Edge attention aided 6D pose estimation of Customized Dataset

Anju S
(SC16B066)

under the guidance of Dr. Deepak Mishra and Dr. Sheeba Rani

DEPARTMENT OF AVIONICS
Indian Institute of Space Science and Technology(IIST)

anjuskumar1313@gmail.com

July 20, 2020

Overview

Robotics

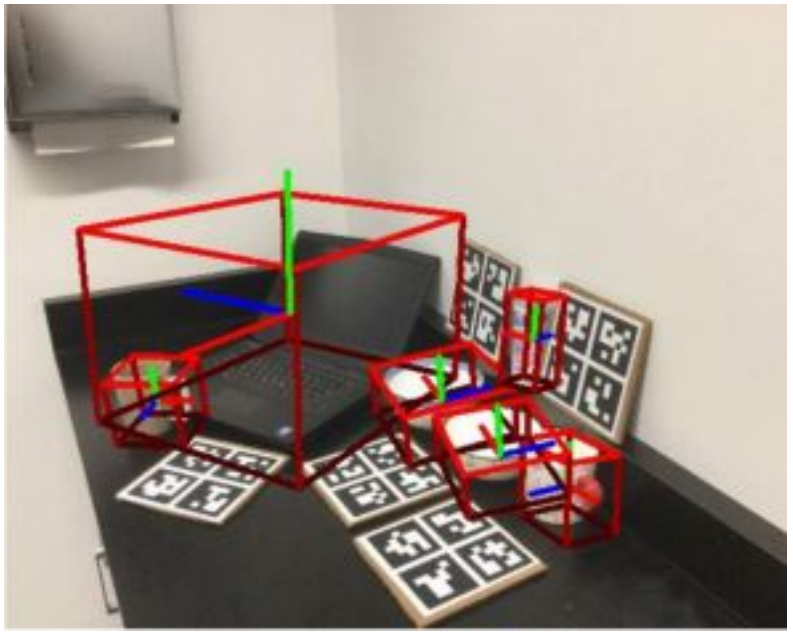
- Numerous robotics applications involve recognition tasks where the robot needs to identify and make sense of its environment in order to perform its function.
- Eg. Mobile apps, self-driving cars, industrial robots(pick and place) etc.

Problem Statement

- A half-humanoid robot that needs to find the 6D pose of objects within its available field of view.
- An Intel Real sense camera is attached to its torso that outputs the RGB and depth images.
- Multiple objects must be located.
- Classification and 6D pose of each object as output.
- The pose estimation should be immune to the lighting condition of the environment.

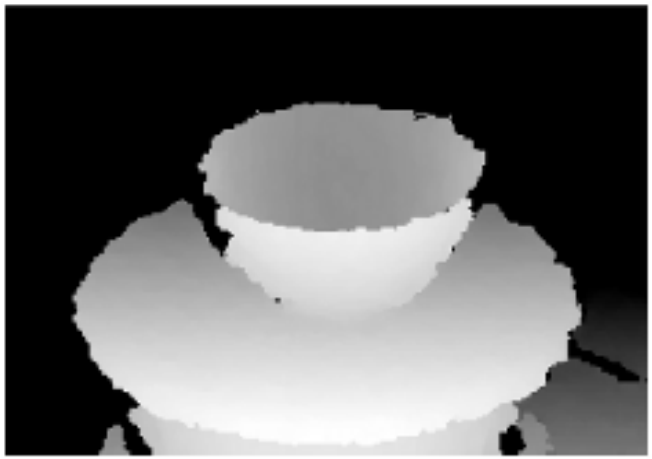
6D pose estimation

- 6-D object pose estimation deals with estimating the **3D position and 3D rotation** of objects in camera-centred coordinates.
- 3D position - (x, y, z) coordinates
  3D rotation - (ψ, θ, ϕ) angles



Two streams of methods are usually used for 6D pose estimation:

1. From RGB images: PnP(Perspective-n-point) algorithm, Fiducial Markers etc.
2. From RGB-D images: Depth data gives us the 3rd dimension.

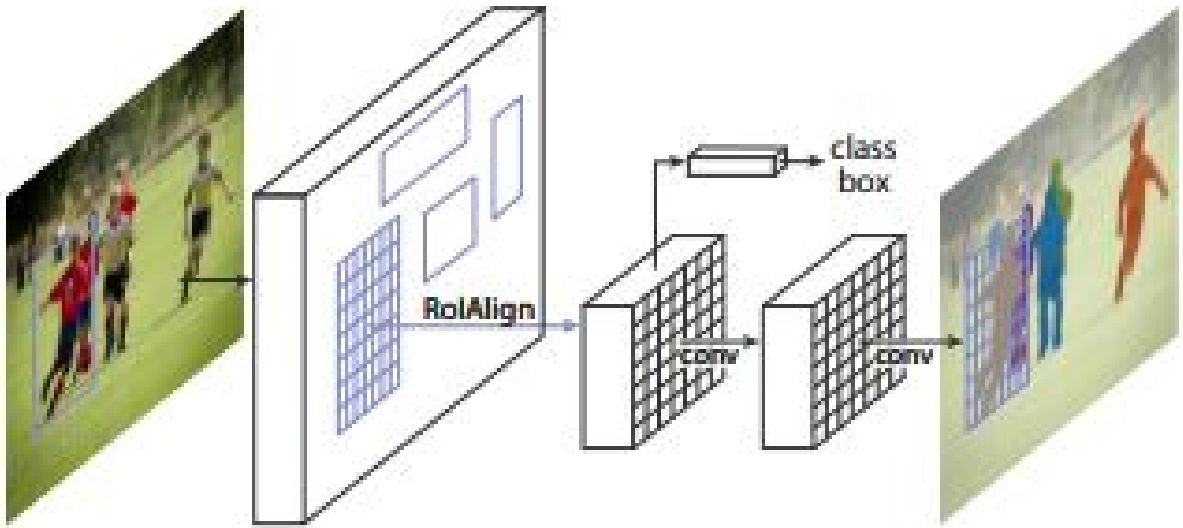


In our project we will be focusing on 6D pose estimation from RGB-D images.

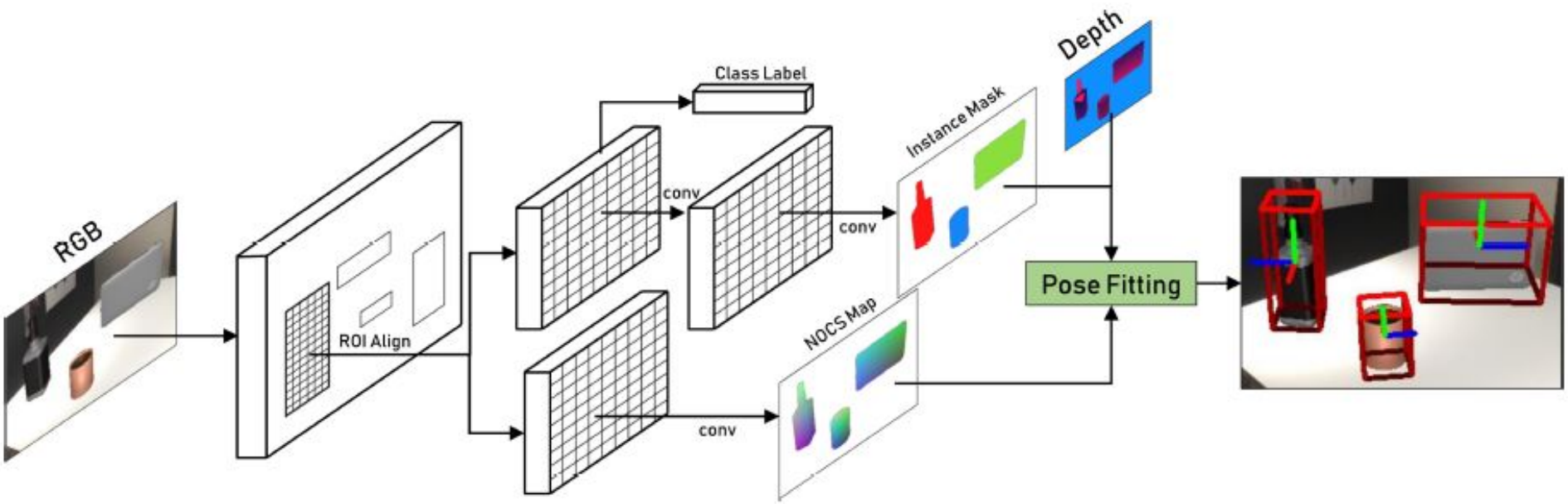
Normalised Object Coordinate Space(NOCS)

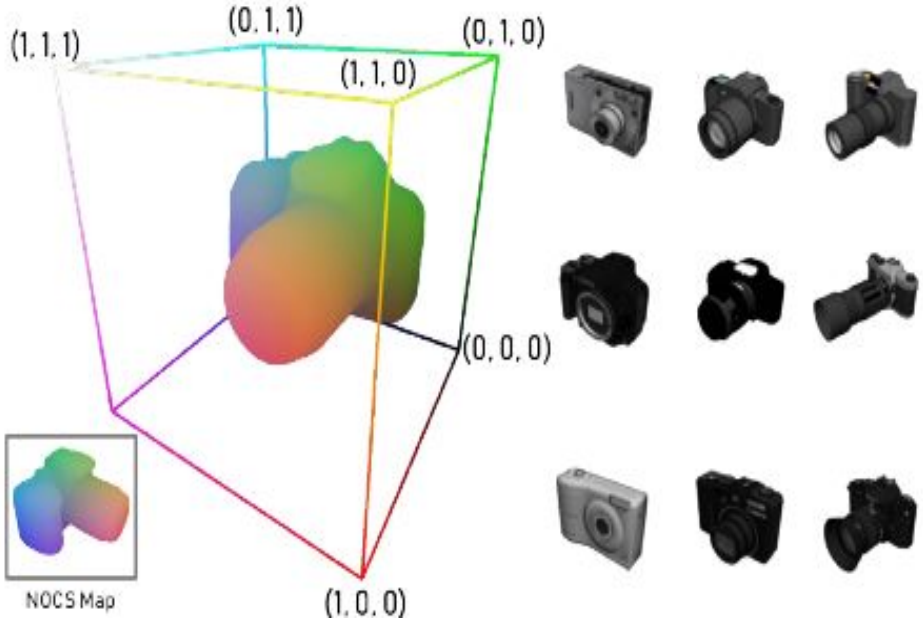
Model

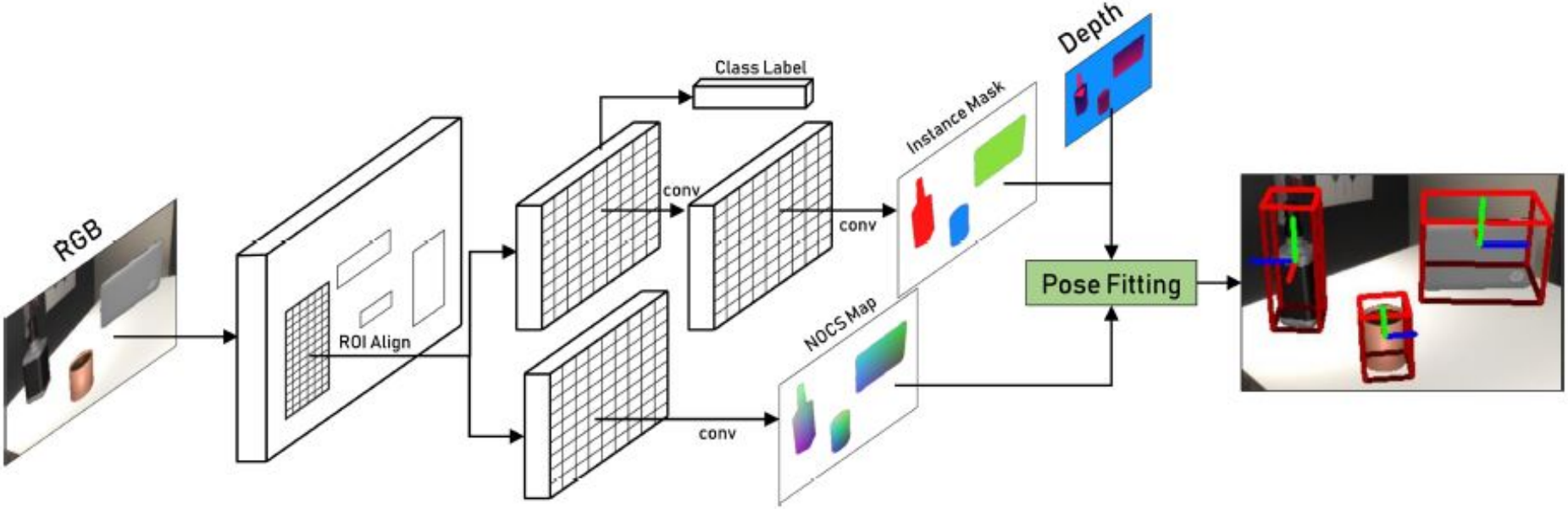
- The base of our model is the **Mask-RCNN** network.
- Mask-RCNN provides us with instance segmentation and classification.



To the Mask-RCNN network one more convolution block is added which outputs an NOCS map for every object.




(1, 1, 1)
(0, 1, 1)
(0, 1, 0)
(1, 1, 0)
(0, 0, 0)
(1, 0, 0)
NOCS Map



- From instance masks and depth data we obtain a 3D point cloud which is compared with the 3D point cloud obtained from NOCS maps to obtain the rotation and translation of each object.

Umeyama algorithm

Let P and Q be the two 3D point cloud data obtained.

- Subtract the respective centroids from both P and Q to bring them to the common center.

$$A = \frac{Q.P^T}{n} \tag{1}$$

$$UDV^T = SVD(A) \tag{2}$$

- Rotation = V.U^T
- Scale Factor(SF) = $\frac{Trace(D)}{Sum(X,Y,Z)}$
  Scale = [SF, SF, SF]
- Translation = Centroid(P) - Centroid(Q).(SF x Rotation)

Implementation

The network is pre-trained with two different datasets:

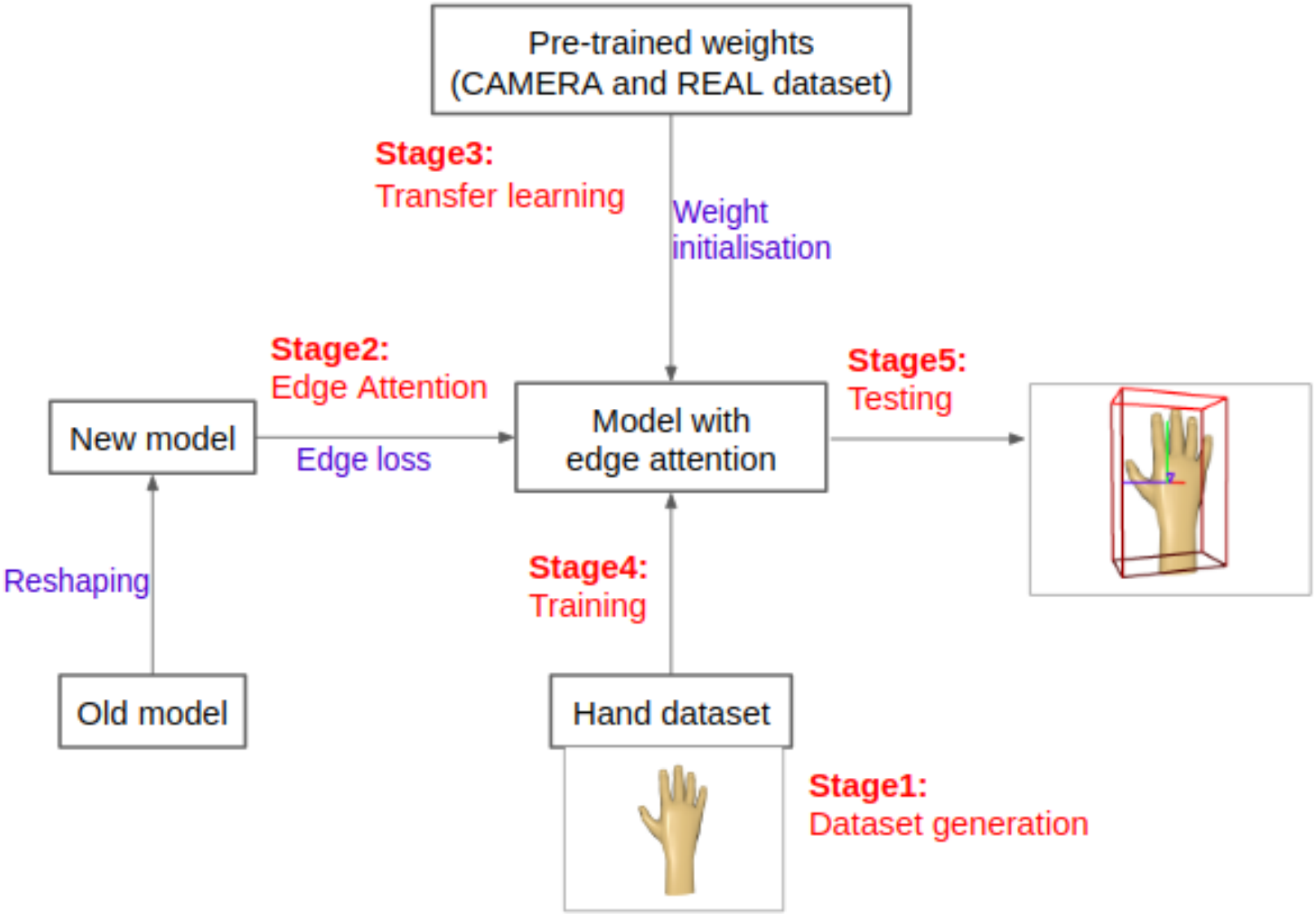
1. CAMERA dataset(Context Aware Mixed Reality Approach):
   - 6 categories of objects(bottle, bowl, camera, can, laptop, and mug) are synthetically placed in real background with different lighting conditions.
   - It consists of 275K training and 25K testing images.
2. REAL-world dataset:
   - It consists of 4300 training, 950 validation and 2750 testing images.

The training is done in 3 stages.
Stage-1: The network head layers are trained for 100 epochs at a learning rate of 0.001.
Stage-2: Layers above stage 4+ are trained for 130 epochs at a learning rate of 0.0001.
Stage-3: All layers together are trained for 400 epochs at a learning rate of 0.00001.

Key contributions of the research work

- Understanding and implemention of the Normalized Object Coordinate Space(NOCS) architecture.
- The creation of a synthetic Hand Dataset using blender software comprising of 1900 training, 300 validation, and 300 testing images.
- Training the NOCS model with the synthetic Hand Dataset using the transfer learning.
- Extensive analysis of the 6D poses obtained from the model using various metrics of evaluation.
- 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.
- Training the model with different Edge loss filters and comparing their performances to find the optimal edge filter.

Flowchart of the proposed work



Stage 1: Hand Dataset generation

- Our current problem in the **humanoid** project requires the identification of the 6D pose of the **hand of the robot** in various orientations and lighting conditions.
- We have used a software called **Blender** for creating the dataset.
- It is a free and open-source 3D computer graphics software used for creating animated films, visual effects, 3D printed models, computer games etc.

Contents of our dataset :

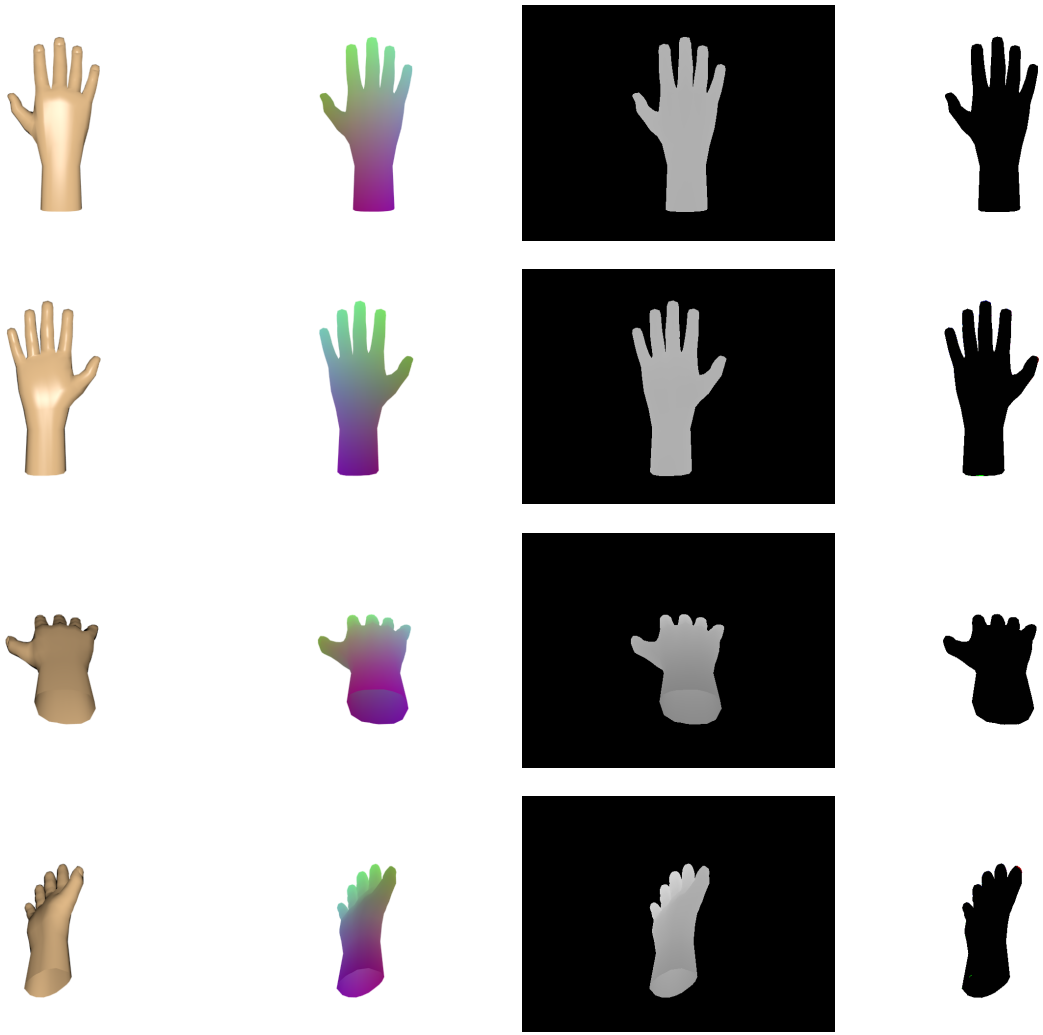
1. RGB hand images
2. Depth maps
3. Ground truth NOCS maps
4. Ground truth Instance masks
5. Ground truth Labels

- A code in python was written using the python-blender library.
- The code outputs RGB, depth and ground truth instance mask and NOCS maps with different orientations when fed with an object's CAD model(.obj files).
- The training dataset consists of 1900 images and validation dataset has 300 images and testing dataset has 300 images comprising of 125 different orientations, 5 different lighting setups and 3 different scales.

Sample images

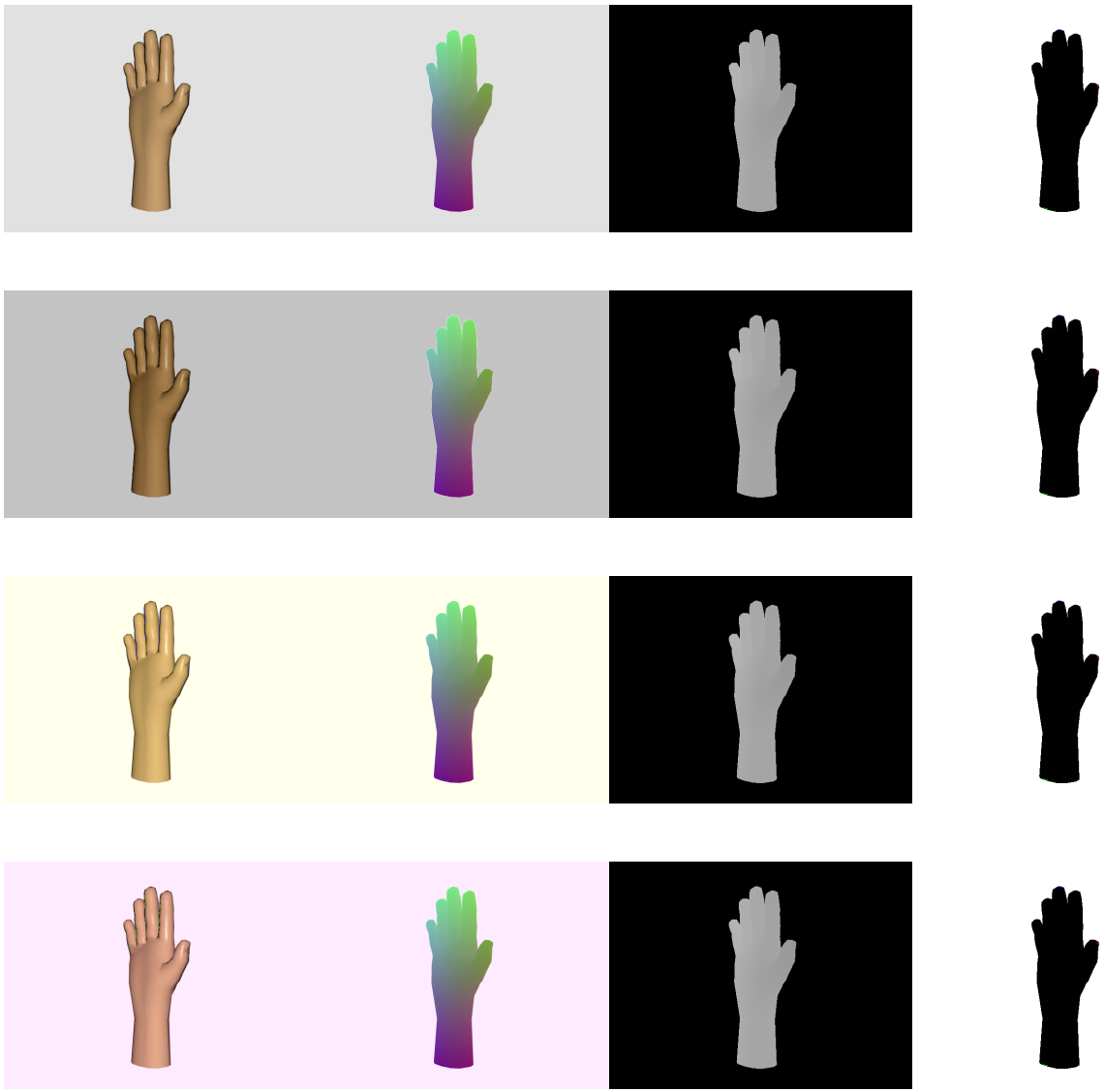
Different orientations

Colour NOCS Depth Instance mask

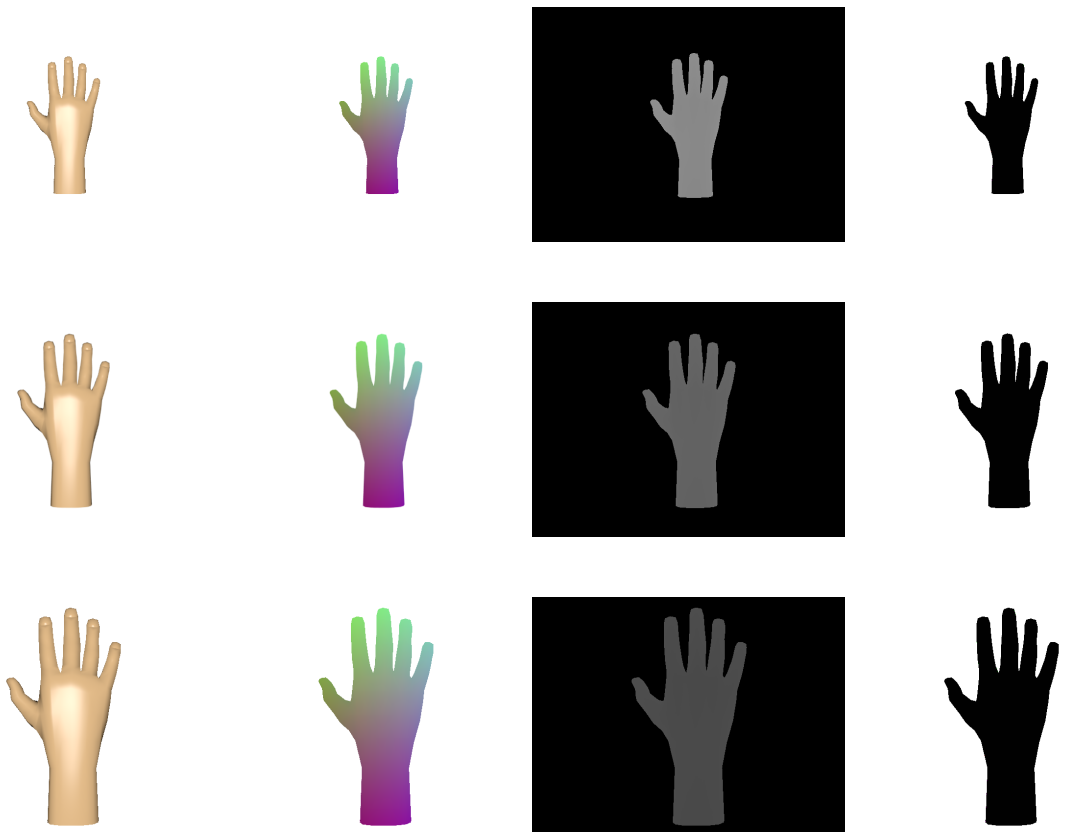


Different lighting conditions

Colour NOCS Depth Instance mask



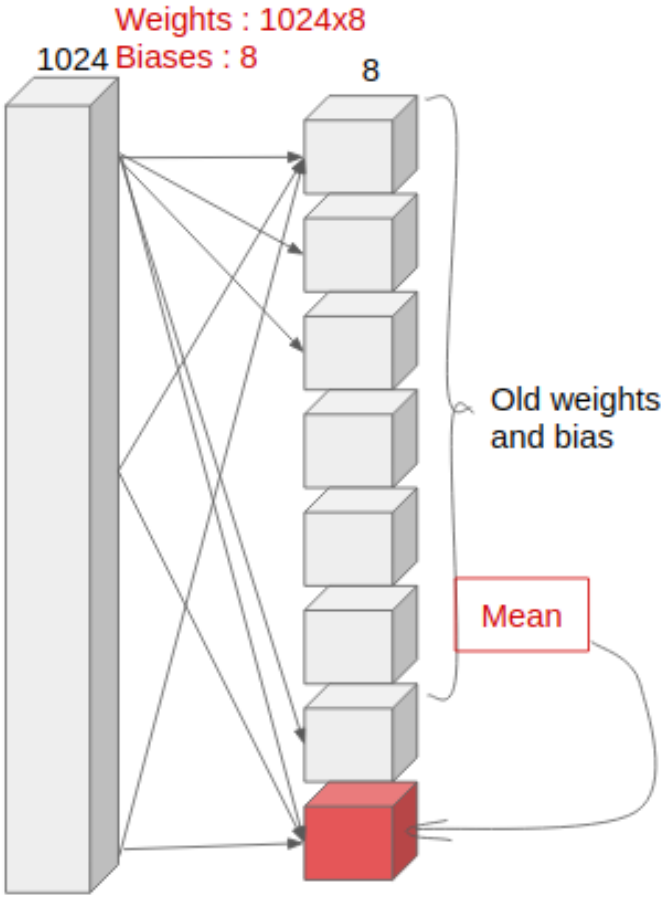
Different scales

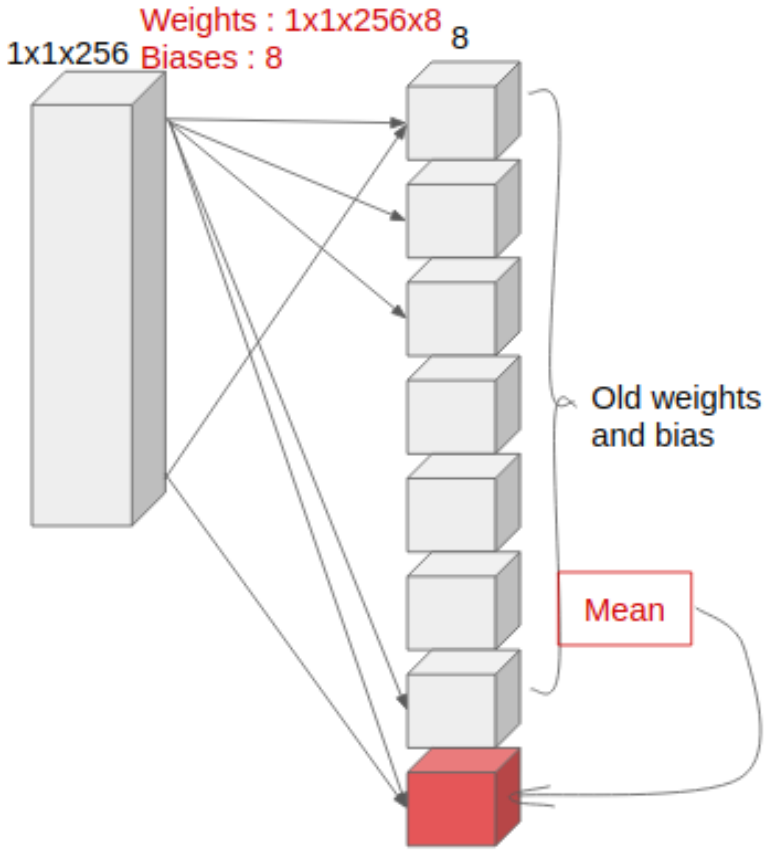


Stage 3: Transfer learning - weight initialization

- Model = backbone + head layers. The backbone layers perform feature extraction from the images and head layers perform end tasks like classification, instance mask and NOCS prediction.
- New model - The backbone of the old model is retained while the head layers are changed to accommodate one extra class.
- The model was already trained for 6 object classes(bottle, bowl, camera, can, laptop, and mug) using a huge collection of datasets. We want to retain the information the model has learnt from these datasets and train over that with the hand dataset. For this purpose, we use a technique called **transfer learning**.
- Transfer learning is storing the knowledge gained while solving one problem and applying it to a different but related problem.

Stage 3 : Transfer learning - weight initialization





Stage 4: Training

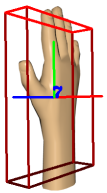
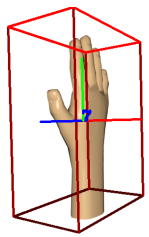
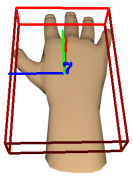
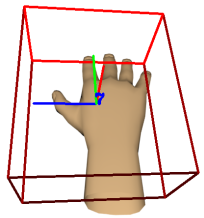
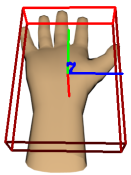
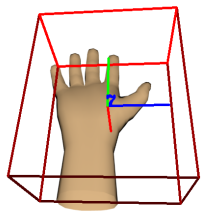
2. Our training dataset consists of **alternating hand images and pre-trained images** so that, while training, the network doesn't get biased towards hand images.

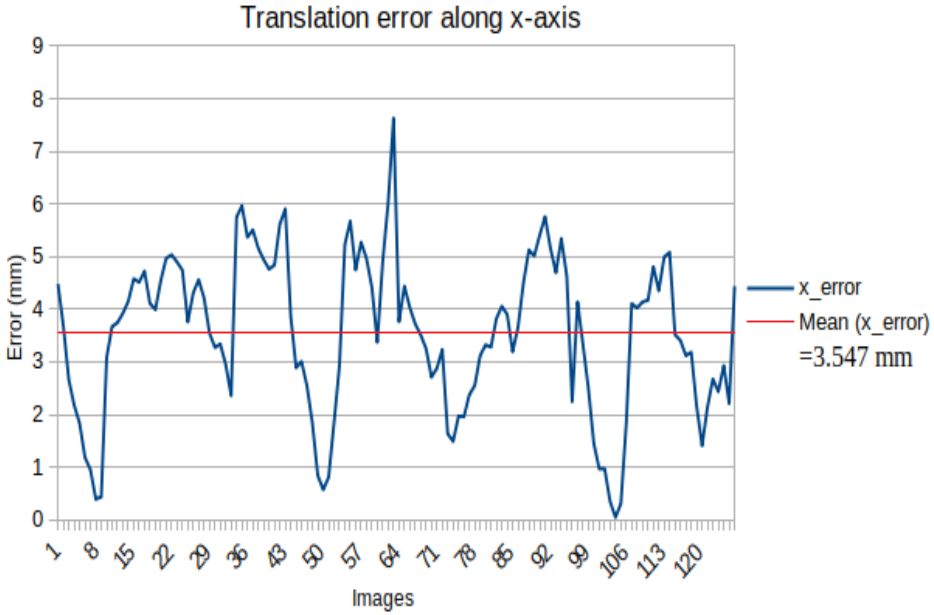
- So for training, we initialize the model with the pre-trained weights using transfer learning, we train the head layers alone for 70 epochs, at a learning rate of 0.001, weight decay of 0.0001 and 1000 steps per epoch.
- The training of the model takes 1min 5secs/epoch in the NVIDIA Tesla V100 system.
- Testing takes about 5.39s per image.

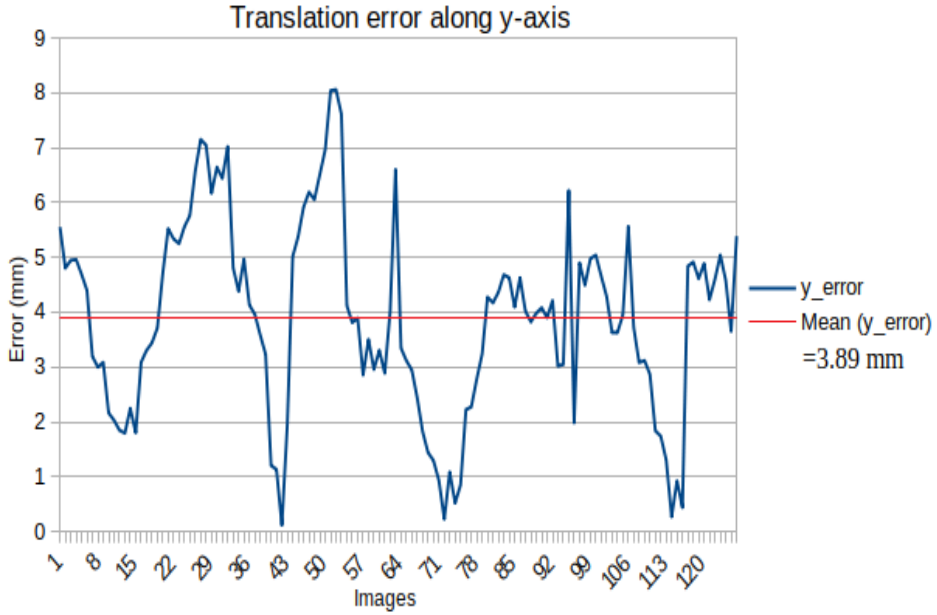
Stage 5: Testing

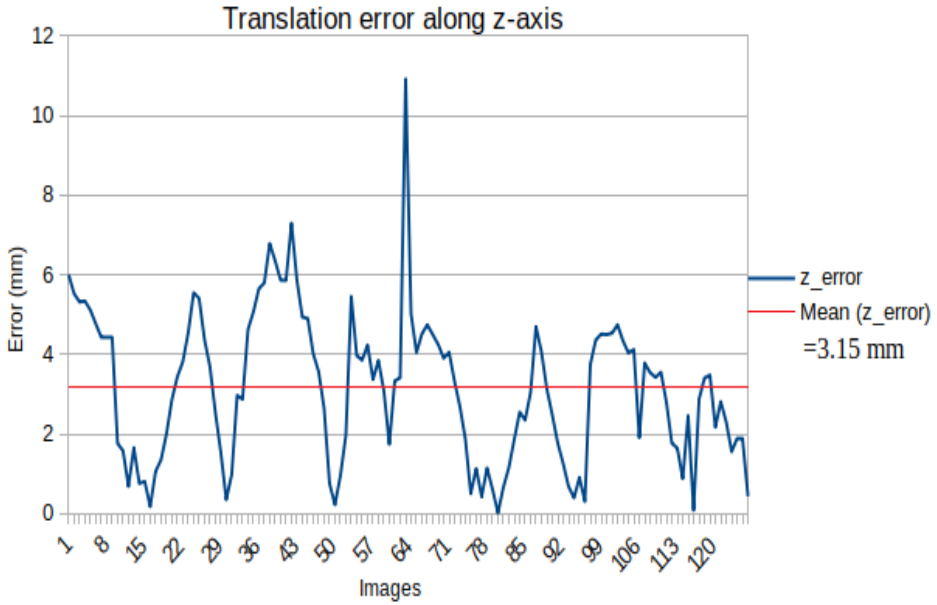
We have tested the models with hand images of different orientations, lightings and scale.

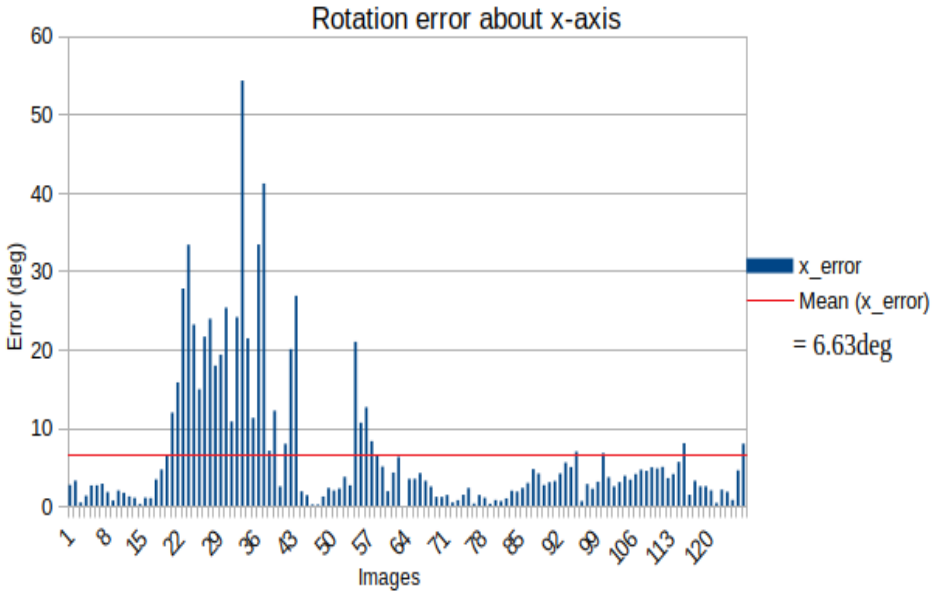
Different orientations

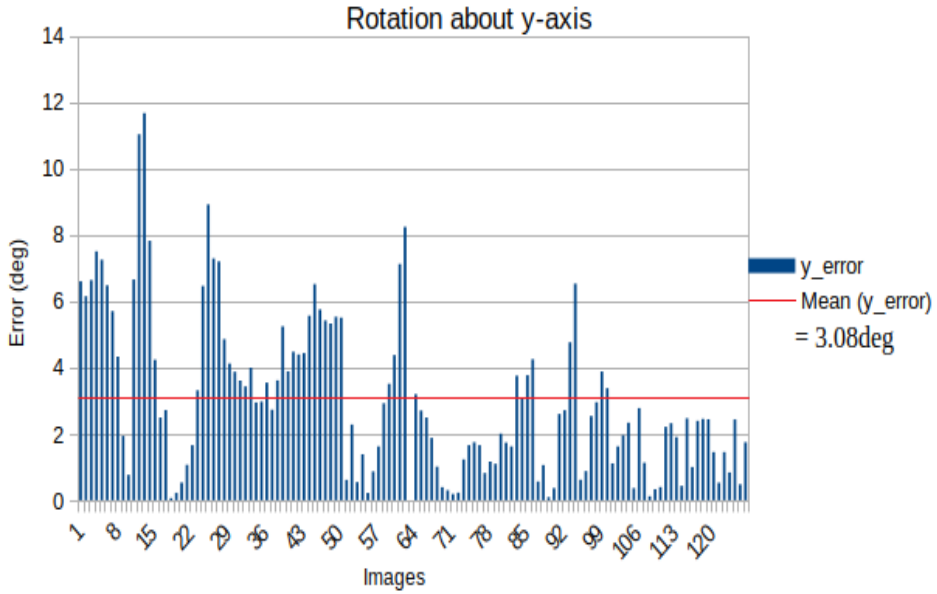
Ground truth		Predicted	
	**T:** (0.1, 0, 17.08) **R:** (1.1,-34.84,-179.9)		**T:** (0,0.36,16.84) **R:** (0.36,-32.9,-178.6)
	**T:** (0, 0, 16) **R:** (44.6,-87.3,-178.1)		**T:** (0.4,0.6,15.84) **R:** (35.3,-83.2,-163.8)
	**T:** (0, 0, 15.85) **R:** (-44.1,87.5,-176.7)		**T:** (-0.4,0.4, 16.48) **R:** (-38,82.3,-169.52)

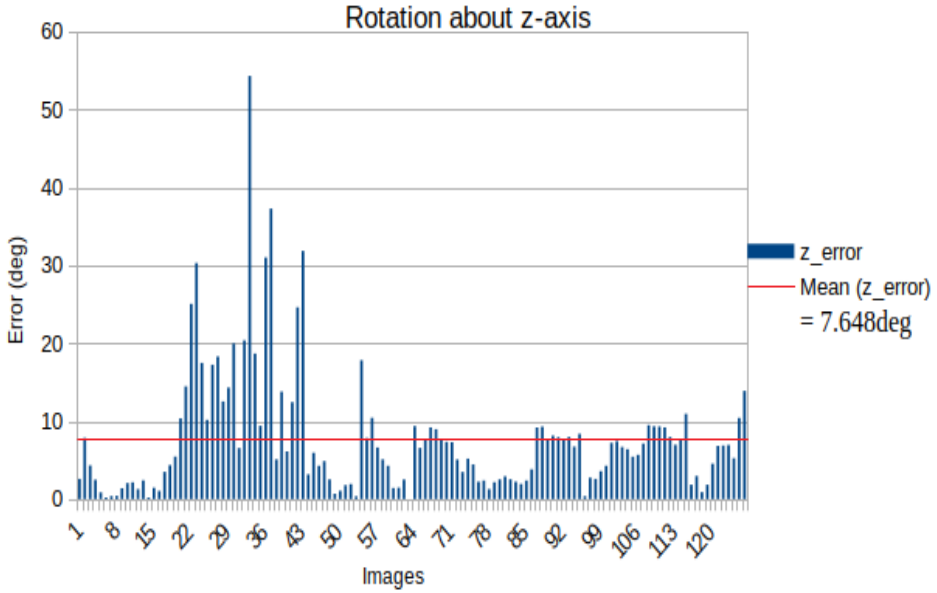

Translation error along x-axis
Error (mm)
Images
x_error
Mean (x_error)
=3.547 mm


Translation error along y-axis
Error (mm)
Images
y_error
Mean (y_error)
=3.89 mm


Translation error along z-axis
Error (mm)
Images
z_error
Mean (z_error)
=3.15 mm


Rotation error about x-axis
Error (deg)
Images
x_error
Mean (x_error)
= 6.63deg


Rotation about y-axis
Error (deg)
Images
y_error
Mean (y_error)
= 3.08deg


Rotation about z-axis
Error (deg)
Images
z_error
Mean (z_error)
= 7.648deg

Anomalies

Figure: 29

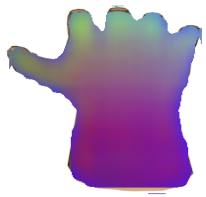
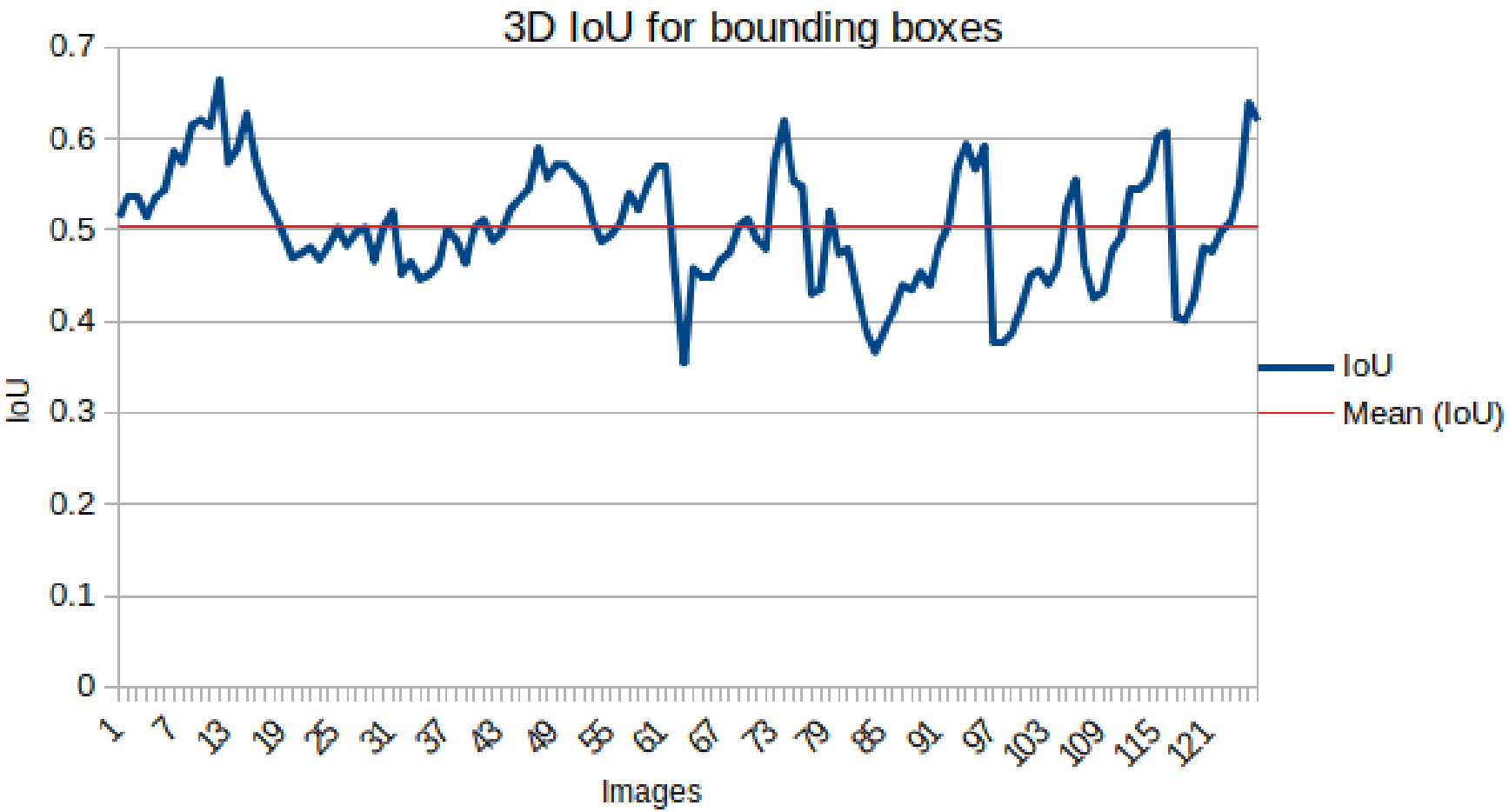


Figure: 29

Figure: 32

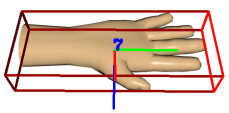
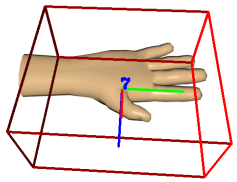
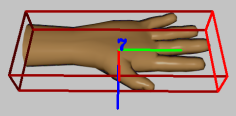
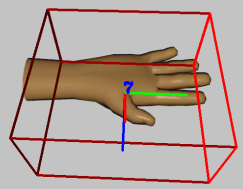
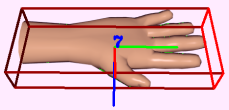
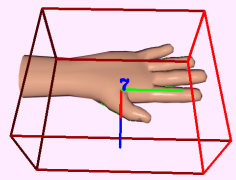
Figure: 32

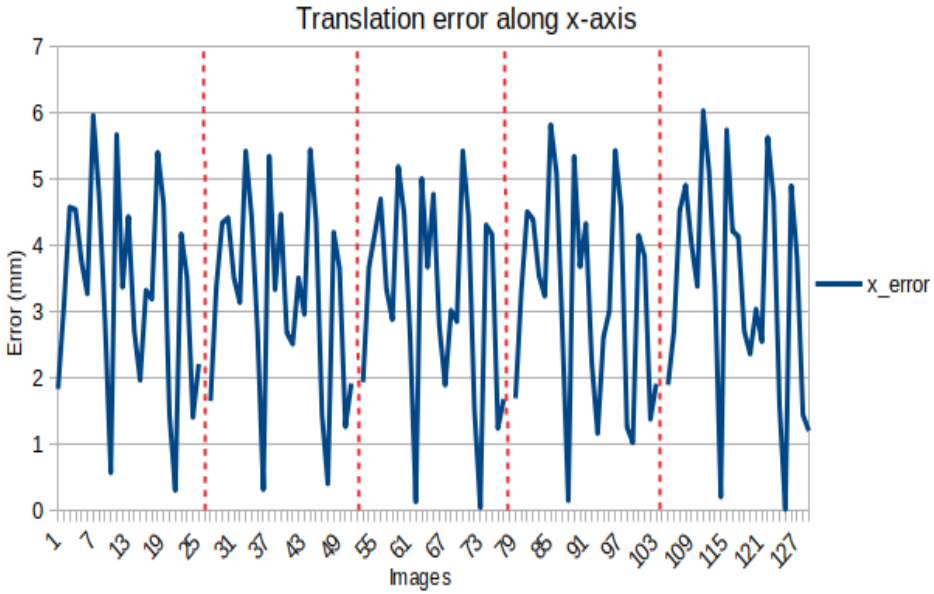
IoU-Intersection/Union

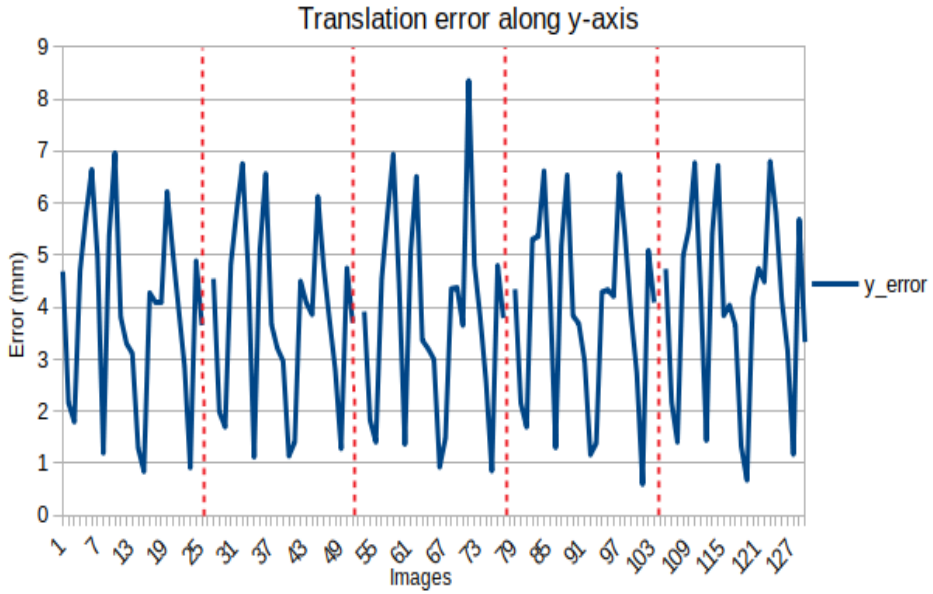


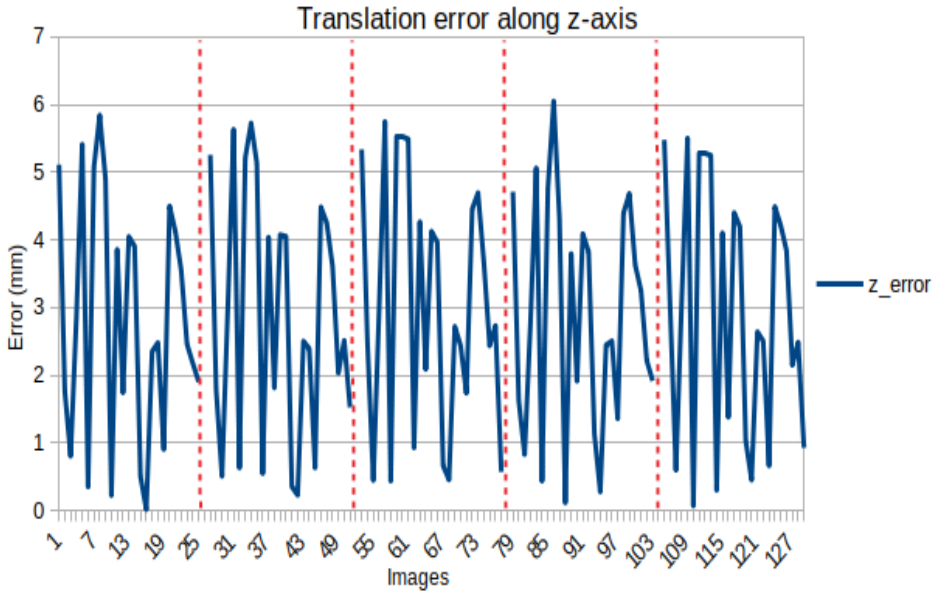
Testing results

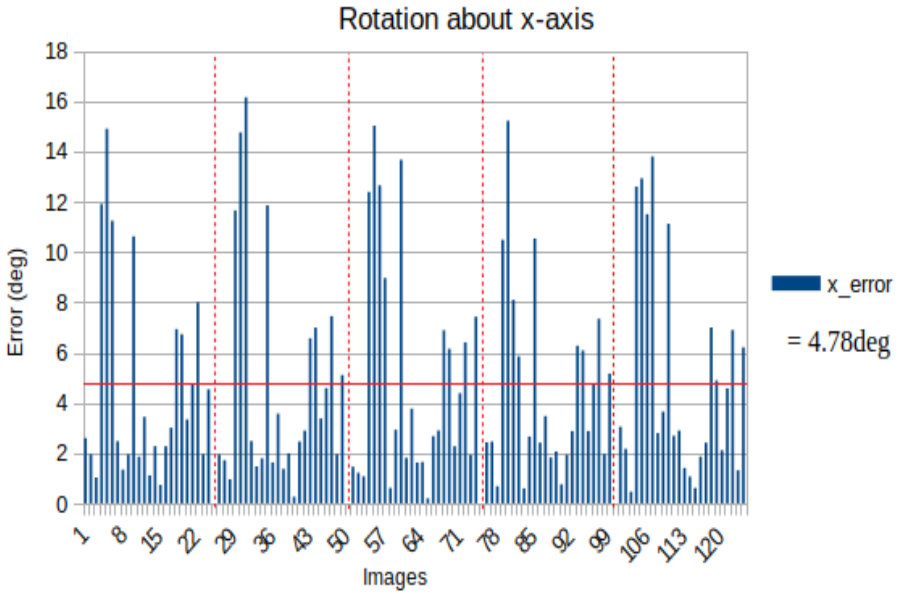
Different lightings

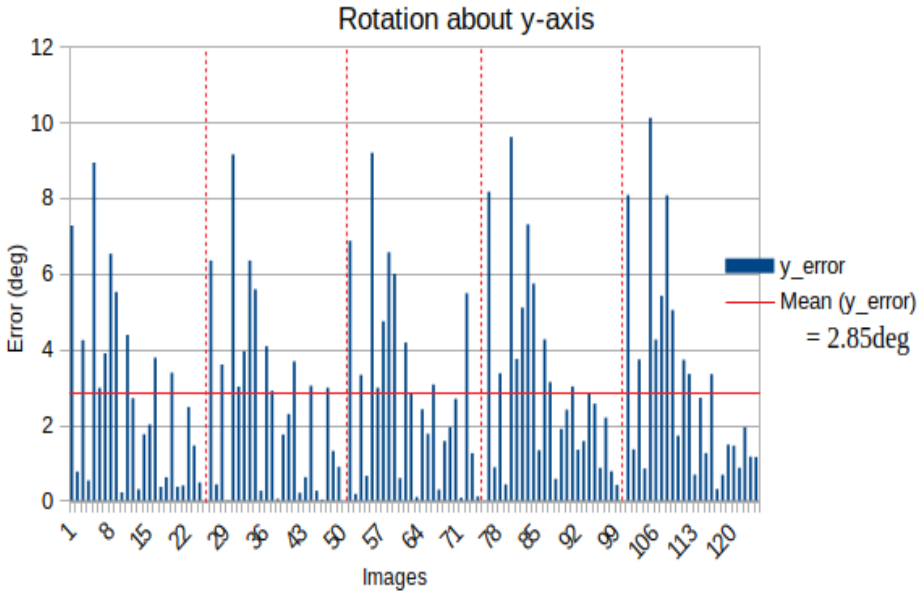
Ground truth		Predicted	
	**T:** (0, 0.15, 16.25) **R:**(-89.7,0.25,-135.6)		**T:** (-0.28,-0.28,16.25) **R:**(-88.9,-1.7,-138.1)
	**T:** (0, 0.15, 16.25) **R:**(-89.7,0.25,-135.6)		**T:** (-0.25,-0.3,16.29) **R:**(-89.8,-1.5,-137.7)
	**T:** (0, 0.15, 16.25) **R:**(-89.7,0.25,-135.6)		**T:** (-0.25,-0.27,16.29) **R:** (-89,-0.99,-137.9)

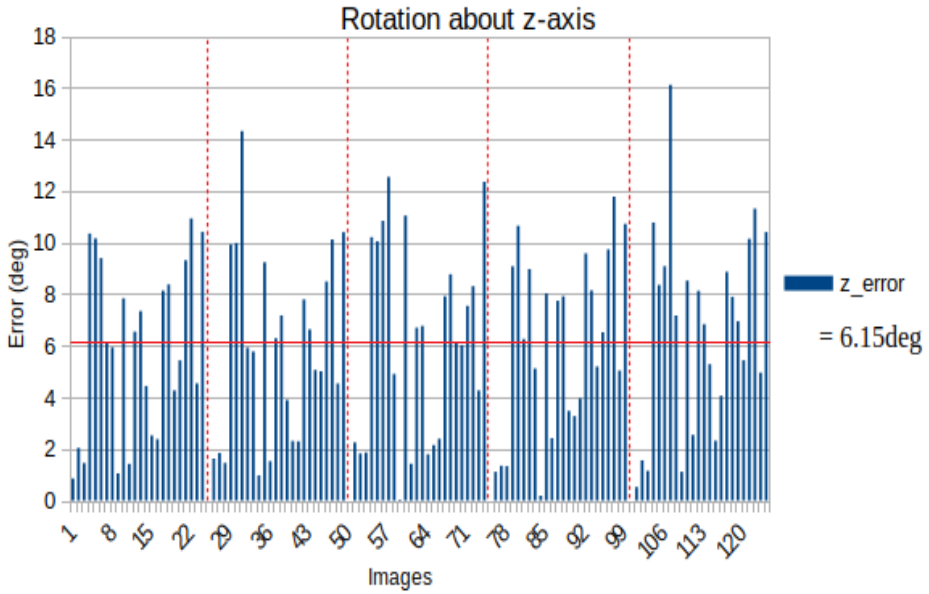

Translation error along x-axis
Error (mm)
Images
x_error


Translation error along y-axis
Error (mm)
Images
y_error

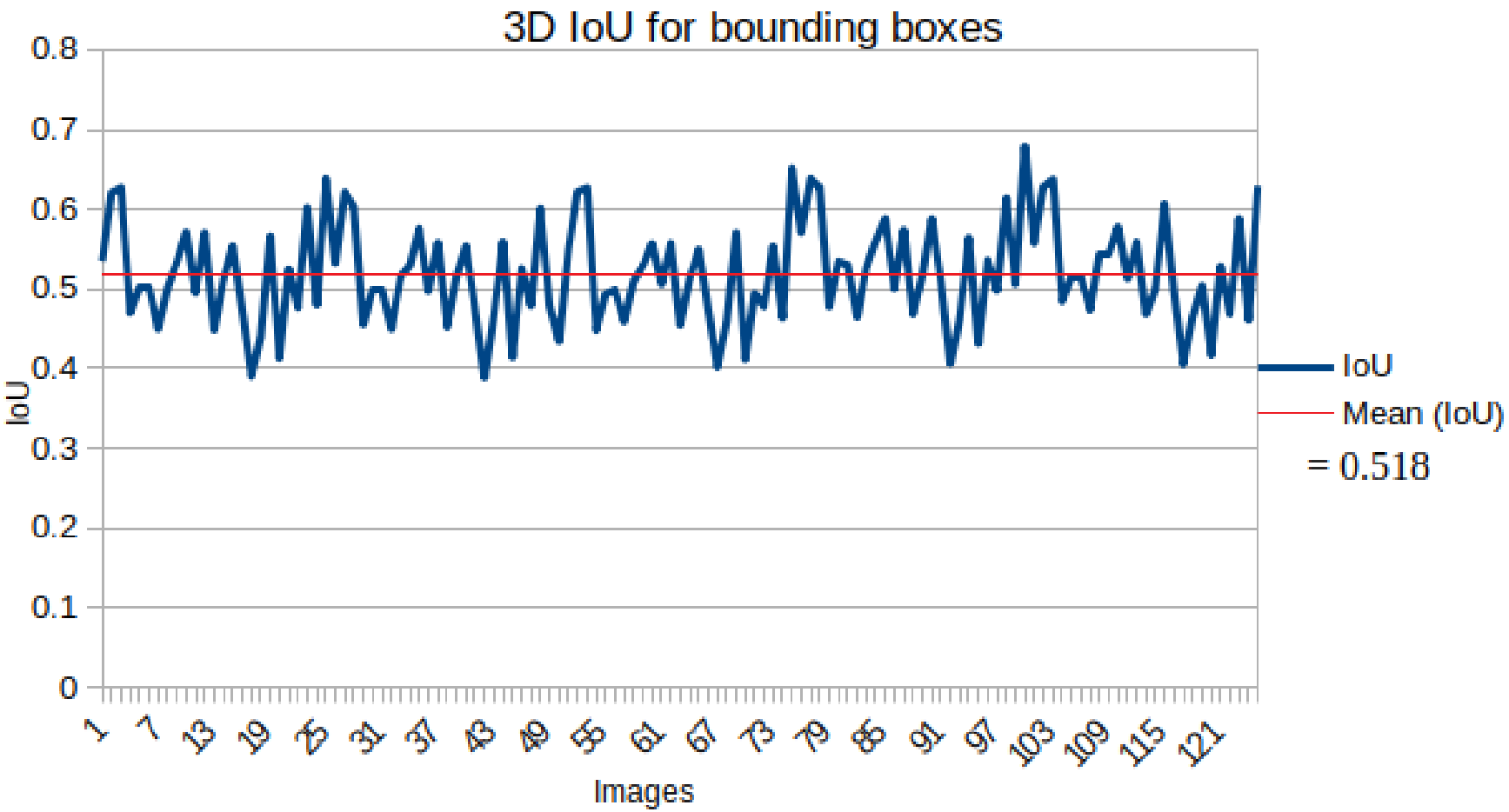

Translation error along z-axis
Error (mm)
Images
z_error


Rotation about x-axis
Error (deg)
Images
x_error
= 4.78deg


Rotation about y-axis
Error (deg)
Images
y_error
Mean (y_error)
= 2.85deg


Rotation about z-axis
Error (deg)
Images
z_error
= 6.15deg

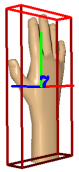
IoU-Intersection/Union



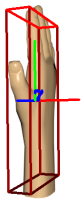
Testing results

Different scales

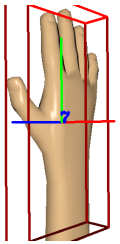
Ground truth



T: (0, 0.15, 16.25)
R:(-89.7,0.25,-135.6)

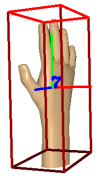


T: (0, 0.15, 16.25)
R:(-89.7,0.25,-135.6)

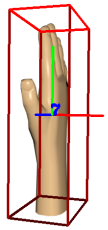


T: (0, 0.15, 16.25)
R:(-89.7,0.25,-135.6)

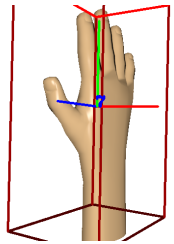
Predicted



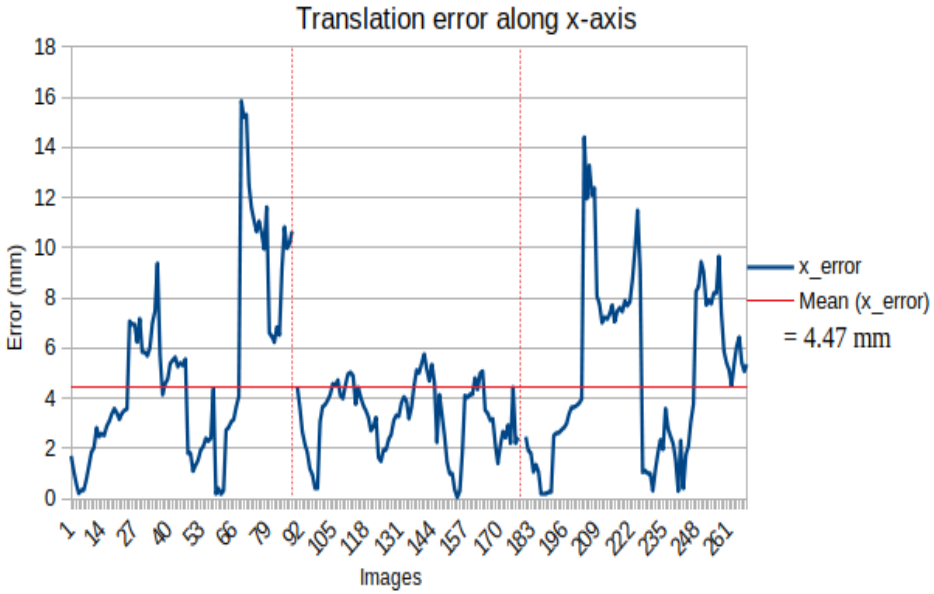
T: (-0.28,-0.28,16.25)
R:(-88.9,-1.7,-138.1)

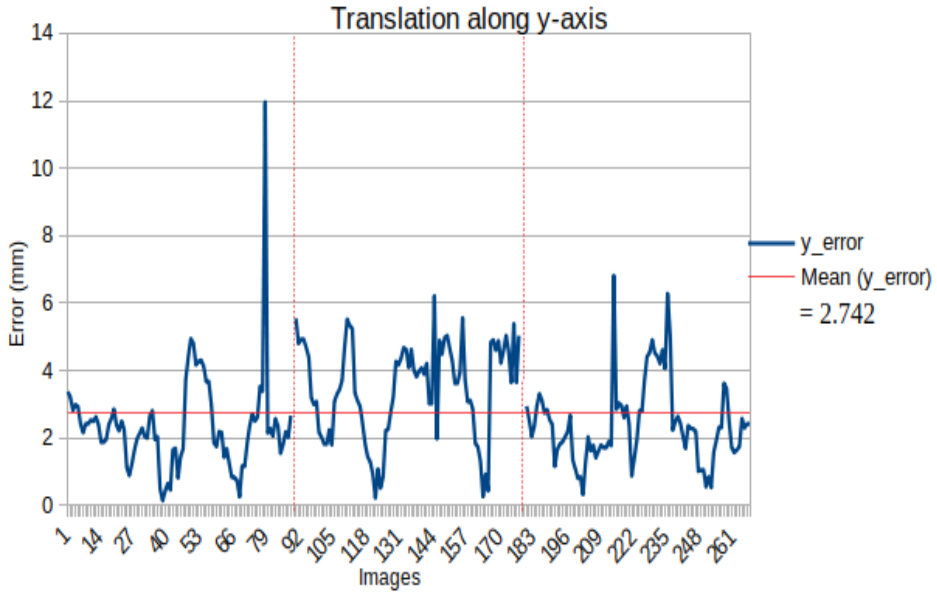


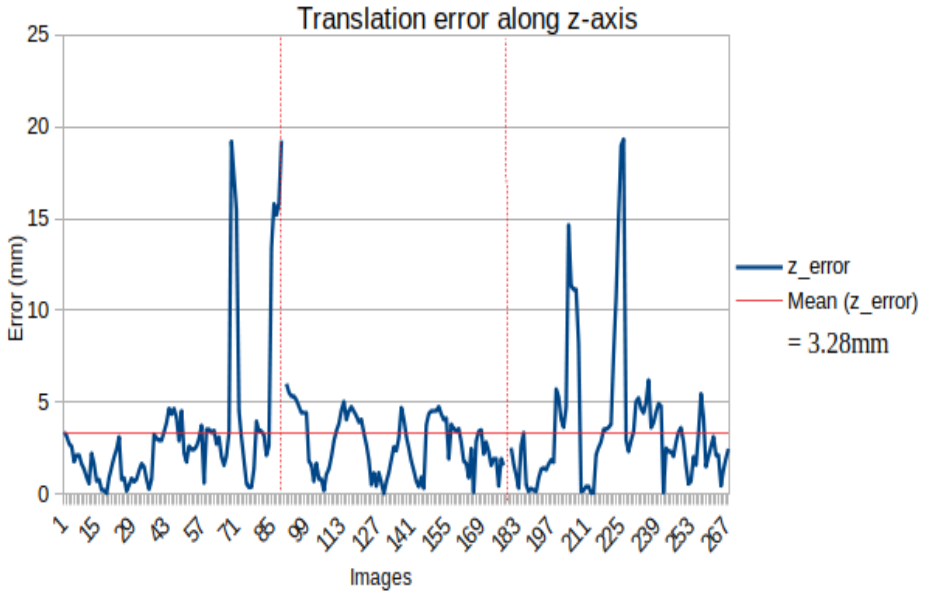
T: (-0.25,-0.3,16.29)
R:(-89.8,-1.5,-137.7)

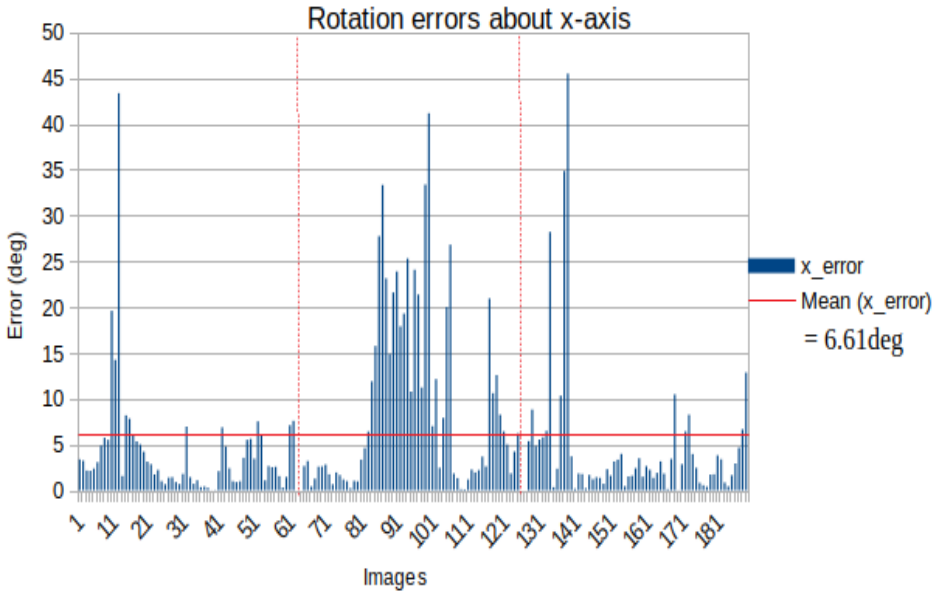


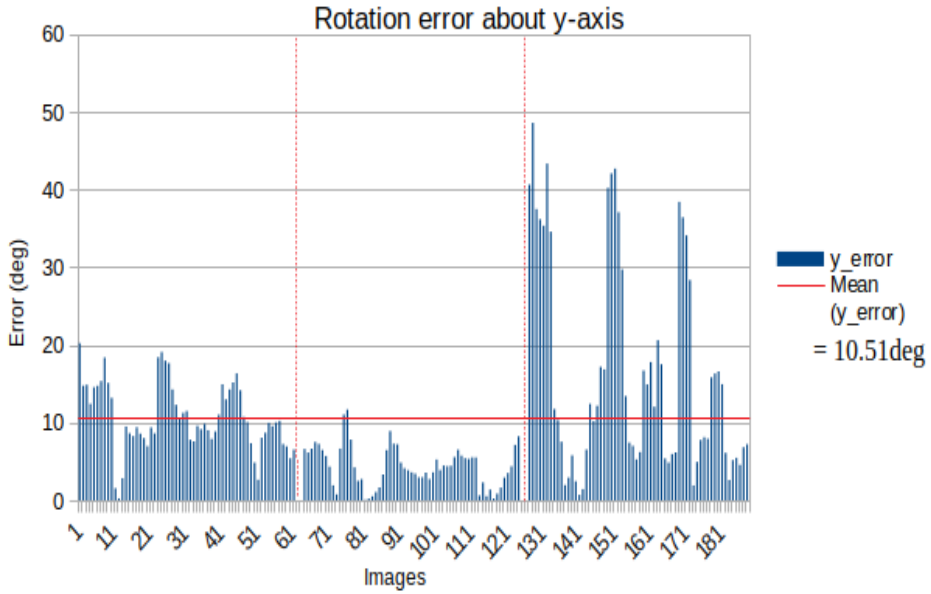
T: (-0.25,-0.27,16.29)
R: (-89,-0.99,-137.9)

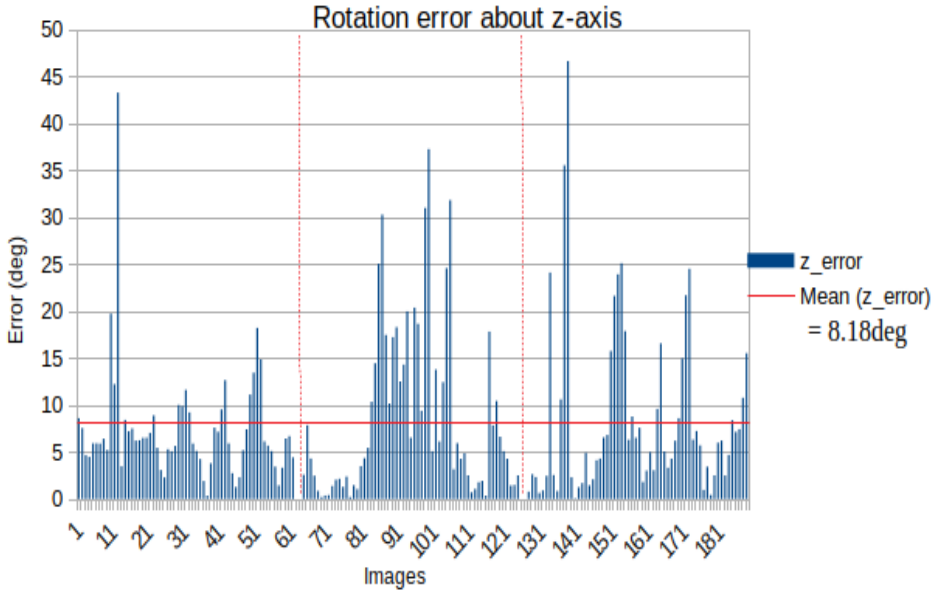

Translation error along x-axis
Error (mm)
Images
x_error
Mean (x_error)
= 4.47 mm


Translation along y-axis
Error (mm)
Images
y_error
Mean (y_error)
= 2.742


Translation error along z-axis
Error (mm)
Images
z_error
Mean (z_error)
= 3.28mm


Rotation errors about x-axis
Error (deg)
Images
x_error
Mean (x_error)
= 6.61deg


Rotation error about y-axis
Error (deg)
Images
y_error
Mean
(y_error)
= 10.51deg

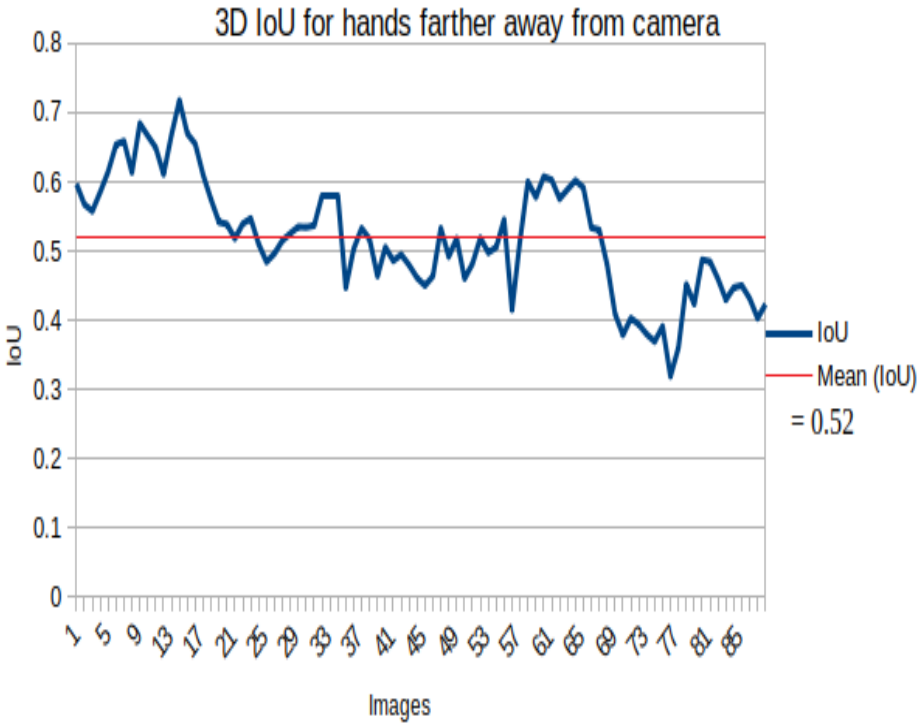

Rotation error about z-axis
Error (deg)
Images
z_error
Mean (z_error)
= 8.18deg

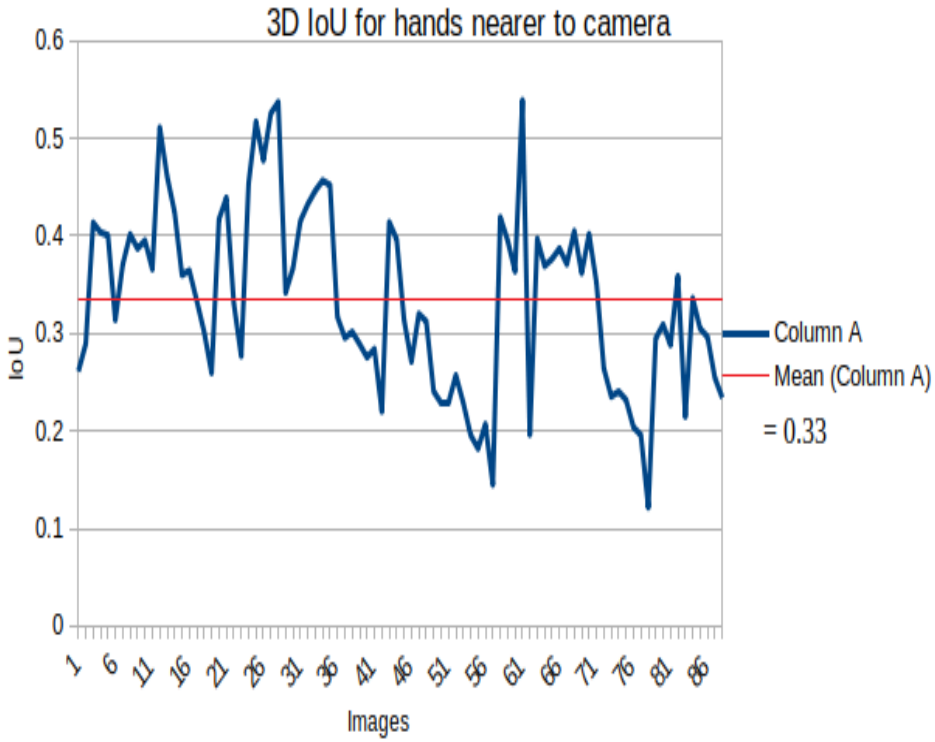
Anomalies






IoU-Intersection/Union

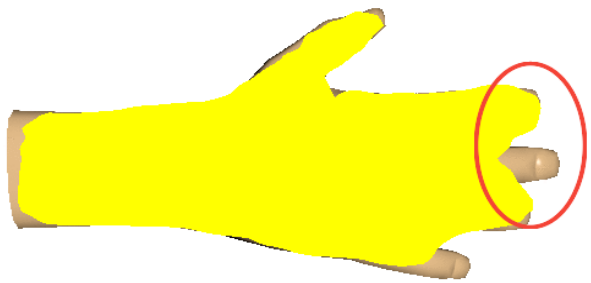




Mask prediction errors

Ground truth

Predicted



Edge-Agreement-Loss

- An Edge Agreement Head is added in the model that uses edge detection filters on instance maps to predict the edges of an object.
- Edge Agreement Loss = Mean square error between ground truth and predicted edge maps.

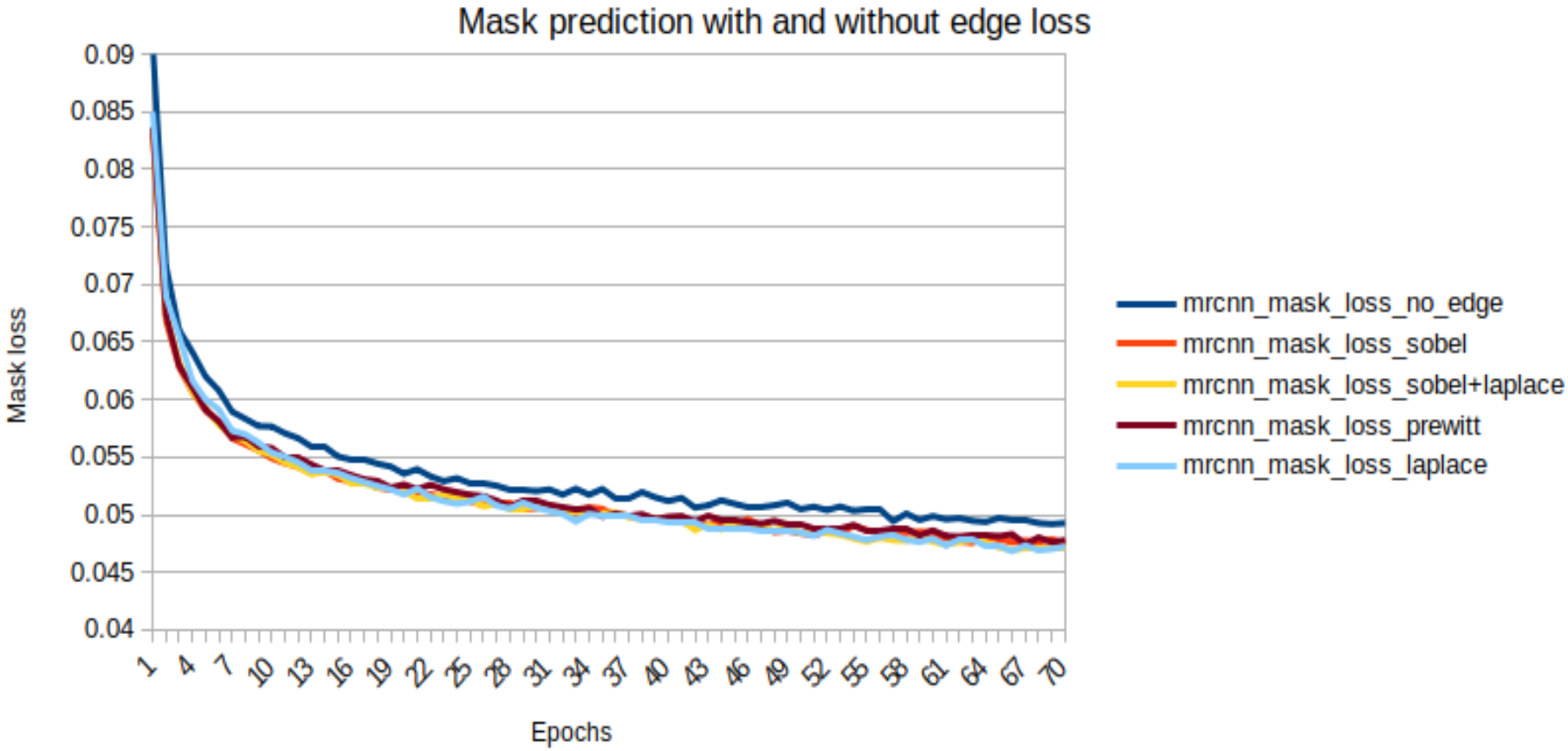
$$\therefore L_{edge}(\tilde{y}, y) = Mean[(\tilde{y} - y)^2] \tag{3}$$

, where y is the ground truth edge pixel and is the predicted edge pixel.

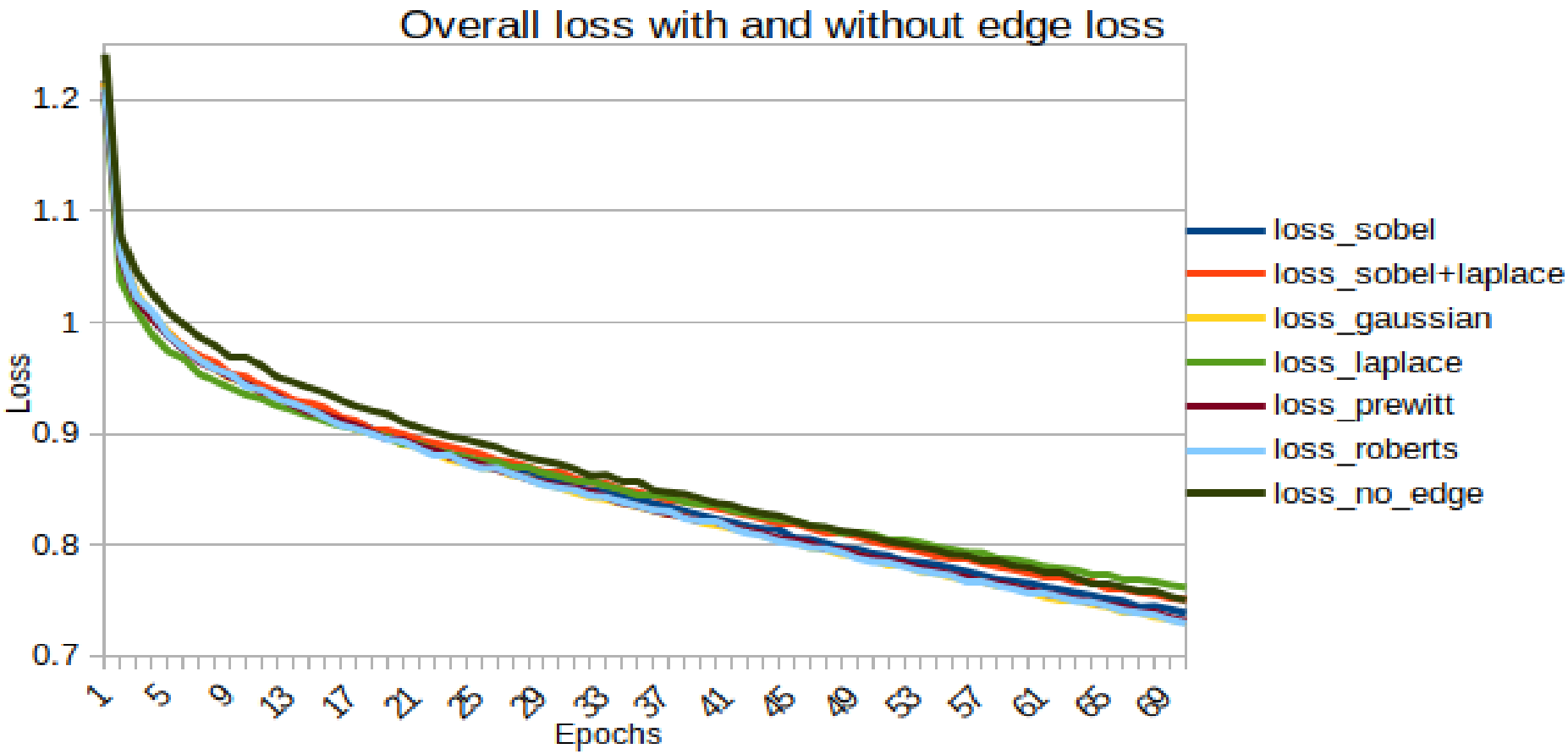
$$L_{Mask}^{new} = L_{Mask} + L_{Edge} \tag{4}$$

- This increases the attention of the edges in objects and hence improves the accuracy of instance mask prediction.

- We have implemented the edge agreement loss using various edge detection filters like sobel filters, laplacian, prewitts filters etc.



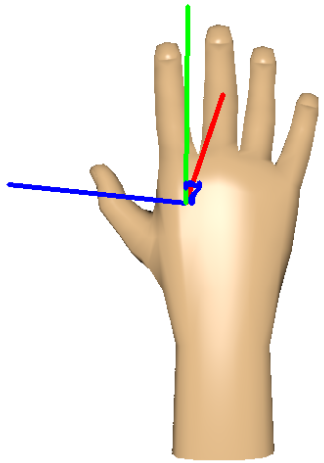
We observe a **4%** decrease in mask loss at the end of 70 epochs.

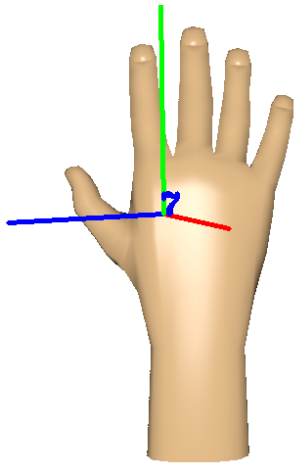


We observe a **2.8%** decrease in the overall loss at the end of 70 epochs.

Without edge loss

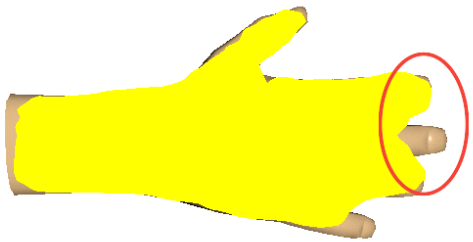
With edge loss

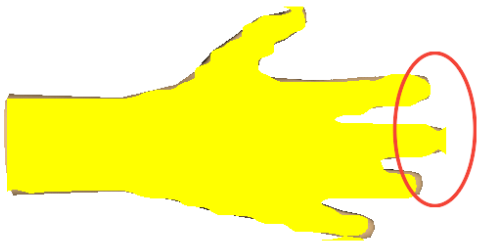


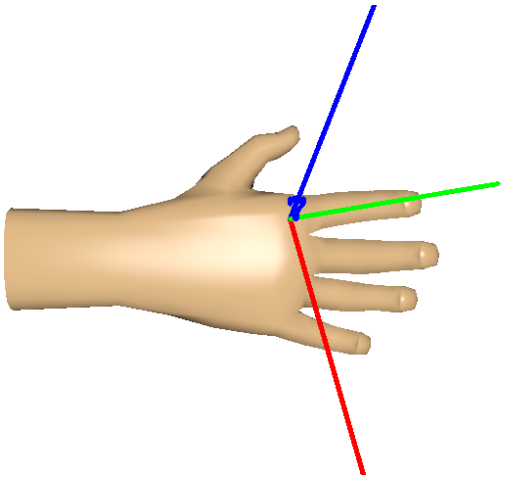


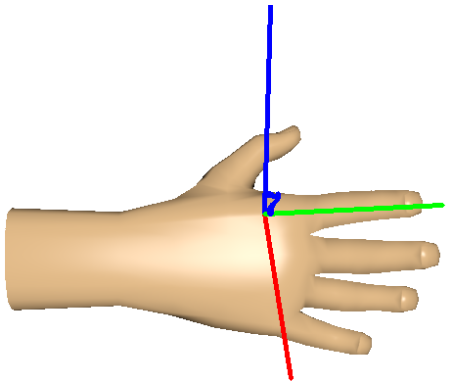
Without edge loss

With edge loss









NOCS prediction errors

Ground truth

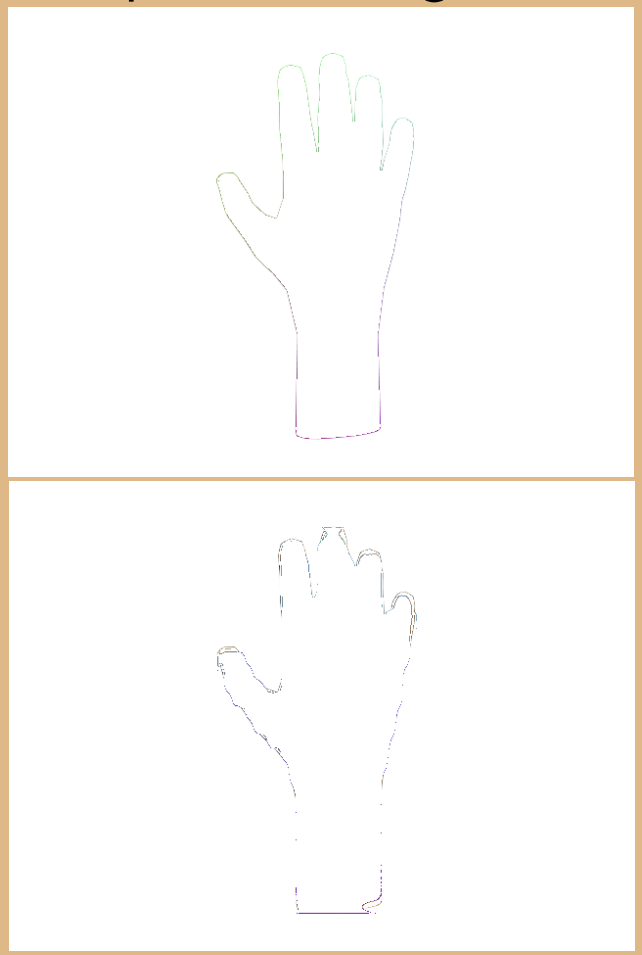
Predicted

NOCS edge detection

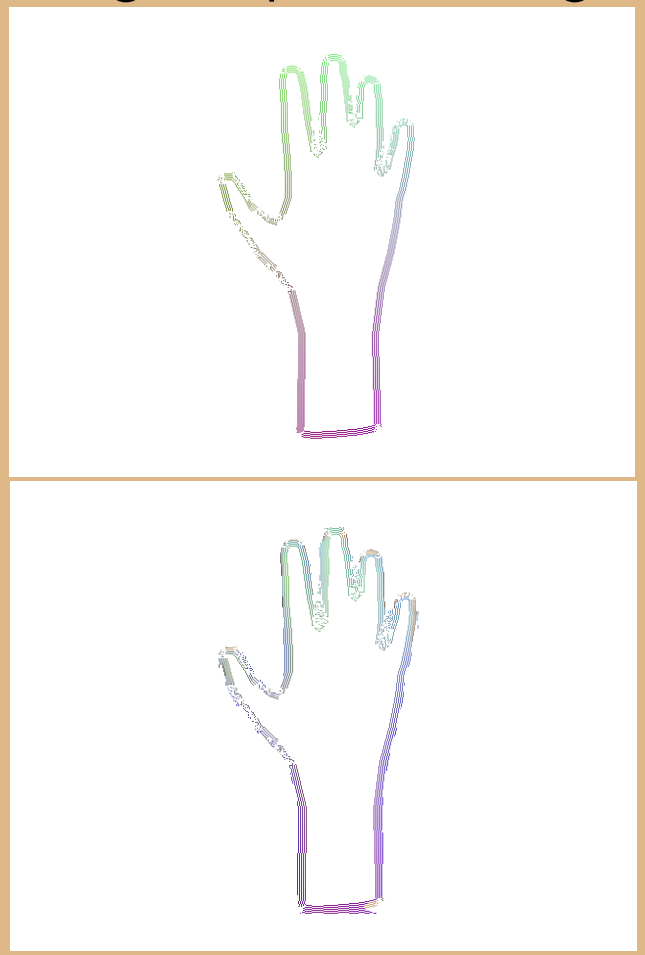
Idea proposal:

- Apply edge detection filter on the NOCS maps.
- Applying the filter multiple times increases the thickness of the edges.
- Find the softmax loss between the NOCS edge maps and add this loss to the overall NOCS loss.
- This will increase the attention on the edges of the NOCS maps and will improve its prediction accuracy.

Edge map after 1 edge filter



Edge map after 5 edge filter



Conclusion

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.

5. 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%.
6. 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 my Github page: `https://github.com/AnjuVolt/EdgeAttention_6Dpose_estimation`

Future works

- Edge Agreement Loss on NOCS maps to improve accuracy of NOCS map prediction.
- The bounding boxes predicted by our model are slightly scaled up when compared to ground truth.
- The model's performance has not yet been tested on occluded images, where objects are only partially visible.

References

- Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, *Mask r-cnn*, Proceedings of the IEEE international conference on computer vision, 2017, pp. 2961–2969.
- Bugra Tekin, Sudipta N Sinha, and Pascal Fua, *Real-time seamless single shot 6d object pose prediction*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 292–301.
- 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*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2642–2651.
- 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).

Thank You