AIKO Space Internship Report

Amirshayan Nasirimajd
amirshayan.nasirimajd@studenti.polito.it

***Abstract*—During Summer I worked three month over the topic of the space craft pose estimation using the multi-task learning. My main duty wast to implement the Robust Multi-Task Learning and Online Refinement for Spacecraft Pose Estimation across Domain Gap method (SPNv2) and reproduce the results over this method.**

I. Introduction

6D pose estimation is the problem of detecting the position of a 3D object, which includes its location and orientation. Recently, this problem has become more noticeable in pose estimation for noncooperative spacecraft to find the position of spacecraft according to a monocular-based camera. Using Monocular-based pose estimation in spacecraft brings the possibility of facing several issues such as the refueling of space assets, active debris removal, and spacecraft docking.
Using Convolutional Neural Networks (CNN) has recently become more prevalent in solving the pose estimation problem. Therefore, European Space Agency (ESA), with cooperation from the Space Rendezvous Laboratory (SLAB), released a dataset known as SPEED, which contains different pictures of Tango spacecraft in different positions. This dataset is used for Pose Estimation Challenge 2019.
Moreover, Multi-task learning is a recent method in computer vision, which able us to train a model which is able to do several tasks such as classification, segmentation, or detection using several heads and a part of shared parameters. In addition, the loss of each head can backpropagate to improve the overall performance of shared parameters. As a result, Using this idea, Park et al. [1] introduce a multi-modal method to face pose estimation of monocular-based noncooperative spacecraft using several heads to do the segmentation, pose estimation, and heatmap calculation.
Furthermore, since the data available in the SEEPD dataset is syntactic, due to the domain gap between synthetic and real data in the [2], online domain refinement has been introduced to reduce this domain gap and increase the accuracy.
Here, we tried to reproduce the result of this method for spacecraft pose estimation in several steps. It is worth mentioning that the main paper works over SPEED+ datasets, and since we have a limit for processing power and time, we decided to use the SPEED datasets instead a lighter version of SPEED+.

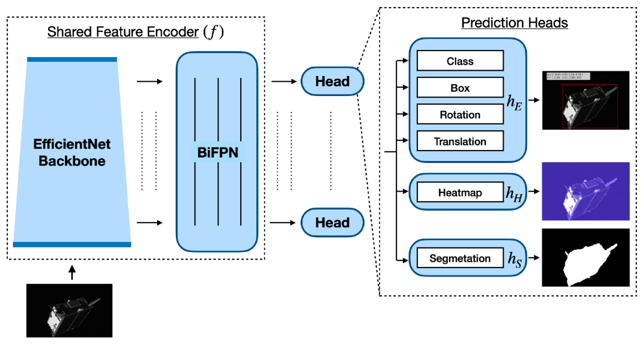

Fig. 1. Offline Multi-task learning part consisting of efficientdet backbone with several head for several tasks.

II. Dataset Overview

The speed datasets [2] consist of 12000 labeled synthetic images of Tango spacecraft in different positions, along with 3000 synthetic and 300 real unlabeled photos with a size of 1920*1200 px. To train data, we use 6000 images of the train data in three partitions, the train data, the validation data,and the test data.
Synthetic images are separated into two-part images with an earth background and solo images of the spacecraft with no background.

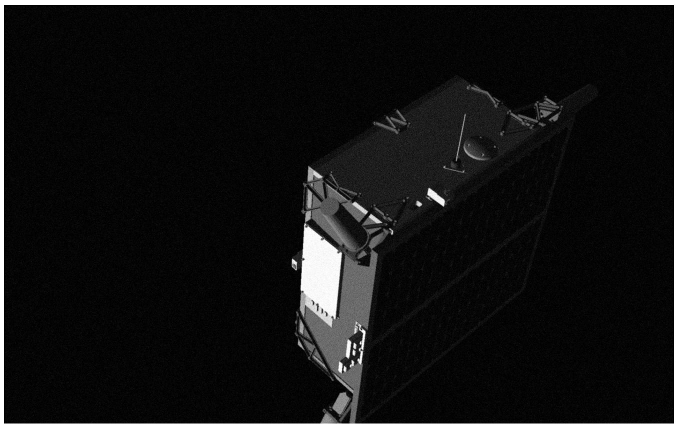

Fig. 2. Sample image of the dataset.

The labels consist of two-parameter first quaternion and relative position. Using these two parameters, we make the labels for the bounding box and 11 main points of the spacecraft (three antennae, four top corners, and four bottom corners). To find these points, we will use a key points file containing the exact coordinates of the spacecraft.

III. Preprocessing

The main step before training is preprocessing. The preprocessing of data happens in several steps. The preprocessing

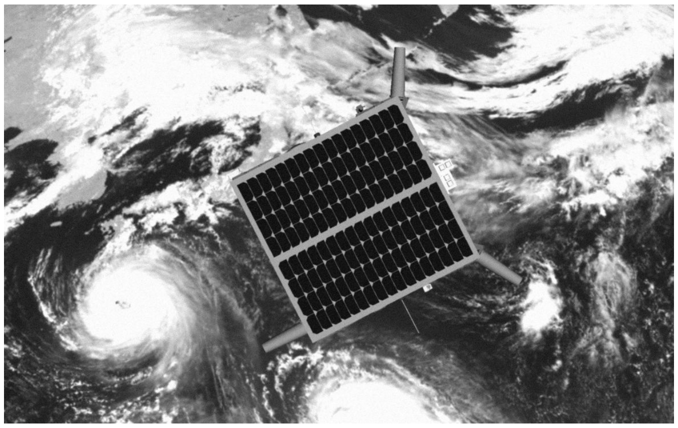

Fig. 3. Sample image with earth background.

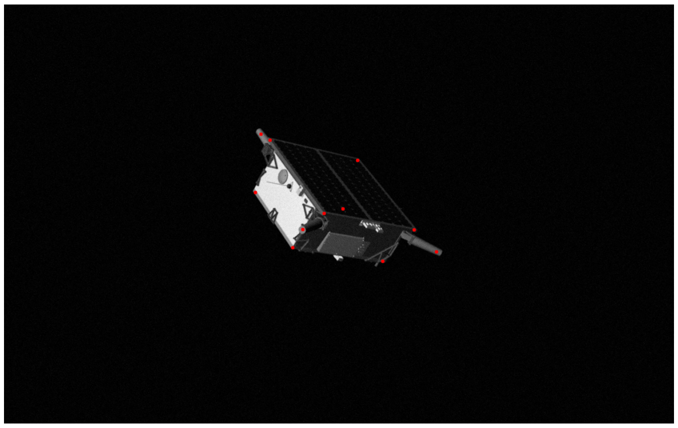

Fig. 5. Highlighted Key points of the spacecraft.

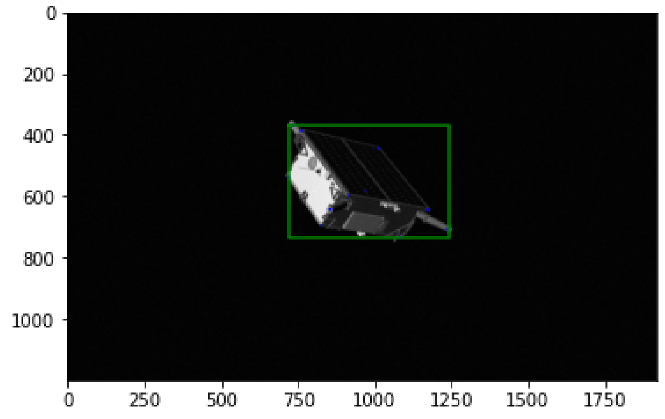

Fig. 4. Bounding Box sample image.

steps are resizing the images, calculating eleven key points on the 2D image, calculating the bounding box, and mask resizing. The preprocessing happens in order to prepare our data and labels in the dataset to be suitable for our method during the training.

- **Resizing:** The Data in the Speed datasets have the size of $1920 \times 1200_{px}$ images. These images were resized to $768 \times 512_{px}$, which minimizes the loss of information due to excessive downscaling and has an approximately matching aspect ratio compared to the original.
- **Calculation of 11 2D Key points:** The following preprocessing step calculates 11 2D key points, which will be projected over the image to define the top corners, four bottom corners, and the top of three antennas of the Tango Spacecraft. Moreover, the 2D key points are not available as a part of labels, and they are needed to create the bounding box and calculate the loss of heatmap, which is one of the main heads of the SPNv2 method. To calculate these key points, 11 3D key points that present the exact points in the real world in XYZ coordinates will be used. These points will be used along with real quaternion and the position of spacecraft, which are available as the main labels of the data in datasets.

To understand better, the following steps provide the 11 2D key points:

  - Calculation of the direction cosine matrix from quaternion (q):

$$q = quaternion/norm(quaternion)$$

$$dcm = \begin{bmatrix} 2q_0^2 - 1 + 2q_1^2 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 \\ 2q_1q_2 - 2q_0q_3 & 2q_0^2 - 1 + 2q_2^2 & 2q_2q_3 + 2q_0q_1 \\ 2q_1q_3 + 2q_0q_2 & 2q_2q_3 - 2q_0q_1 & 2q_0^2 - 1 + 2q_3^2 \end{bmatrix}$$

  - Transformation of 3D points of the object into the image:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} dcm & position \end{bmatrix} \begin{bmatrix} 3DKeyPoints \end{bmatrix}$$

$$x_0 = X/Z$$
$$y_0 = Y/Z$$

  - Apply distortion:

$$r_2 = x_0^2 + y_0^2$$
$$cdist = 1 + dist_0 \times r_2 + dist_1 \times r_2^2 + dist_4 \times r_2^3$$
$$x = cdist \times x_0 + dist_2 \times 2x_0y_0 + dist_3 \times (r_2 + 2x_0y_0)$$
$$y = cdist \times y_0 + dist_2 \times (r_2 + 2x_0y_0) + dist_3 \times 2x_0y_0$$

  - Apply camera matrix:

$$2DKeypoints = \begin{bmatrix} camera_{0,0} \times x + camera_{0,2} \\ camera_{1,1} \times y + camera_{1,2} \end{bmatrix}$$

- **Calculation of the bounding box:** In the next step, after calculating the 2D key points, we try to calculate four points to be the image's bounding box. These four points are the combination of $X_{max}, X_{min}, Y_{max}, andY_{min}$. Therefore, the four points are (X_{max}, Y_{min}), (X_{max}, Y_{max}), (X_{min}, Y_{max}), and (X_{min}, Y_{max}). This bounding box is used to calculate IoU loss in the model.
- **Mask resizing:** Masks are the labels for the segmentation heads, and they will be resized from $1920 \times 1200_{px}$ to

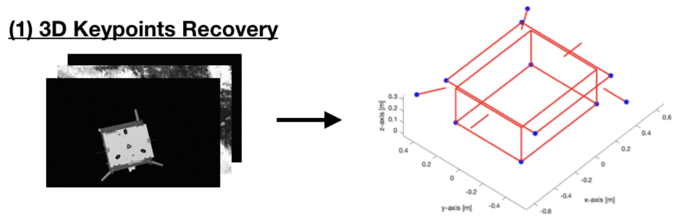

Fig. 6. Sample projection of the key points and object bounding box.

$192 \times 128_{px}$. Moreover, by downsizing the masks, we reduce the amount of computation for our segmentation head.

A. 3D Key points recovery

As mentioned, to create the bounding box and correctly identify the object location, we need 3D keypoints, which are 11 points in 3D space that Identify four top corners, four bottom corners, and the top of three antennas. Since these key points weren't available, we needed to regenerate them to use them in preprocessing of data. Moreover, these key points will be used to identify their exact match of points in 2D coordinates in the image, which are the main labels for the model bounding box, and to calculate the loss for the heatmap head by computing Perspective-n-Point (PnP) [3] problem.

Basely on the [4], we need to use the following minimization problem to restore the 3D:

$$minimize \sum_j ||s_j P^h_{2D,k} - K[R_j|t_j]P^h_{3D,k}||_2$$

To solve the above problem, we need at least 12 images in different positions and pick the 2D coordination of visible 11 key points in each image. In the next step, we will try to minimize the above equation, in which s_j is an unknown variable of scaling for image j, P2D is the 2D visible coordinates of image j, K is the camera matrix, R_j is the relative attitude, and t is the position of spacecraft.
However, unfortunately, we didn't achieve the 3D key points using this equation. Therefore, we decided to search for any available key points. As a result, we found the implementation of the [5], which provides us with key points, and transformed the labeled dataset, which cleared each of the 11 key points. But, after plotting the 2D points using the [3] provided key points, we could see the object's shape but on a bigger scale than the image.
Therefore, we used the provided labels of the dataset in the [5], picked up 100 images randomly, and again used the above equation in the following format:

$$minimize \sum_j ||P^h_{2D,k} - K[R_j|t_j]sP^h_{3D,k}||_2$$

In this case, the only variable is S, a scaling parameter for the 3D key points used in the [5]. The TABLE I is the final result of the 11 3D key points.

X	Y	Z
0.357983824	-0.37268308	0.312240995
0.356490163	0.371753994	0.312240995
-0.357409781	0.370936601	0.312194781
-0.356632852	-0.371349322	0.312294545
0.356775999	-0.254264043	0.000142638929
0.357198063	0.293192891	0.00168673247
-0.355823763	0.294407287	0.00169531064
-0.355063428	-0.2506908064	0.0029392834
0.296202464	-0.562132172	0.243598807
0.525819083	0.473696887	0.246265841
-0.528115513	0.47233981	0.246027345

TABLE I
11 3D KEY POINTS VALUES

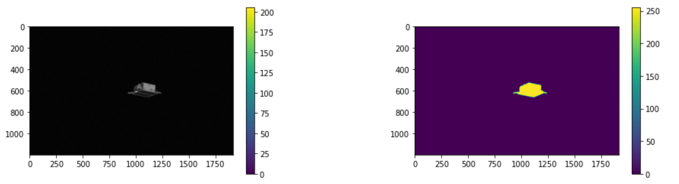

Fig. 7. A grabcut algorithm sample result.

B. Mask making

Unfortunately, while segmentation is one of the leading heads in our implementation, there has been no mask provided by speed datasets as the label for segmentation. Therefore, we tried to produce a mask for the segmentation using some of the classic image segmentation algorithms to provide an autonomous system for mask making over the Speed dataset training data.
To do so, we followed the following steps as our main algorithm:

1 Using grab cut
2 Using Mean Thresholding
3 logical OR of the two previous results
4 Using keypoint cropping, the spacecraft
5 Denoising
6 Using flood fill

1) Grab cut: Grab cut [6] is an algorithm that separates the background from the foreground in an image. In our case, the foreground is the spacecraft; using this algorithm, we make a mask that considers the spacecraft the foreground and is part of the mask.

2) Mean Thresholding: Unfortunately, grabcut didn't always retrieve good results because of the dark areas over the surface of the spacecraft; it wasn't able to separate the spacecraft from the dark background.

Therefore, we also decided to use the Mean Thresholding method [7] to find the object in the image. This method returns a threshold value based on the mean of grayscale values. Using this threshold, we are able to whiten the pixels over the threshold and darken the area below the threshold. With this method, we were able to recover more area of the object in the images. Moreover, the Mean method returned brighter results in comparison to other methods after several tests.

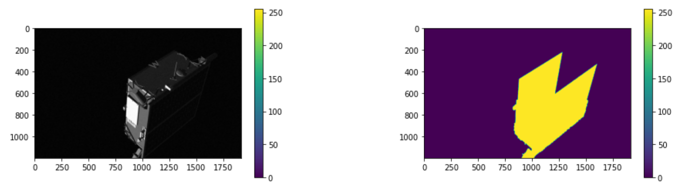

Fig. 8. A grabcut algorithm lack of accuracy in a sample result.

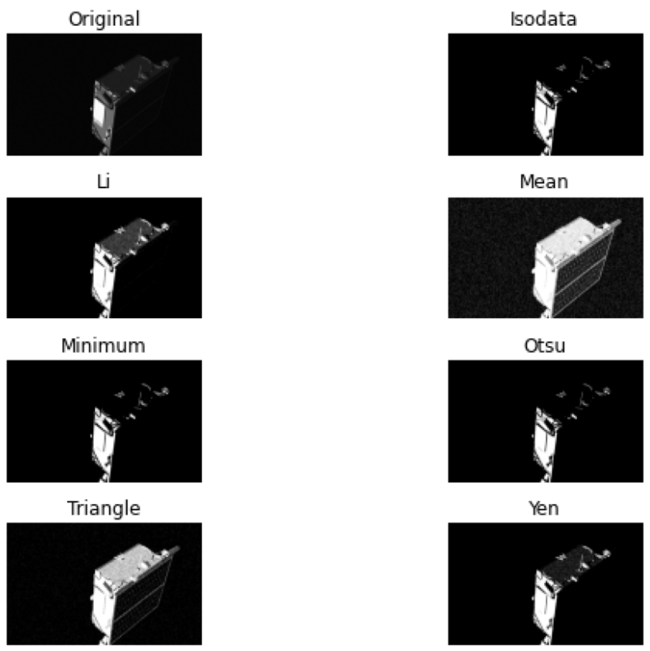

Fig. 9. Different thresholding methods.

3) *Logical OR:* To get the best results for both images, we decided to use logical OR over the two previous images. In this case, any part of an object which is not covered by one of the methods will be covered by the other.

4) *Keypoint Cropping:* Due to the noisy nature of pictures and mean thresholding, we decided to use the key points to find the objects bounding boxes. As a result, we crop the object from the mask and remove all the left pixels as the noise.

5) *Denoising:* While we removed a lot of border-box noises, there are still some noises in the bounding box. In order to remove all the left noise, we use two layers of fastdenoising OpenCV function algorithm.

6) *Flood Fill:* Although most of the spacecraft will be detected by the previous methods, in some cases, there are some holes in the object, which are the areas that the previous algorithms have not detected. Therefore to fill these holes, we use the Flood Fill algorithm that fills any enclosed space in the picture.

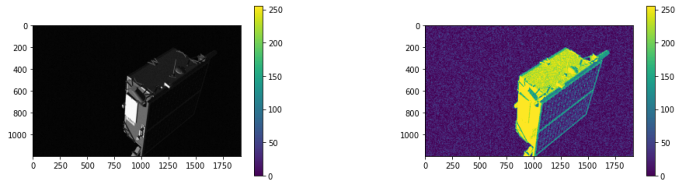

Fig. 10. Mean Threshold sample result.

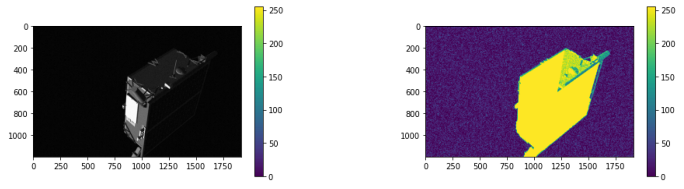

Fig. 11. Logical Or sample result of Mean Thresholding and Grab Cut.

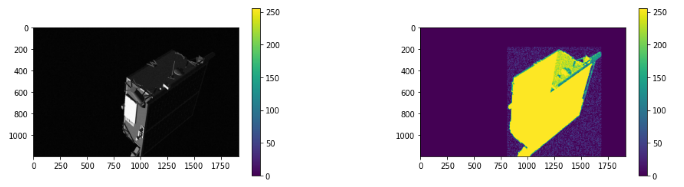

Fig. 12. Cropping the surrounding the image for noise decrease sample result.

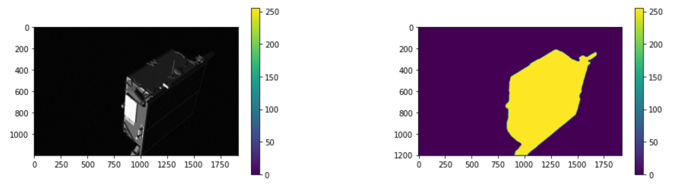

Fig. 13. Denoising the image sample.

IV. SPNv2 Offline Main Structure

The offline part of SPNv2 is a form of implementation of the Efficentpose architecture [8] specified for 6D pose estimation in spacecraft. Moreover, Efficentpose is an implementation of EfficentDet architecture for general 6D pose estimation of objects to keep the computational overhead relatively small in an intuitive way by adding to classification and bounding box subnet two new subnets to predict rotation and translation, respectively. Here, we first investigate the implementation and architecture of EfficentDet and the new heads implemented in SPNv2 for spacecraft pose estimation besides the unique subset of Efficentpose.

A. *EfficentDet*

This architecture uses used EfficientNet backbone, which is used to keep the efficiency of the model by scaling it in terms of width, depth, and resolution. Using the parameter ,

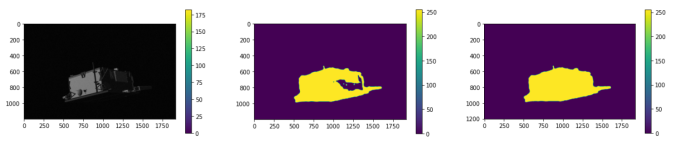

Fig. 14. Filling holes in the image sample.

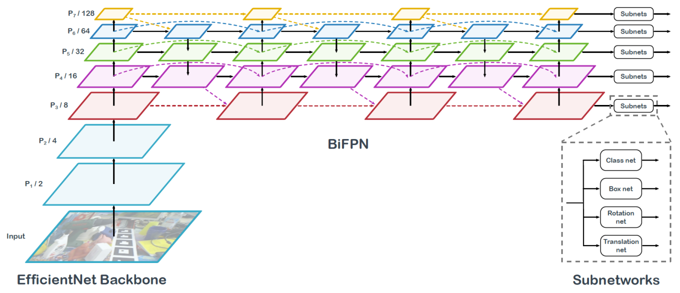

Fig. 15. Efficentpose Architecture.

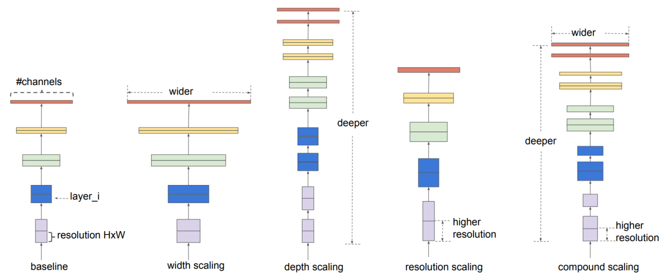

Fig. 16. Different type of scaling.

we manage the range of scaling in our model.

Using several layers of BiFPN repeatedly, we forward the extracted feature from these layers to several subnets (heads) to do different tasks, such as classification and segmentation, simultaneously to make a multi-task framework.

Moreover, Efficientdet uses the bidirectional feature pyramid network (BiFPN) to transfer the feature information of different layers to each other and keeps a dynamic interaction between the High-level and Low-level features in the model in several layers.

B. *EfficentPose Heads*

Here since we have features from several layers of a pyramid network, we can define several heads to do the pose estimation task. Therefore, in [?], they added two new subnetworks parallel to existing subnetworks (the classification and bounding box regression). However, the new subnets predict the rotation R and translation t, respectively. Moreover, in SPNv2, we call this head pose estimation head, which is our main head. In addition, SPNv2 also added two more heads, one for segmentation and one for a heatmap, and uses the loss of these two heads to increase the accuracy of the shared extracted features.

V. Experiments

A. *Implementation*

In SPNv2 implementation, the implementation uses the pre-implemented efficient pose. The below loss function was used:

$$E_{pose} = E_R(R^{\sim}, R) + E_T(t^{\sim}, t)/||t||,$$

$$E_R(R^{\sim}, R) = arcos\frac{tr(R^T, R^{\sim}) - 1}{2},$$
$$E_T(t^{\sim}, t) = ||t^{\sim} - t||$$

Moreover, The first part of implementation is preprocessing, which does all the preprocessing methods we discussed previously. After preprocessing, the train.py file will run the training process to train the model. The training process first loads a pre-trained model of One layer of bidirectional feature pyramid network efficeintdet. The model will be trained over eight epochs since, based on the pose loss of validation in training, the training is no more helpful after eight epochs. Moreover, other parameters, such as learning rate or ϕ, are based on the SPNv2 default used parameters.

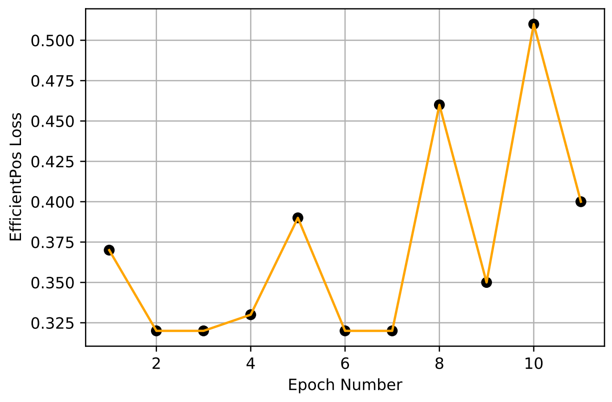

Fig. 17. Efficentpose Architecture.

B. *Baseline*

We used the suggested hyperparameter from the SPNv2 implementation to achieve the baseline. The only main difference is that we trained the model until three epochs more than the epoch with the minimum loss for the validation set.

In addition, since we are using different systems over google colab for each test, the random seed of every system is different, and because of that, the results for the same hyperparameter could be different in each test. Therefore, we performed each test five times and used the mean results as the final result for each set of hyperparameters.

Moreover, we define stopping criteria using the validation set to stop overfitting. In addition, if, after three epochs, we do not see any increase in the performance of the model over the validation set, we stop the training and save the best state of the model over the validation set.

Texture randomization, adjusting brightness, blur, solar flare, random erase, and noise are different augmentation methods. However, we used only four of them represented in the base paper (texture randomization, solar flare, and random erase). Moreover, the validation factor is the fraction of data that is going to be used as the validation over each training epoch, the LR is the learning rate of the optimizer, Loss

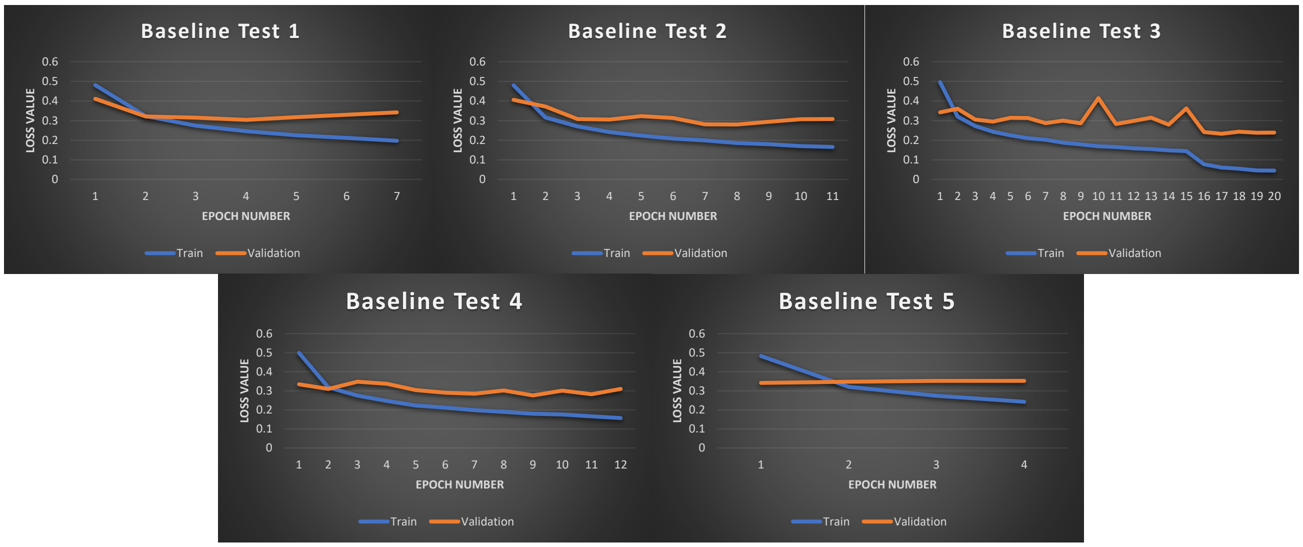

Fig. 18. Efficentpose Architecture.

Hyperparameters	Value
Input Size	[768, 512]
Epoch	20
Optimizer	AdamW
Learning Rate	1.0e-3
Loss Heads	[Heatmap,Efficientpose,Segmentation]
Loss Factors	[1.0, 1.0, 1.0]
Efficientdet PHI	3
GroupNorm size	16

TABLE II
BASIC CONFIGURATIONS

Method	IoU	$E_{r(degree)}$	$E_{t(meter)}$	E_{pose}
SPNv2 - H	-	6.304	2.622	0.271
SPNv2 - P	0.811	14.356	0.552	0.312

TABLE III
THE TABLE CONTAINS THE LOSS OF THE SPNV2 HEATMAP HEAD (SPNV2 - H) AND THE SPNV2 EFFICIENTPOS HEAD (SPNV2 - P). WE ALSO HAVE THE SEGMENTATION HEAD, WHICH HELPS US DURING THE TRAINING. THE RESULTS ARE THE MEAN OF FIVE TESTS.

Heads are the heads that had been trained, and the PHI is the scaling parameters, and the type of normalization is the Group Normalization with the size of 16.
Our data split for training is 70% train data, 20% validation, and 10% train data.
The table contains the Loss of the SPNv2 heatmap head (SPNv2 - H) and the SPNv2 efficientpos head (SPNv2 - P). We also have the segmentation head, which helps us during the training.

Method	IoU	$E_{r(degree)}$	$E_{t(meter)}$	E_{pose}
SPNv2 - H	-	0.584889	0.307	0.00991
SPNv2 - P	0.008325	1.771056	0.138359	0.040071

TABLE IV
THE TABLE CONTAINS THE LOSS OF THE SPNV2 HEATMAP HEAD (SPNV2 - H) AND THE SPNV2 EFFICIENTPOS HEAD (SPNV2 - P). WE ALSO HAVE THE SEGMENTATION HEAD, WHICH HELPS US DURING THE TRAINING.

Method	IoU	$E_{r(degree)}$	$E_{t(meter)}$	E_{pose}
SPNv2 - H	-	5.824	2.7634	0.2606
SPNv2 - P	0.7898	14.7328	0.5836	0.3038

TABLE V
TO COMPARE THE EFFECT OF THE SEGMENTATION HEAD, WE ALSO DID FIVE TESTS OF THE TRAINING AND TESTED THE MODEL WITHOUT THE SEGMENTATION HEAD.

Method	IoU	$E_{r(degree)}$	$E_{t(meter)}$	E_{pose}
SPNv2 - H	-	0.333467	0.00313	0.005941
SPNv2 - P	0.012558	0.464257	0.062107	0.011777

TABLE VI
THE TABLE REPRESENTS THE STANDARD DEVIATION FOR FIVE TESTS WITHOUT THE SEGMENTATION HEAD.

To study more about the baseline result, I tried to check if there is any relation between the position of the camera and the Loss of prediction. Moreover, after checking the Loss of each sample with its distance to the camera and plotting it over the below charts, it is possible that there is a correlation between the Loss and distance. Moreover, by increasing the distance of spacecraft from the camera, the average Loss also increases.

C. *Augmentation methods*

In the next step, since the number of data is limited, we decided to use different augmentation methods to study the possibility of improving the robustness of the model and solve the limited number of data problems. The methods of augmentation we used are the following:

- Style Augmentation: In this augmentation, the texture of the object will be replaced with a random new texture using the implementation of the [9], and during training, with the probability of 0.5, each image can be replaced by a style augmented image.
- Apply Solar Flare: Applying this method will produce a solar reflection color in the image.

Augmentation Type	Method	IoU	$E_{r(degree)}$	$E_{t(meter)}$	E_{pose}
Style Augmentation	SPNv2 - H	-	7.95	2.77	0.301
Style Augmentation	SPNv2 - P	0.79	16.72	0.67	0.344
Apply Solar Flare	SPNv2 - H	-	6.52	2.76	0.273
Apply Solar Flare	SPNv2 - P	0.81	15.44	0.64	0.321
Apply Random Erase	SPNv2 - H	-	6.65	2.76	0.276
Apply Random Erase	SPNv2 - P	0.79	17.16	0.75	0.365

TABLE VII
TABLE CONTAINS THE MEAN RESULTS OF FIVE TESTS FOR DIFFERENT AUGMENTATION METHODS.

Augmentation Type	Method	IoU	$E_{r(degree)}$	$E_{t(meter)}$	E_{pose}
Style Augmentation	SPNv2 - H	-	1.5019	0.0140	0.0285
Style Augmentation	SPNv2 - P	0.01	2.0401	0.102025	0.0395
Apply Solar Flare	SPNv2 - H	-	0.3270	0.0035	0.0062
Apply Solar Flare	SPNv2 - P	0.0080	0.5054	0.036011	0.0055
Apply Random Erase	SPNv2 - H	-	0.8840	0.0123	0.0182
Apply Random Erase	SPNv2 - P	0.0266	1.6943	0.189919	0.0461

TABLE VIII
TABLE CONTAINS THE MEAN RESULTS OF FIVE TESTS FOR DIFFERENT AUGMENTATION METHODS.

- Apply Random Erase: This method will remove a part of the image.

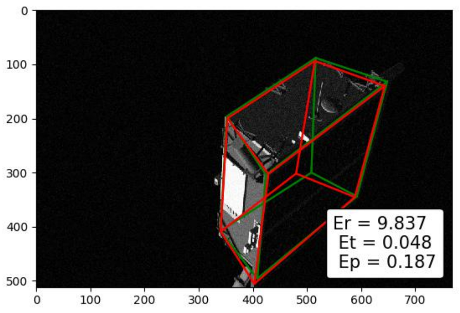

Fig. 19. Sample prediction result, the red lines define the ground truth, and the green one defines the prediction.

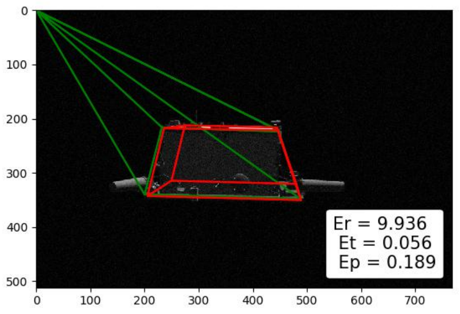

Fig. 20. Sample prediction result, the red lines define the ground truth, and the green one defines the prediction.

D. Statistical Tests

To better understand and compare methods,there are more statistical Tests based on the student table between baseline and other methods. This procedure calculates the difference between the observed means in two independent samples. A significance value (P-value) and 95% Confidence Interval (CI) of the difference are reported. The P-value is the probability of obtaining the observed difference between the samples if the null hypothesis were true. The null hypothesis is the hypothesis that the difference is 0.

Difference	0.047
Standard error	0.025
95% CI	-0.0115 to 0.1047
t-statistic	1.850
DF	8
Significance level	P = 0.1015

TABLE IX
STATISTICAL TESTS BASE LINE & STYLE AUGMENTATION

Difference	0.024
Standard error	0.018
95% CI	-0.0179 to 0.06555
t-statistic	1.315
DF	8
Significance level	P = 0.2248

TABLE X
STATISTICAL TESTS BASE LINE & APPLY SOLAR FLARE

VI. CONCLUSION

By analyzing the results, we can see that in the two main heads, the best results for the rotation error belong to the Heatmap head, and the best results for position error belong to the Pose Estimation head.
Moreover, by comparing the results of standard deviation results, we can see the total pose loss has a wide range. We believe this could be the results of the in-accurate mask head since the primary baseline consists of three heads (Heatmap, Pose Estimation, and Segmentation), and to calculate the mask labels due to their unavailability, we used some classical method to calculate the mask label. In addition, this result becomes much bolder when we remove the segmentation head

Difference	0.067
Standard error	0.027
95% CI	-0.0041 to 0.1303
t-statistic	2.457
DF	8
Significance level	P = 0.0395

TABLE XI
STATISTICAL TESTS BASE LINE & APPLY RANDOM ERASE

Difference	0.006
Standard error	0.019
95% CI	-0.0371 to 0.0491
t-statistic	0.321
DF	8
Significance level	P = 0.7563

TABLE XII
STATISTICAL TESTS BASE LINE & BASE LINE WITHOUT SEGMENTATION

from the model, and we can see the standard deviation of the new test reduce drastically.

Furthermore, we believe our results could improve more if we can increase the data amount. we decreased the training data due to several reasons:

- First, we switched from the SPEED Plus dataset to the SPEED dataset. While the SPEED plus has more than 47000 training data, the SPEED only has 12000 training data. The main reason for this was the lack of processing power since, in the paper, they used four distributed GPUs. We could only use the Colab pro, which also has limited hours of GPU time, which restricts our training time.
- The second point is that the test data in the dataset do not have labels. Therefore, we needed to split the data to test and train to build, train and test the model.
- The third point is that since the complexity of calculating mask for the images with an earth background was high, we only used images without an earth background, which reduced the number of our data to 6000.

The other point that is needed to understand is the difference between the mean and p_value results for different methods. For the baseline with and without segmentation, it is clear why we have better pose loss due to the previous results, so we only going to talk about the augmentation methods and their effects.

Based on the paper, the style augmentation of the results improved, but for us, we didn't see any improvement in our results. However, after the T-Test, we saw that the P-Value was about 0.10, which means our result difference means could converge to zero in the more amount of test or data. This also can be concluded for the Solar Flare augmentation with the P_Value of 0.22. However, In the case of the Random Erase, while it didn't improve the results both in the paper and we could see that there is no improvement, but the P_Value is enormously low, about 0.03, which means in a high amount of the tests this result would not give us the same results with a probability of 0.97. This range of difference in results still can be because of the above results for lack of data and its in-accurate masks or the complexity of the model.
Another important point needed to notice is the key points prediction of the image test data. In some cases, we can see a point either not correctly predicted or not predicted. These two cases can also result in to decrease in loss and error. However, one way to decrease overall loss and face the above problem is to find more than 11 key points of an object, which increase the accuracy of solving PnP problem.

References

[1] T. H. Park and S. D'Amico, "Robust multi-task learning and online refinement for spacecraft pose estimation across domain gap," *arXiv preprint arXiv:2203.04275*, 2022.

[2] S. Sharma, T. H. Park, and S. D'Amico, "Spacecraft pose estimation dataset (speed)," in *Stanford Digit. Repository*, 2019.

[3] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *International journal of computer vision*, vol. 81, no. 2, pp. 155–166, 2009.

[4] T. H. Park, S. Sharma, and S. D'Amico, "Towards robust learning-based pose estimation of noncooperative spacecraft," *arXiv preprint arXiv:1909.00392*, 2019.

[5] B. Chen, J. Cao, A. Parra, and T.-J. Chin, "Satellite pose estimation with deep landmark regression and nonlinear pose refinement," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pp. 0–0, 2019.

[6] C. Rother, V. Kolmogorov, and A. Blake, "" grabcut" interactive foreground extraction using iterated graph cuts," *ACM transactions on graphics (TOG)*, vol. 23, no. 3, pp. 309–314, 2004.

[7] C. A. Glasbey, "An analysis of histogram-based thresholding algorithms," *CVGIP: Graphical models and image processing*, vol. 55, no. 6, pp. 532–537, 1993.

[8] Y. Bukschat and M. Vetter, "Efficientpose: An efficient, accurate and scalable end-to-end 6d multi object pose estimation approach," *arXiv preprint arXiv:2011.04307*, 2020.

[9] P. T. Jackson, A. A. Abarghouei, S. Bonner, T. P. Breckon, and B. Obara, "Style augmentation: data augmentation via style randomization.," in *CVPR workshops*, vol. 6, pp. 10–11, 2019.