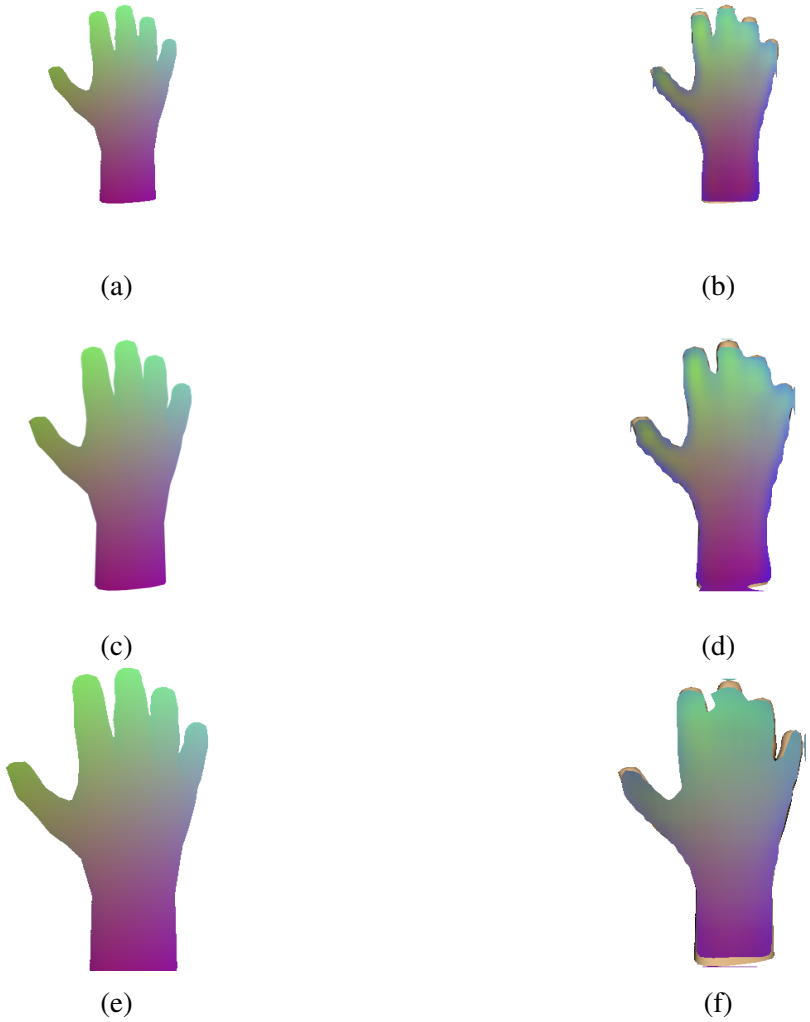


Fig. 4.27 This graph plots rotation error about the y-axis(Green axis) in degrees vs 186 testing images. The 2 dotted red lines divide the plot into 3 sections of hand with varying distances from the camera. The left section shows hands that are 14cm away from the camera, the middle being 16cm and right being 20cm.

Anomalies

From the graph of the rotation error about y-axis, we observe a significantly higher error for hand images that are closer to the camera. Shown below are NOCS map comparisons between the same hand placed at different distances. We can observe that for hand images closer to the camera, the NOCS maps predictions are more errant when compared to the hands that are farther away.



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

Fig. 4.28 Col-1: Ground truth NOCS map, Col-2: Predicted NOCS map.
Shown above are the ground truth and predicted NOCS maps of the same hand at 3 different distances from the camera. The NOCS map predicted when the hand is close to the camera is less accurate and precise. The more is the error in the NOCS map, more is the error in the estimated pose.

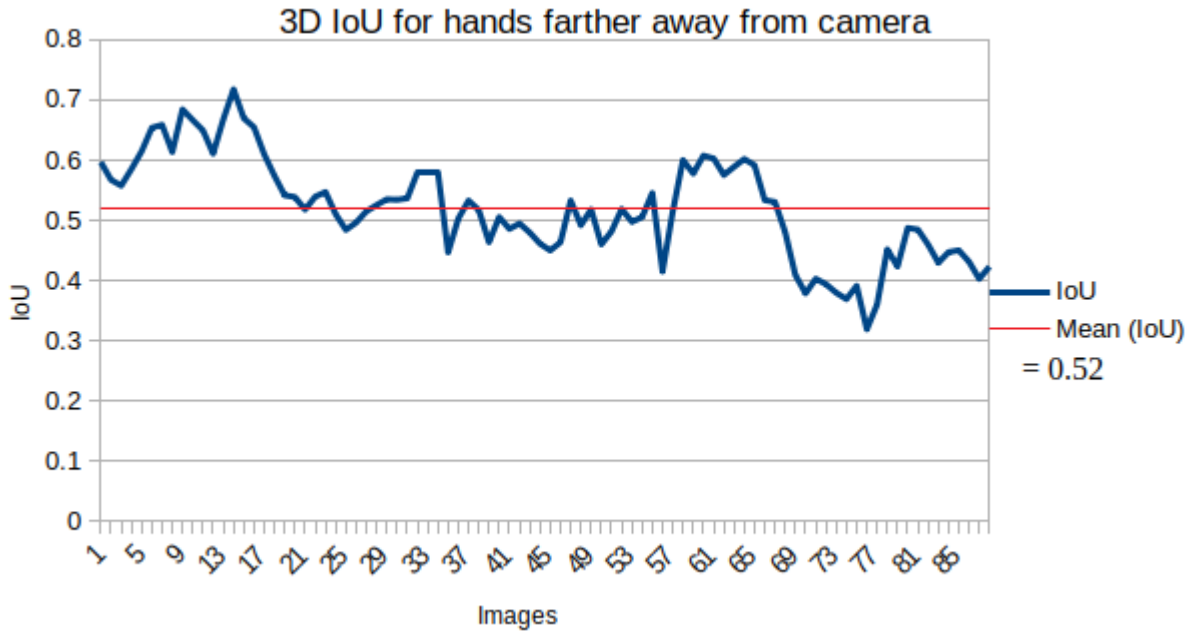


Fig. 4.29 The figure shows 3D IoU between predicted 3D ground truth and predicted bounding boxes vs testing images when the hand is farther from the camera. We obtain a mean IoU of 0.52, which is considered to be an acceptable prediction for bounding boxes.

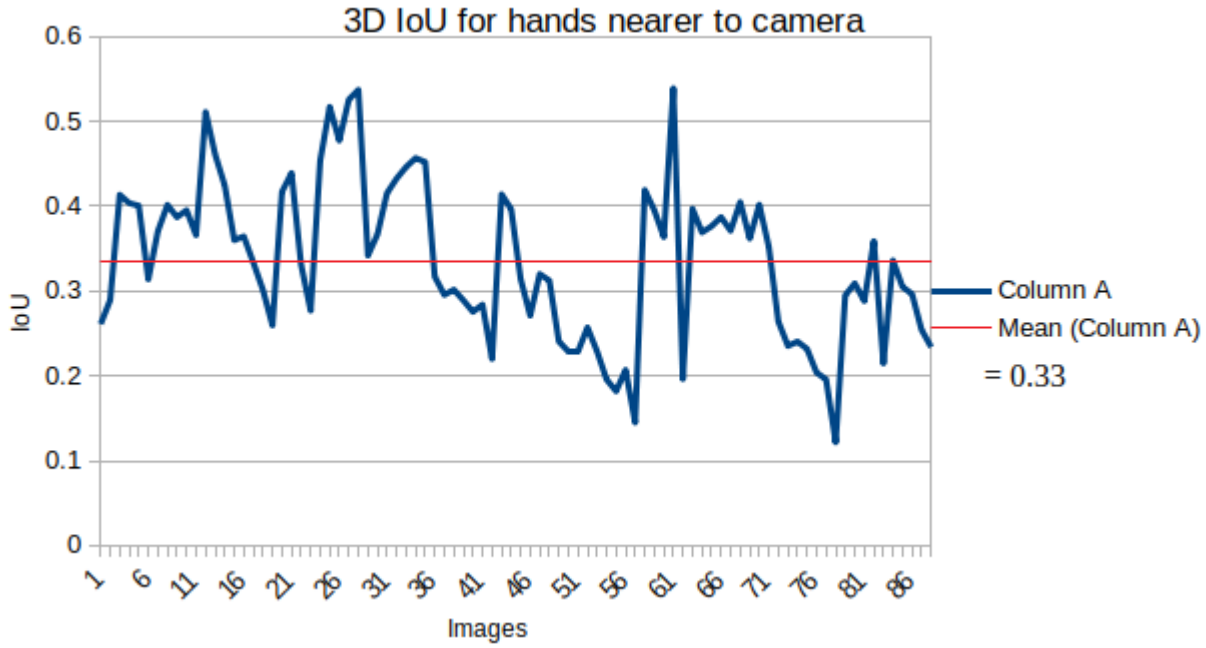


Fig. 4.30 The figure shows 3D IoU between predicted 3D ground truth and predicted bounding boxes vs testing images when the hand is nearer to the camera. We obtain a mean IoU of 0.33, which indicates poor bounding box prediction. The 3D bounding box in our model is calculated from the dimensions of the 3D point cloud generated from the NOCS map and hence a lesser mean IoU is reflective of the NOCS map error.

4.2 Edge-Agreement-Loss

For 6D pose estimation using the NOCS model, instance map prediction holds a lot of importance. The instance mask is used to set the boundary of the NOCS map to be considered for pose estimation. Hence if some area of the object is not detected in the instance mask, then that area will be left out in the NOCS map too and the information about the orientation contained in those pixels will be lost. Hence improving the accuracy of instance segmentation is crucial in our problem.
We add an auxiliary Edge Agreement Head [15] in our model which predicts the edge maps from the ground truth and predicted instance masks. A mean square error loss is found out from the edge maps, which we call the edge agreement loss. This loss is added to the total mask loss. The edge loss improves the coverage at edges of objects and helps in achieving the same loss at a faster rate. We have used various edge loss filters mentioned in sec 2.4 while training and compared the results to know which works the best for our problem.

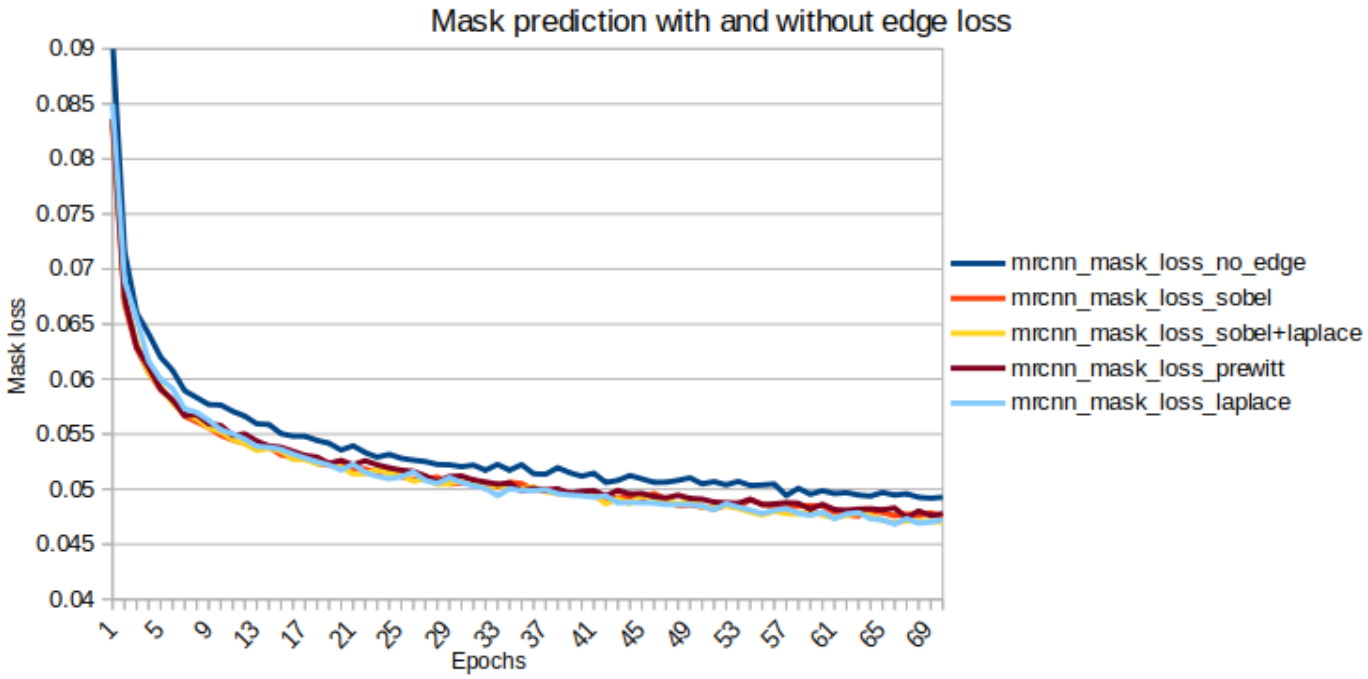


Fig. 4.31 The figure shows training mask prediction loss for the baseline model and NOCS model + Edge loss head for different edge detection filters over the first 70 epochs. The losses obtained after using edge detection filters are lower than the top blue line which shows the baseline loss. We observe a **4%** decrease in mask loss at the end of 70 epochs.

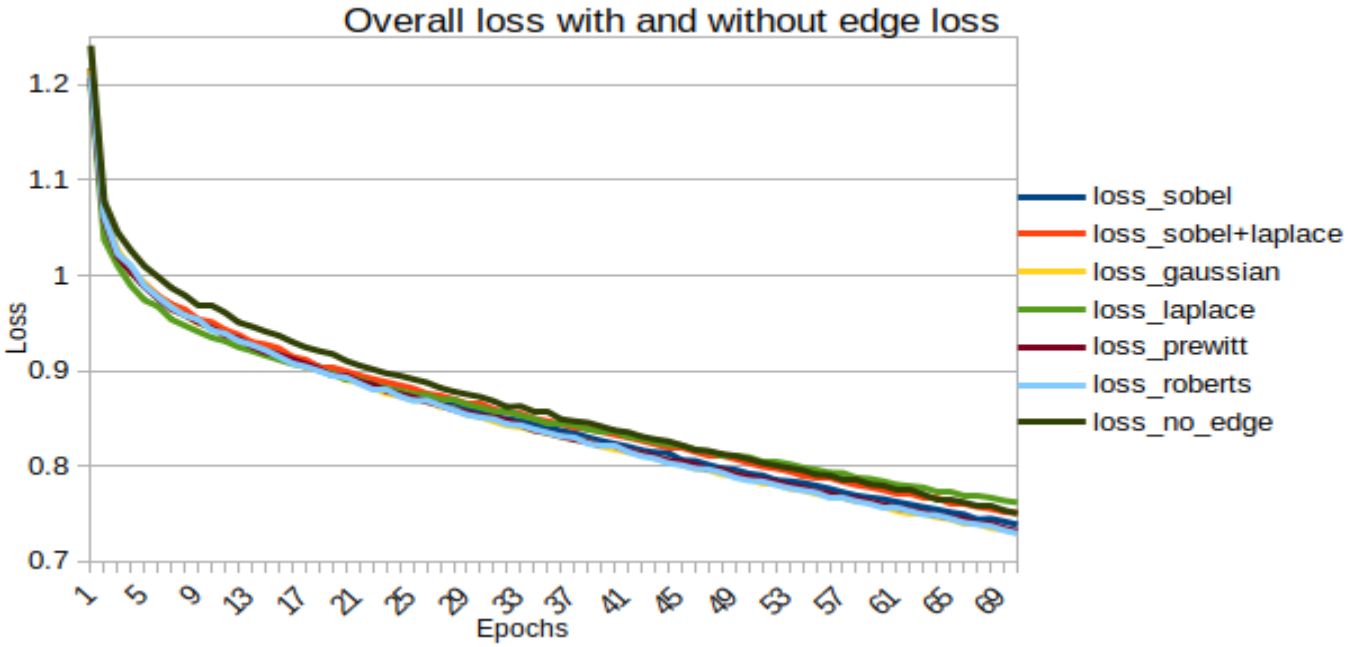
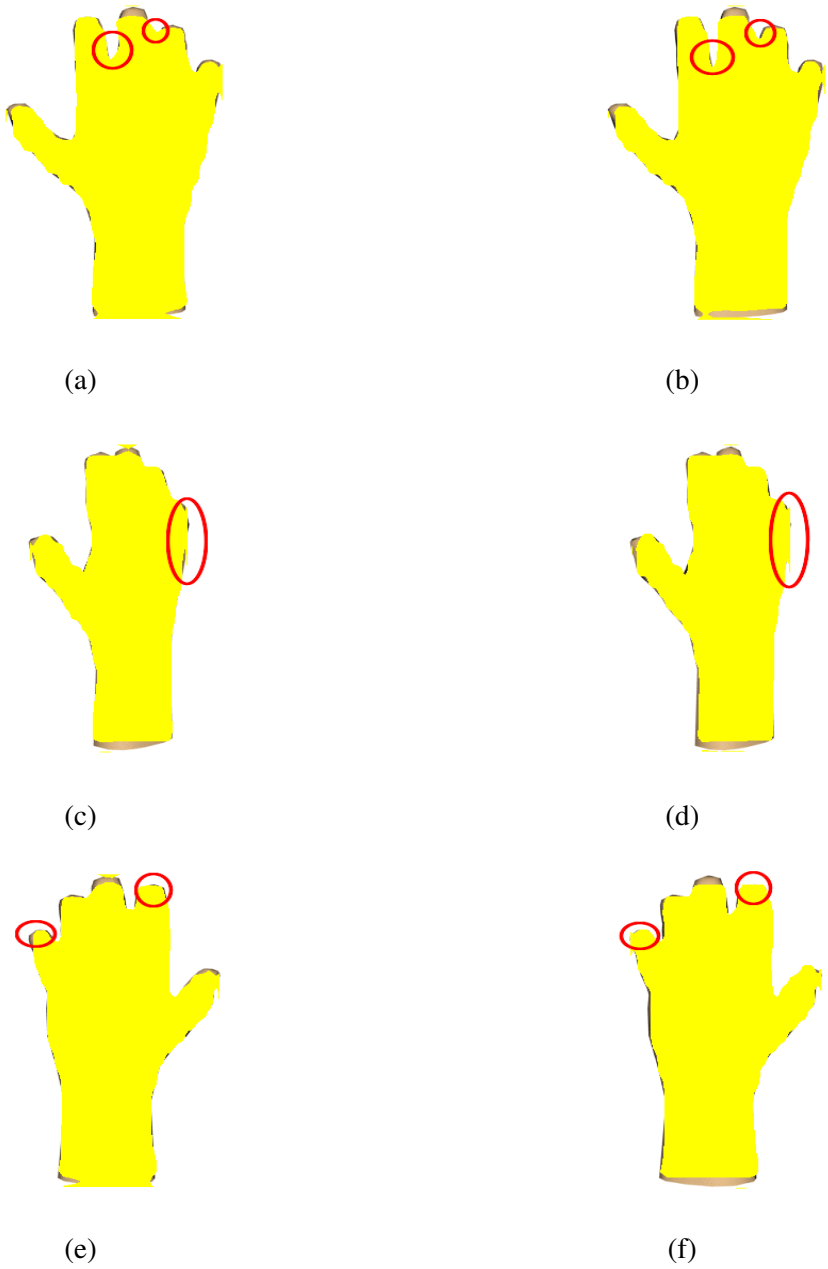


Fig. 4.32 The figure shows the overall training loss for the baseline model and NOCS model + Edge loss head for different edge detection filters over the first 70 epochs. The losses obtained after using edge detection filters are lower than the top blue line which shows the baseline loss for all filters except Laplace. We observe a **2.8%** decrease in the overall loss at the end of 70 epochs.



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

Fig. 4.33 The above figures show 3 examples of instance maps predicted by a model trained without the addition of edge loss(1^{st} col) vs the model trained with the Laplace filter(2^{nd} col). We observe that with the usage of edge loss, the predicted masks have better coverage of the object area, cleaner edgers without outliers, and more distinct features like identification of the separation between fingers(a,b). All these differences have been highlighted in the images with red marks.

The effect of the addition of edge loss can be better observed in images with hand a lot closer to the camera. Since we had used only a few images with lesser distances to the camera, the training of these kinds of images still had a lot of errors. Shown below are the images tested on two models trained with and without edge loss.

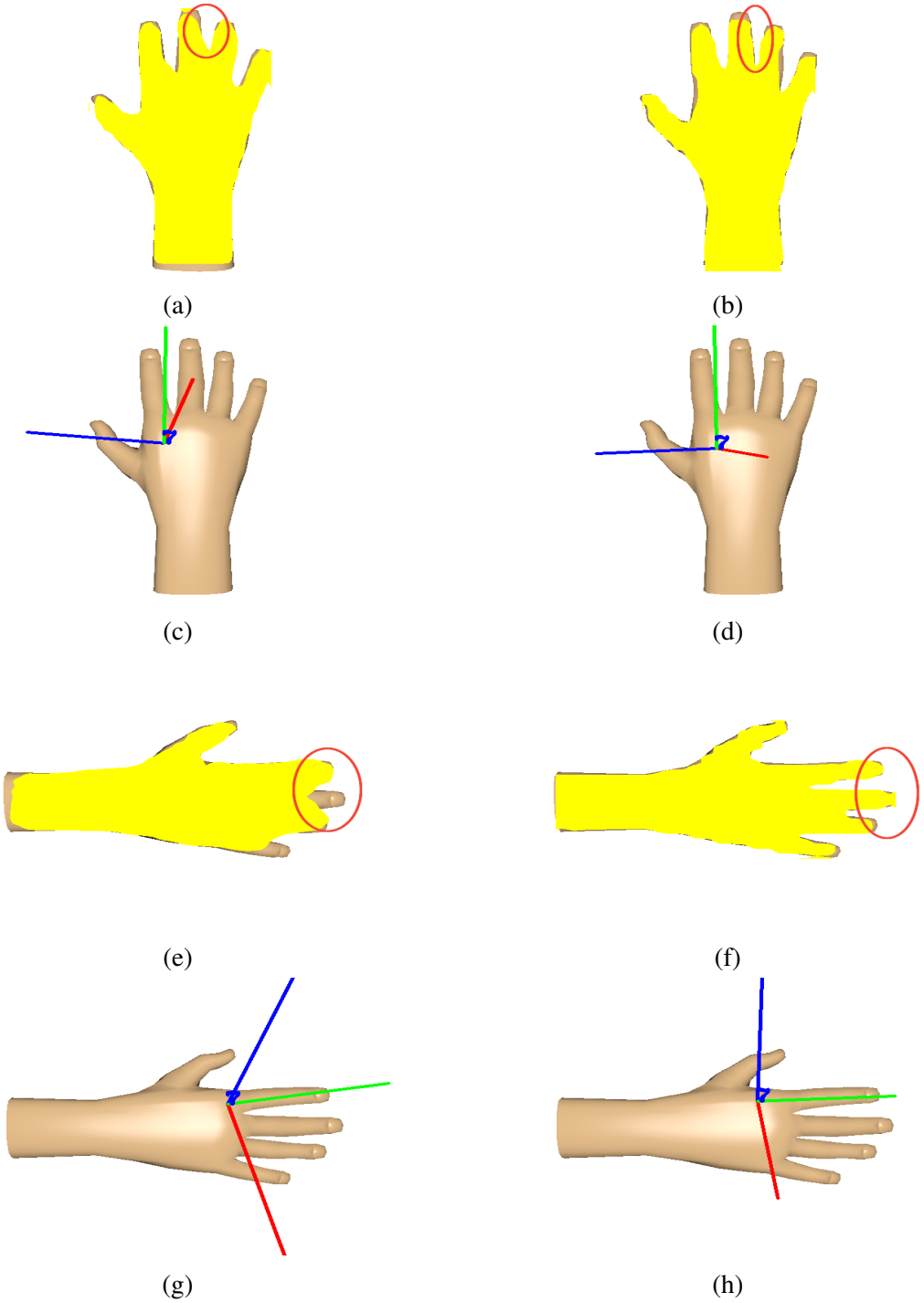
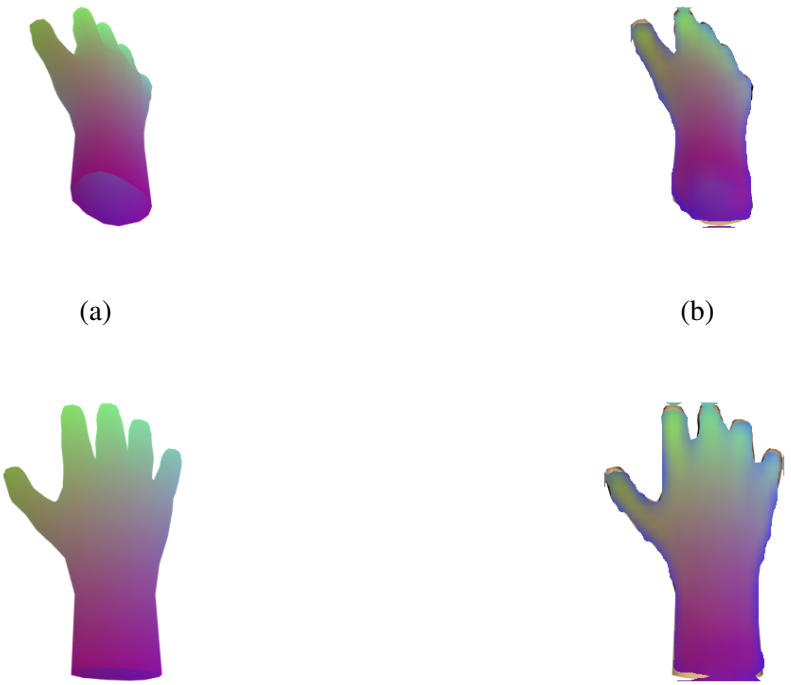


Fig. 4.34 1^{st} col-predictions by model without edge loss, 2^{nd} col-predictions by model with Laplace edge filter. The 6D pose diagrams of hand images have been indicated right below their corresponding masks. From these diagrams, we can infer the direct relation between the accuracy of predicted of instance masks with the 6D pose predicted. As the mask predicted accuracy increases, the NOCS map accuracy increases, hence directly impacting the final 6D pose of the object.

4.2.1 Edge errors in NOCS map



(a) (b)

(c) (d)

Fig. 4.35 (a),(c)-ground truth,(b),(d)-predicted NOCS maps. If we observe the edges we can notice that they are a lot more erratic when compared to the rest of the map area. Errors in the predicted masks lead to errors in the 6D pose estimated.

In order to rectify the edge errors that occur in the predicted NOCS maps, we propose the use of a similar loss function to that of he Edge Agreement Loss used for mask prediction. Firstly, the edge detection filters are used on the NOCS maps, which will give us the edge mask image of the hand. We have also found out through experiments that using the filters **multiple times yields thicker edges** of the objects. The ground truth and predicted NOCS map edges are extracted by applying the edge mask over the ground truth and predicted NOCS maps respectively. A softmax loss in case of classification or a smooth L1 loss in case of regression can be used to compute the edge error in NOCS map. This new loss is added to the NOCS loss of the whole map. This method increases the attention to the edges of the NOCS map and can improve the edge prediction in the NOCS map.
We are currently working on incorporating this idea into our model.

(a) (b)

Fig. 4.36 The ground truth(a) and predicted(b) NOCS edges maps extracted from their respective NOCS maps using the canny edge detection filter used 5 times on the map. The distinct colour difference between the fingers of the hand can be noted. These edge maps can now be used to find out the NOCS edge loss mentioned in the section above.

4.3 Summary of results obtained from our research work

1. The system used for training and testing of the model is the NVIDIA Tesla V100. The training process takes 1min 5secs/epoch, where 1 epoch comprises of 1000 iterations. Prediction and 6D pose estimation take 5.39sec/image.

2. The model was trained for the identification of 6D pose of synthetically created humanoid hand dataset. The 6D pose and bounding boxes have been estimated for hand images varying in orientation, lighting, and scales. Average translation error obtained for dataset containing hand images = 3.529mm and average rotation error = 6.22°.

3. From the analysis, it can be concluded that varying lighting conditions have a negligible effect on the 6D pose predicted while scale variation has been found to be slightly more erratic.

4. The Edge Agreement Head was added to improve the accuracy of mask prediction. At the end of 70 epochs, there was a decrement of mask prediction loss by 4% and an over loss decrement of 2.8%.

5. The addition of Edge Prediction Head and edge loss has improved mask coverage, decreased outliers, and bettered identification of features.

The Hand Dataset created along with codes is available on this Github page: https://github.com/AnjuVolt/EdgeAttention_6Dpose_estimation

Chapter 5

Conclusions

Computer Vision is a field of study where a computer learns to identify and characterize objects present in images/videos. We use computer vision for 6D pose and dimension estimation of hand-scale objects using the NOCS model as the baseline. We create a synthetic hand dataset consisting of 1900 training images, 300 validation, and 300 testing images in various orientations, 5 different lighting setups, and at 3 different distances from the camera. A new model is created with modifications made in the head layers of the network to adapt to additional classes, along with the addition of an Edge Agreement Head to improve mask accuracy. The model is trained on the created hand dataset. The testing results show the accurate prediction of the 6D pose of the humanoid hand in varying orientation, lighting, and scale. The addition of edge loss has decreased the instance mask prediction loss by 4% and the overall loss by 2.8%.

Possible future works:

The NOCS maps predicted by the model have been observed to be erratic along the edges of the objects. To rectify this, we are currently working on implementing an auxiliary edge prediction head and edge loss for NOCS map prediction, which can improve the accuracy of the 6D pose even further. The bounding boxes predicted by the proposed model are currently scaled up for all objects. This area has a scope for a lot of improvement. The model is yet to be trained and tested on occluded datasets where objects are only partially visible.

References

[1] Fakhr-eddine Ababsa and Malik Mallem. Robust camera pose estimation using 2d fiducials tracking for real-time augmented reality systems. In *Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*, pages 431–435, 2004.

[2] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[3] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[4] Shiqi Li and Chi Xu. Efficient lookup table based camera pose estimation for augmented reality. *Computer animation and virtual worlds*, 22(1):47–58, 2011.

[5] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. Fully convolutional instance-aware semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2359–2367, 2017.

[6] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection, 2016.

[7] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4561–4570, 2019.

[8] Bugra Tekin, Sudipta N Sinha, and Pascal Fua. Real-time seamless single shot 6d object pose prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 292–301, 2018.

[9] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (4):376–380, 1991.

[10] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In

Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2642–2651, 2019.

[11] Wikipedia contributors. Arg max — Wikipedia, the free encyclopedia, 2019. [Online; accessed 22-October-2019].

[12] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[13] Jong-Hyun Yoon, Jong-Seung Park, and Chungkyue Kim. Increasing camera pose estimation accuracy using multiple markers. In *International Conference on Artificial Reality and Telexistence*, pages 239–248. Springer, 2006.

[14] Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. Dpod: 6d pose object detector and refiner. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1941–1950, 2019.

[15] Roland S Zimmermann and Julien N Siems. Faster training of mask r-cnn by focusing on instance boundaries. *Computer Vision and Image Understanding*, 188:102795, 2019.

Appendix A

Introduction to Neural Networks used

The NOCS model uses a ResNet50 + Feature Pyramid Network(FPN) backbone for feature extraction similar to the Mask-RCNN network [3].

A.1 FPN

Feature Pyramid Network(FPN) is a feature extractor used for multi-scale object detection. It has the conventional pyramidal bottom-up convolutional blocks. Along with this is a top-down architecture where the outputs of each convolutional block starting from the top are upsampled and added to the next layer, which builds high-level semantic feature maps at different scales [6]. This can be used in a variety of applications as a generic feature extractor.

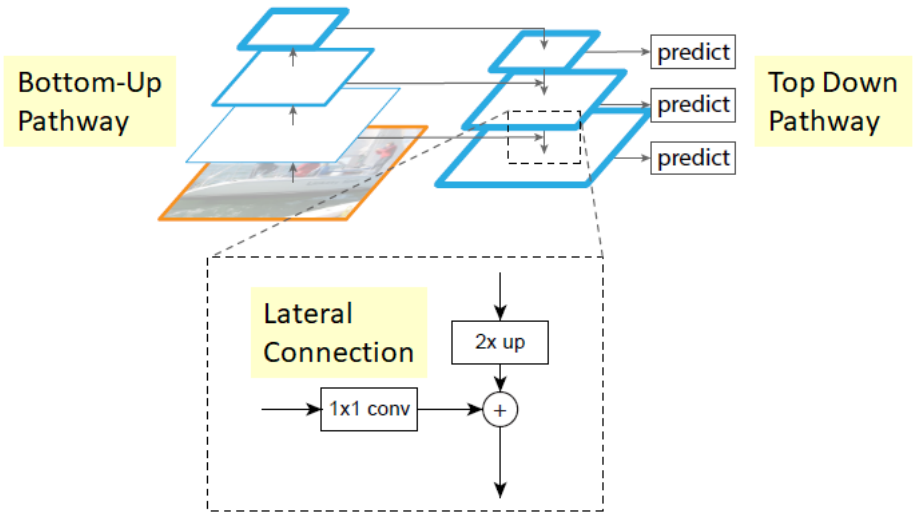


Fig. A.1 The figure shows the Feature Pyramid Network(FPN) block. The bottom-up convolutional layers and top-down pathway generating different feature maps can be seen. The lateral connection between the layers in the top-down path does the upsampling of output from the present layer and adds that to the output of the next layer.[1]

A.2 ResNet50

A residual neural network (ResNet) is an artificial neural network (ANN) whose idea originated from the pyramidal cells in the cerebral cortex. Residual neural networks do this by employing skip connections, or short-cuts that jump over a few layers. Typical ResNets are implemented using double or triple layer skip connections with non-linearities (ReLU) and batch normalization in between.

layer name	34-layer	50-layer	101-layer
conv1	$7 \times 7, 64$, stride 2		
conv2_x	3×3 max pool, stride 2		
	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
conv4_x	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$
conv5_x	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	average pool, 2048-d fc		

Fig. A.2 Comparison between the convolutional blocks used in ResNet34, ResNet50 and ResNet101. ResNet50 has 25.6M trainable parameters whereas ResNet34 has 21.5M and ResNet101 has 44.5M. [2]

[1] https://towardsdatascience.com/review-fpn-feature-pyramid-network-object-detection-262fc7482610

[2] https://www.researchgate.net/figure/Architectures-for-ResNet34-ResNet50-and-ResNet101-in-this-paper-Building-blocks-are_tbl1_334288428

Appendix B

Metrics

B.1 Argmin

Also known as the arguments of the minimum (abbreviated arg min or argmin) are the elements, or points, of the domain of some function at which the function values are minimum [11].

$$argmin_x f(x) = \{x | x \in S \wedge \forall_y \in S : f(y) \geq f(x)\} \tag{B.1}$$

are points x for which $f(x)$ achieves its smallest value [11].

B.2 MSE

Mean Square Error(MSE) calculates the average of the squared difference between true and predicted values. It is a commonly used error in machine learning. It gives higher weightage to larger errors when compared to linear error metrics and gives lesser weightage to outliers when compared to error metrics of degree > 2.

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \tag{B.2}$$

B.3 IoU

Intersection over Union(IoU) is an evaluation metric used to measure the accuracy of predicted bounding boxes. As the name suggests, it is the ratio of the intersection of the ground truth and predicted bounding boxes over the union of the ground truth and predicted bounding boxes. The metric's values remain in the range of (0,1), 0 being the worst prediction where the intersection is 0, and 1 being the perfect prediction where intersection = union.

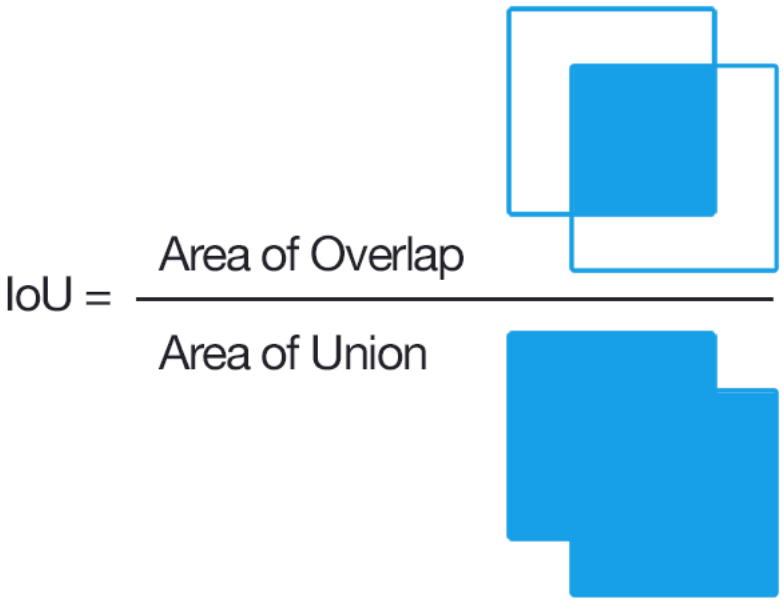


Fig. B.1 IoU metric = intersection of the ground truth and predicted bounding boxes over the union of the ground truth and predicted bounding boxes.

Title Sc16b066

ORIGINALITY REPORT

5%	3%	5%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	export.arxiv.org Internet Source	1%
2	"Computer Vision – ECCV 2018", Springer Science and Business Media LLC, 2018 Publication	<1%
3	towardsdatascience.com Internet Source	<1%
4	Computer Vision, 2014. Publication	<1%
5	"Communications, Signal Processing, and Systems", Springer Science and Business Media LLC, 2020 Publication	<1%
6	Shiqi Li. "Efficient lookup table based camera pose estimation for augmented reality", Computer Animation and Virtual Worlds, 01/2011 Publication	<1%
7	www.spiedigitallibrary.org Internet Source	<1%