2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Satellite-Model-Free Deep Learning based Pose Estimation of Non-cooperative Satellite and Tracking using Navigation Filter

Shubham Shukla*,1, Raunak Srivastava[2], Rolif Lima*,3 and Titas Bera*,4

***Abstract*— One core component of Active Debris Removal (ADR) and On-Orbit Servicing (OOS) missions in space is to estimate and track the relative pose of a non-cooperative satellite in close proximity. Conventionally, Image Processing methods have been popular in pose estimation by employing manual feature extraction techniques. But the performance of such methods plateaus in the challenging illumination conditions and sensor capability constraints in space, because of which Deep Learning (DL)-based approaches have gained traction. This paper aims to provide an improvement over the existing state-of-the-art direct pose estimation methods from a monocular camera, without relying on any 3D model of the target satellite. The main contribution of this work is to develop a general purpose satellite-invariant pose estimation architecture with improved accuracy and implement an adaptive navigation filter over it to track the pose continuously over a stream of images. The pose estimation module includes a modified DenseNet architecture. In order to test the generalization capability, the proposed pose estimation module is tested on the SPEED, SPEED+, SHIRT and URSO datasets and compared with other existing methods. The advantage of the proposed method is that the same model architecture is able to give accurate pose estimation results for different satellite datasets. To perform continuous tracking of the relative pose, an adaptive EKF (Extended Kalman Filter) is implemented on the initial pose estimates. For performance evaluation of the navigation filter, the accuracy goals required for the relative navigation of Hubble Space Telescope SM4 mission are considered while testing on the SHIRT dataset.**

I. INTRODUCTION

With the growing interest and research progress in satellite robotics operations like Active Debris Removal (ADR) and On-Orbit Servicing (OOS), there has been a commensurate increase in the need for solving the problem of relative pose estimation of a non-cooperative satellite. To perform autonomous rendevous and docking of a chaser satellite to a target satellite, the relative state of the target is important to be determined and tracked to a fair and acceptable amount of accuracy. This can be achieved using sensor inputs from either of monocular camera, stereo or depth camera or Lidar or a combination of these. However, solving this problem entails a resource constraint on the amount of computational power and energy requirements of the chaser satellite. For this reason, the use of a monocular camera is the preferred choice among other sensors.

However, pose estimation using only a monocular camera in space environment comes with its own challenges. The biggest among them being extreme illumination conditions of satellite imagery. This, added to the lack of atmosphere and light diffusion and situations like being in Earth shadow region or in direct sunlight can result in severely dim images or images with high glare from the satellites' solar panels. Besides, the background presence of the Earth poses a challenge in distinguising the satellite, especially at large distances. The low resolution of imagery due to hardware constraints also poses a limitation in getting an accurate estimate of the relative pose.

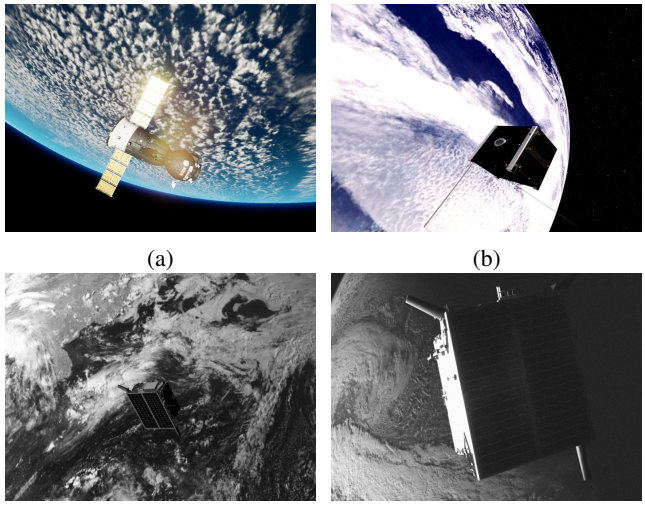


Fig. 1: Images from publicly available datasets. (a) Soyuz aircraft from the URSO dataset. (b) cubesat from the SwissCube dataset. (c) and (d) synthetic and real lab images respectively from the SPEED+ dataset.

Conventionally, image processing approaches have been used for pose estimation. These involve using methods like SIFT, SURF, etc. to extract handcrafted features such as corners, edges of the satellite in question. These features are then matched with the satellite model to evaluate the relative pose. However, in light of above challenges, these methods fail to capture all such required features causing the pose estimation to fail. Also, an image possesses features more than just in the forms of lines, edges or corners. This makes the use of Deep Learning (DL) based methods a suitable alternative to the image processing based methods. Using DL, the input images can be used to train a model and give more efficiency in capturing the various underlying features in the image. It also increases the robustness of the

*Authors are with Tata Consultacy Services - Research
[1]shukla.shubh@tcs.com
[2]raunaksrivastava1410@gmail.com
[3]rolif.lima@tcs.com
[4]titas.bera@tcs.com

accuracy in harsh illumination conditions which can cause loss of observable features. The usage of DL based methods in terrestrial applications and the corresponding maturing of research in the field has paved way for employing DL based methods for satellite pose estimation in space environment.

However, a challenge that satellite pose estimation presents in the usage of DL based methods is the availability of publicly available datasets for training and evaluating the DL models. The Satellite Pose Estimation Challenge (SPEC) - 2019 [1] first made available the SPEED dataset [2] using a model of a Tango satellite as the target. There have been a few other datasets subsequently, such as URSO [3] and SwissCube [4]. However, these consist of only synthetically rendered images and do not reflect the actual space environment. Datasets such the SPEED+ [5], SHIRT [6] and SPARK dataset [7] do provide lab generated real images, but they do not generalise to a wide range of satellite models. Fig. 1 shows the sapcecraft images from some of these datasets. For this paper, the URSO, SPEED, SPEED+ and SHIRT datasets have been used.

The main contribution of this paper is two fold:
1) to determine the initial pose estimate using a DL based method without using any 3D model of the satellite
2) to remove the outliers in the pose estimate and continuously track the pose using an Extended Kalman Filter (EKF) based navigation filter

Majority of the works currently in literature depend on the target satellite model for estimating the pose. However, the proposed method deos not rely on any model of the satellite, thus making it more generalisable. Moreover, to the best of the authors' knowledge, there is no other literature using direct pose estimation (i.e. without using target satellite 3D model) integrated with a navigation filter to continuously track the relative pose.

II. Literature Overview

The problem of determining the relative pose of a satellite using Deep Learning (DL) from monocular images has been solved using two approaches: Direct method and Indirect method. For this paper, a direct method is defined as one which does not require any 3D model of the satellite whose pose needs to be determined. Such a DL framework, on receiving the monocular image as input, provides the relative pose as the output without involving any intermediate steps. An indirect method, on the other hand, uses the DL framework as an image processing intermediate module to extract handcrafted features of the satellite (using its 3D model) from the input image. Subsequently, these features are used in 2D-3D pose correspondence methods like Perspective-n-Point (PnP) to get the pose output. Subsequently, an iterative pose refinement method is required to achieve better accuracy of the relative pose. The literature survey papers [8], [9] well document these available methods.

Among the direct pose estimation methods, Spacecraft Pose Network (SPN) in [10] & [11] is among the very first works using CNN based satellite pose estimation. It uses a 5 layer CNN based feature extractor and three prediction branches. The attitude branch classifies the orientation, which is then used with the 2D bounding box and geometrical constraints to determine the relative position. The model is trained and tested on the SPEED dataset. URSONet proposed in [3] uses a ResNet based backbone to determine position through regression and orientation through probabilistic soft classification. [12] uses a two stage network, firstly a YOLOv5 for satellite detection and secondly a modified HRNet architecture for estimating pose directly. Other works in [13], [14] and [15] either do not calculate the full pose (determining either position or velocity) or do not give good accuracy. Direct approaches offer the benefit of being simple in design and execution, besides doing away with any dependence on the availability of a 3D model of the satellite. They also do not require the computationally heavy iterative pose estimation and refinement methods used in indirect methods.

The notable works among the indirect pose estimation methods are discussed here. The authors in [16] propose a HRNet backbone based Faster-RCNN to extract predetermined keypoints of the target satellite. Pose is then computed using PnP + RANSAC, and refined using SA-LMPE optimiser. [4] proposes a Feature Pyramid Network (with DarkNet as backbone) to regress 2D landmarks at various scales and use a RANSAC + PnP strategy to fuse the pose estimates. However, these methods evaluate the estimated pose only on the synthetic dataset. There are other works which implement the pose estimation on real lab images. SPNv2 in [17] consists of a shared multi-scale feature encoder and multiple prediction heads that perform different tasks of keypoint prediction, pose estimation and segmentation masks on a shared feature output. It also proposes an Online Domain Refinement (ODR) to bridge the domain gap while testing on real images. [18] suggests a transformer based HRNet architecture to simultaneously predict heatmaps and segmentation. [19] proposes a two stack hourglass network with multi stage loss computation to detect keypoint heatmaps and depth estimate. However, both previous methods use pseudo-label generation for domain adaptation, which may not be suitable computationally given the satellite hardware constraints.

The challenging imagery in space can cause the pose estimate to give erroneous results, which makes having a

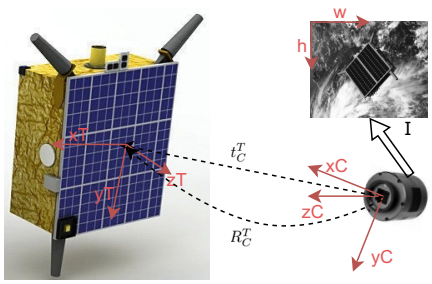


Fig. 2: Referance frames, relative position, relative orientation and camera image as used in given problem statement.

navigation filter imperative in real life applications. There is very few literature on integrating a navigation filter with DL based pose estimation. [20] combine a Convolutional Neural Network (CNN) for feature detection with a Covariant Efficient Procrustes Perspective-n-Points (CEPPnP) solver and an Extended Kalman Filter (EKF) to enable robust monocular pose estimation. [21] adds an Adaptive Unscented Kalman Filter to SPNv2 [17] architecture. However, these require a 3D model to estimate the pre-determined keypoints.

III. Problem Formulation

The problem of relative pose estimation of an uncooperative satellite consists of determining the relative transformation, i.e. position and orientation of the target satellite with respect to the chaser satellite. As mentioned previously, a monocular camera mounted on the chaser satellite is the suitable and preferred sensor choice due to power and computation constraints in space. In fig. 2, $F_C\ (x_C, y_C, z_C)$ represents the camera center frame mounted on the chaser satellite, whereas $F_T\ (x_T, y_T, z_T)$ represents the body center frame of the target satellite. Hence, the relative position is defined as the translation (t_C^T) of the target body center frame with respect to the camera center frame. Similarly, the relative orientation is defined as the rotation (R_C^T) of the target body center frame with respect to the camera center frame. t_C^T and R_C^T respectively are represented as follows:

$$t_C^T = (x_C^T, y_C^T, z_C^T) \epsilon R^3 \tag{1}$$

$$R_C^T = (q_0, q_1, q_2, q_3) \epsilon R^4 \tag{2}$$

where (q_0, q_1, q_2, q_3) is the quaternion q.

The general problem involved here requires determining t_C^T and R_C^T using monocular or RGB images from the chaser camera. The images I are of the form (h, w, c), each respectively representing the height, width and number of channels of the image. c can be 1 in case of grayscale images or 3 in case of colored images.

The method proposed in this paper involves using the image I to extract features specific to the target satellite and evaluate the relative pose as shown in Fig. 2. Since the proposed approach is a direct method, both the feature extraction/image processing and pose estimation is done using a single, end-to-end Deep Learning model.

IV. Pose Estimation

The proposed Deep Learning based pose estimation approach consists of a modified DenseNet [22] model as the backbone which acts as the encoder. DenseNets offer the benefit of having improved flow of information and gradients, since each layer has direct access to the gradients from the loss function and the original input signal. This leads to implicit deep supervision and capturing global features. The dense connections have a regularizing effect, which reduces overfitting and leads to more accurate models. Also, DenseNets are highly parameter-efficient and require less computations to achieve state-of-the-art performance.

A. DenseNet

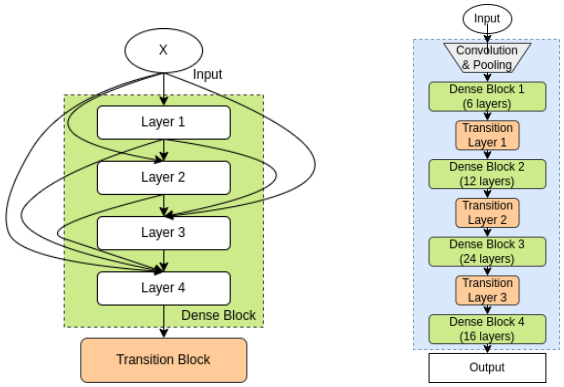
[23] provides a detailed information on the DenseNet architecture and how the features propagate in the network. DenseNets concatenate outputs from all previous layers, enabling reuse and direct supervision of features. DenseNet consists of cascaded dense blocks, shown in fig. 3a. For this paper, the variant Densenet-121 has been considered. It consists of 4 DenseBlocks, each having different number of convolutional layers. An overview of this variant, showing the number of layers in each DenseBlock is given in fig. 3b.

B. Proposed Architecture

This section details the direct, end-to-end, architecture (fig. 4) used to estimate the satellite's relative pose. The first part of the proposed architecture consists of the feature extraction layer. For this purpose, the final classifier layer of DenseNet-121, giving output of the shape $(1000, 1)$, is discarded. The pre-final layer of original DenseNet-121 consists of Batch Normalization (BN), which receives the input from DenseBlock-4, and gives output of the shape $(15, 24, 1024)$. In the proposed architecture, this BN layer is appended with an Average Pooling layer, which acts as the image feature bottleneck. Thus, instead of having the output of the shape $(1000, 1)$, the modified DenseNet-121 gives the output of the shape $(n, n, 1024)$. This helps in passing on a richer feature map to the subsequent layers of the proposed network. Here, n can has been varied from 2 to 3.

The second part of the proposed network consists of two Fully Connected Layers (FCLs) parallel branches. Each branch receives the linearized input from the shared feature encoder of part one. The third part consists of a prediction head for each position and orientation branch. The position is estimated through direct regression. The position loss function L_{pos} used is the Euclidean distance between the ground truth translation vector and the predicted translation vector.

$$L_{pos} = \sum_{i}^{3} ||t_{gt}^i - t_{pr}^i||_2 \tag{3}$$



(a) DenseBlock (b) DenseNet-121

Fig. 3: (a) Flow of information and connections within a DenseBlock. (b) Overview of the DenseNet-121 architecture.

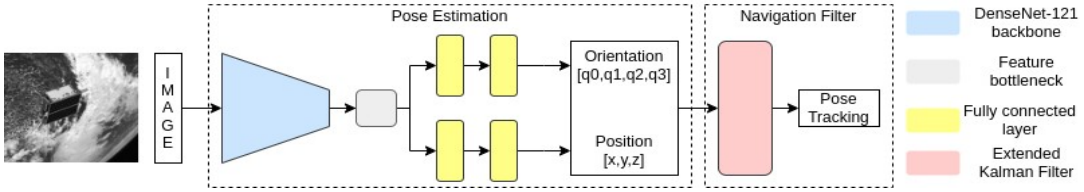


Fig. 4: Design of the proposed pose estimation architecture and navigation filter.

where t_{gt} and t_{pr} are the ground truth and predicted translation vectors respectively. Departing from the usual practise of using normalized Euclidean distance in other literature, here the absolute Euclidean distance is preferred because it gives better results when testing on images having varied range of the target satellite from the camera. The orientation estimation is done using soft classification method, by discretizing the output angular space as in [3]. Each ground truth orientation label is encoded as a Gaussian random variable, so that the network learns to output probability mass functions. The loss function L_{ori} used for the orientation estimate is the standard negative log-likelihood between the ground truth quaternion q_{gt} and predicted quaternion q_{pr}. The total loss function L_{total} used to train the model is:

$$L_{total} = L_{pos} + L_{ori} \tag{4}$$

The idea behind having separate FCL branches for position and orientation estimate is to have different learning characteristics since each is estimated using a different method.

V. Navigation Filter

The pose of the target obtained from the deep learning based methodology needs to be passed through a navigation filter for two primary reasons: One is to reduce the noise in the obtained pose before it can be used by a control algorithm to track the object. Secondly, to ensure that we have a prediction of the target's pose for a few time frames even if we do not obtain measurements due to certain target maneuvers, etc. Here an Extended Kalman Filter is used with the following details.

Since the motion of the target is unknown, we use a constant velocity model to propagate the pose of the target in an EKF. The states and output measurements for the Kalman filter are taken as

$$\mathbf{x} = [\mathbf{p}, \mathbf{v}, \mathbf{q}, \omega]^T \tag{5}$$

$$\mathbf{z} = [\mathbf{p}, \mathbf{q}]^T \tag{6}$$

where, $\mathbf{p} = [x, y, z]^T$ and $\mathbf{v} = [\dot{x}, \dot{y}, \dot{z}]^T$ represents the position and velocity vector of the target in the chaser's body frame. $\mathbf{q} = [q_x, q_y, q_z, q_w]^T$ and $\omega = [\omega_x, \omega_y, \omega_z]^T$ represent the orientation and angular velocity of the target with respect to the chaser.

The position and velocity can be propagated in the discrete form using a constant velocity model:

$$\begin{aligned} \dot{\mathbf{p}} &= \mathbf{v} + \mathbf{w}_p \\ \dot{\mathbf{v}} &= \mathbf{w}_v \end{aligned} \tag{7}$$

where $\mathbf{w}_p$ and $\mathbf{w}_v$ are the corresponding process noise. This model is used in a standard EKF to propagate the states and covariance matrix, compute Kalman gain, and finally update the states using the position measurements.

Similarly, the quaternions and angular velocity (total 7 DOF) can be propagated using a constant velocity model:

$$\begin{aligned} \dot{\mathbf{q}} &= \frac{1}{2}\begin{bmatrix}\omega \\ 0\end{bmatrix} \otimes \mathbf{q} \\ \dot{\omega} &= J^{-1}(-\omega \times J\omega) \end{aligned} \tag{8}$$

However, using an EKF with the entire quaternion (4 DOF) leads to singularity issues in the concerned covariance matrix as the system propagates. For this reason, we use an error state-based EKF for the estimation of quaternion and angular velocity of the target. The states comprise the error in quaternion and angular velocity (6 DOF). The error state can be expressed as :

$$\tilde{\mathbf{x}} = \begin{bmatrix}\delta q \\ \Delta\omega\end{bmatrix} \tag{9}$$

The error state dynamics using a constant velocity model can be expressed as:

$$\begin{aligned} \delta\dot{q} &= -[\omega\times]\delta q + \frac{1}{2}\Delta\omega + \mathbf{w}_q \\ \Delta\dot{\omega} &= J^{-1}[(J\omega)^{\times} - \omega \times J]\Delta\omega + \mathbf{w}_\omega \end{aligned} \tag{10}$$

where $\mathbf{w}_q$ and $\mathbf{w}_\omega$ are the corresponding process noise. The above equations are used in the discrete form to compute the state transition matrix which is used to propagate the covariance matrix.

For propagation, the entire 7 DOF state is used as mentioned above in Eq. 8. The error states (6 DOF) are reset to be 0 at the start of every EKF iteration. Thereafter, the state transition matrix is obtained through the error dynamics in Eq. 10. This state transition matrix is used for error covariance propagation. Thereafter, the corresponding Kalman gain is found which is then used to compute the updated error state using the measured quaternion (z_q).

$$\delta z = z_q \otimes \hat{q}^{-1} \tag{11}$$

It is expected that the difference (or amount of rotation) from the current estimate of quaternion to the measured quaternion is small. Hence, the scalar part (q_w) for δz tends to one. In such cases, the innovation term and hence the updated error state can be computed as

$$\hat{\tilde{\mathbf{x}}} = \begin{bmatrix}\delta\hat{q} \\ \Delta\hat{\omega}\end{bmatrix} = K \begin{bmatrix}\delta z_x \\ \delta z_y \\ \delta z_z\end{bmatrix} \tag{12}$$

Once the updated error state is obtained, the actual states can be updated as:

$$\delta\hat{q} = \begin{bmatrix} \delta\hat{q} \\ \sqrt{1 - \delta\hat{q}^T\delta\hat{q}} \end{bmatrix} \tag{13}$$

$$\hat{\mathbf{q}} = \delta\hat{q} \otimes \hat{\mathbf{q}} \tag{14}$$

$$\hat{\omega} = \Delta\hat{\omega} + \hat{\omega} \tag{15}$$

To improve the robustness of the navigation filter with respect to erroneous pose estimates due to extreme illumination conditions, the EKF model is made adaptive. In order to do so, the measurement noise covariance matrix (R) is made to vary depending on the intensity of the pixel values of the input image. Having a high value of R for such images makes the EKF rely less on the pose estimate. Due to extremely dim lightning or glares off the solar panels, the features may not be evidently visible. Hence, a certain minimum and maximum values of standard deviation (sd) of the pixel values is used as a threshold and R is replaced by R_{EKF} as follows

$$R_{EKF} = R_{coeff} * R \tag{16}$$

where,

$$R_{coeff} = \begin{cases} 100, & \text{if } sd < sd_{min} \text{ and } sd > sd_{max} \\ 1, & \text{otherwise} \end{cases} \tag{17}$$

VI. Experiment

All the experiments, i.e. training, pose estimation using the trained model and pose tracking using navigation filter are conducted on a system with AMD Ryzen 32-cores×64 processor and 256GB of RAM. The graphics processor used for training the DL model is 48GB NVIDIA RTX A6000. Further details of each experiment step are presented next.

A. Initial Pose Estimate

Taking forward from the proposed architecture design, the feature output size of the DenseNet-121 backbone encoder is $(n, n, 1024)$. Using the feature bottleneck of size 3×3 greatly increases the network parameters with only a marginal increase in the pose estimation accuracy. Hence, the proposed model is trained and tested with the feature bottleneck size of 2×2 (i.e. $n = 2$). This output feature map of shape $(2, 2, 1024)$ is linearized, giving the output of the encoder of shape $(4096, 1)$. The two Fully Connected Layers (FCLs) in each branch have 512 nodes each. The position prediction head has output of the the shape $(3, 1)$. For soft classification in the orientation prediction head, the discretization of output angular space is done taking number of bins in each dimension as 16. Thus, the orientation output is of the shape $(4096, 1)$. The input image is resized to $(768, 480)$. The training and evaluation of the pose estimation is done using Pytorch. This initial pose estimate is evaluated for the URSO, SPEED, SPEED+ and SHIRT datasets. The parameter count of this model is $\sim 13.25M$.

1) Training & Testing: For training, a batch size of 32 is used with Stochastic Gradient Descent (SGD) as the optimizer, having momentum of 0.9. Training is done running the model for 60 epochs. The initial learning rate chosen is 0.01 and a MultiStepLR scheduler (with γ=0.1) is used to decay the learning rate after 30^{th} and 45^{th} iterations. To minimize overfitting during training and making the model robust to varying illumination conditions, extensive data augmentations are performed on the training dataset. These include randomly adding noise and blur, randomly varying brightness and contrast, and randomly blacking out patches of the image. The model is trained separately as follows: for Soyuz aircraft using URSO training images; for Tango aircraft using SPEED+ synthetic training images.

While testing the model, the batch size used is 1, thus taking individual input images for estimating the pose. The evaluation for Soyuz is done using URSO test dataset. For Tango satellite, the testing is done on SPEED, SPEED+ and SHIRT datasets.

B. Pose Tracking using Navigation Filter

The navigation filter has been tested on the SHIRT dataset. With the navigation filter integrated, the initial pose estimate from the model is fed as measurement values to the EKF. For making the EKF adaptive, the following values are chosen for the filter parameters defined in section V.

$$R_{coeff} = 100,\ sd_{min} = 0.016598,\ sd_{max} = 0.14686$$

VII. Results

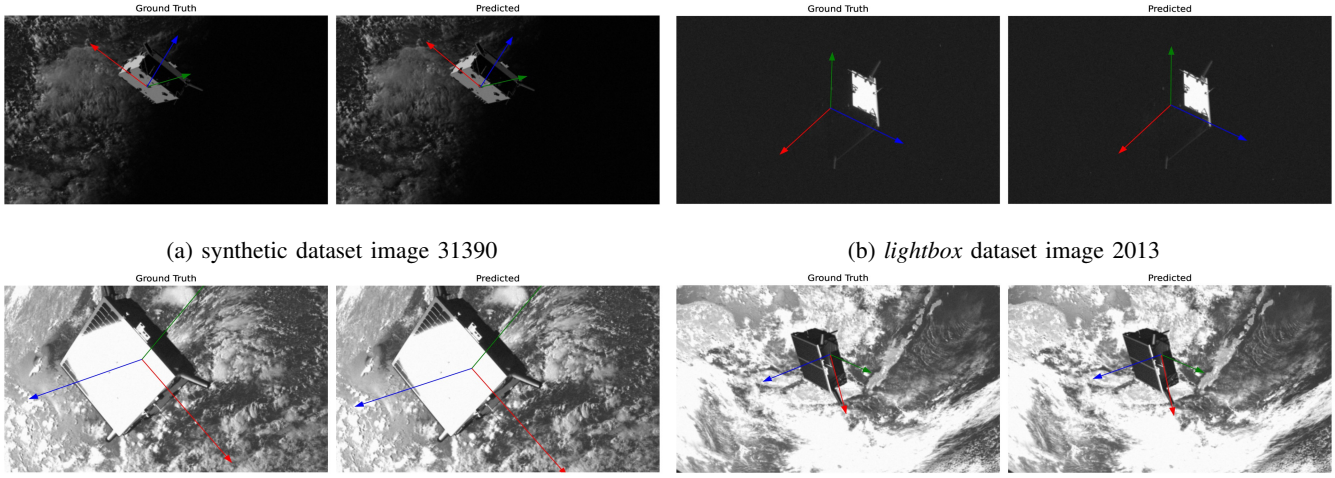
The performance of the estimated pose is compared with the accuracy requirements (Table II) specified in the NASA Hubble Space Telescope (HST) Servicing Mission-4 (SM4) [24].

A. Initial Pose Estimate

The performance of DL based pose estimation is evaluated for the *soyuz_easy* and *soyuz_hard* images of the URSO dataset, along with the synthetic and *lightbox* images of both the SPEED+ and SHIRT datasets. Since for SPEED dataset pose labels for the test images are not available, the evaluation is done on the 12,000 images of the training set. Table III shows that the pose estimate using the proposed method fares better or comparable to that of URSONet taken from [3], despite having significantly smaller parameter size.

To study the performance of the proposed method with other direct pose estimation methods, the SPEED dataset is considered as in the SPEED+ and SHIRT datasets, mostly indirect pose estimation methods have been used. Table IV lists the state-of-the-art model free pose estimation methods along with the architecture parameter size (in millions) and their pose accuracy results. The comparison clearly indicates the supremacy of the proposed architecture in terms of both accuracy and parameter size.

Fig. 5 displays the estimated position and orientation of few of the images from the SPEED+ dataset. The corresponding quantitative pose values are given in table I. The



(a) synthetic dataset image 31390

(b) *lightbox* dataset image 2013

(c) *lightbox* dataset image 4421

(d) *lightbox* dataset image 311

Fig. 5: Visualization of the estimated position and orientation results of images from the SPEED+ dataset.

TABLE I: Pose estimate values for Fig 5

Image	Ground Truth		Predicted	
	Position (x, y, z) (m)	Orientation (pitch, yaw, roll) (°)	Position (x, y, z) (m)	Orientation (pitch, yaw, roll) (°)
Fig. 5a	(-0.262, -0.342, 6.96)	(-142.55, 25.24, 63.49)	(-0.239, -0.338, 6.948)	(-139.57, 26.45, 62.35)
Fig. 5b	(-0.078, -0.022, 5.297)	(-179.13, 53.82, -45.89)	(-0.008, -0.054, 5.074)	(-179.22, 54.89, -45.05)
Fig. 5c	(-0.173, -0.152, 3.621)	(-39.48, -37.54, -159.77)	(-0.179, -0.079, 3.476)	(-40.02, -36.04, -162.83)
Fig. 5d	(-0.107, -0.357, 6.829)	(-66.08, -57.42, -134.0)	(-0.116, -0.353, 6.795)	(-71.59, -57.45, -140.13)

TABLE II: RNS Pose Accuracy Goals

Target Range (m)	Lateral Accuracy (m)	Range Accuracy (m)	Roll Accuracy	Yaw/Pitch Accuracy
150	1.0	1.0	15°	15°
30	0.3	0.5	5°	5°
5	0.1	0.1	1°	5°

TABLE III: URSO dataset initial pose estimate position and orientation errors

Image type	Parameter Size (in M)	*soyuz_easy*		*soyuz_hard*	
		Position Error (m)	Orientation Error (°)	Position Error (m)	Orientation Error (°)
URSONet	500	**0.56**	8	**1.1**	13
Proposed	**13.25**	0.69	**6.74**	1.32	**12.7**

TABLE IV: SPEED dataset initial pose estimate performance comparison with other model free direct pose estimation methods

Method/ Reference	Parameter Size (in M)	Position Error (m)	Orientation Error (°)
SPN [10]	-NA-	0.7832	8.4254
URSONet [3]	500	0.17	4.00
Mobile-URSONet [25]	**7.4**	0.56	6.29
LSPNet [26]	47.8	0.456	13.96
Proposed	13.25	**0.1085**	**2.6842**

robustness of the pose estimate can be ascertained from fig. 5b and 5c, where despite dim lighting and solar glare respectively, the pose errors are quite small, especially in orientation. Table V shows the mean position and orientation error for the synthetic and lightbox images of the SPEED+ dataset. These error values are lesser than other state-of-the-art direct methods and comparable to keypoint based indirect methods. The overall performance of the model can be judged from table VI. It shows out of total image samples, in how many does the pose error satisfy the accuracy requirements of table II.

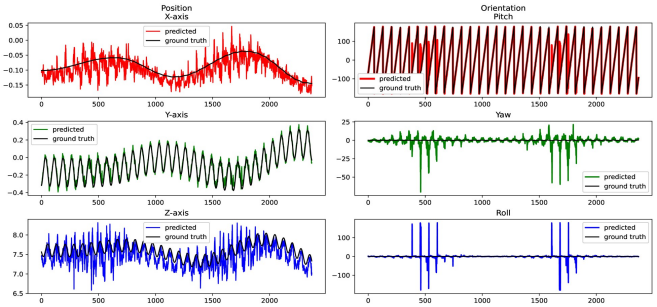
TABLE V: SPEED+ dataset initial pose estimate position and orientation errors

Image type	Position Error (m)	Orientation Error (°)
synthetic	0.096	2.79
lightbox	0.39	14.41

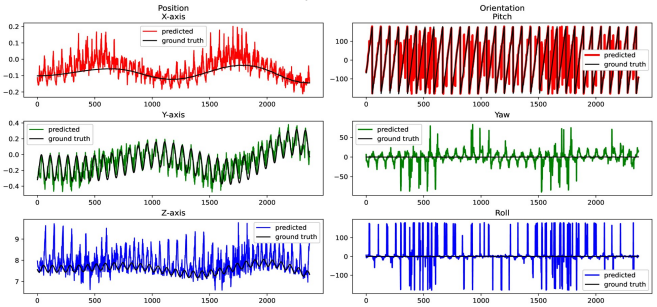
TABLE VI: SPEED+ dataset initial pose estimate success rate (in percentage)

Image type	Lateral Accuracy	Range Accuracy	Roll Accuracy	Yaw/Pitch Accuracy
synthetic	99.87	90.52	80.87	87.33
lightbox	91.08	62.76	58.31	46.31

For the initial pose estimate results on the SHIRT dataset, fig. 6 and 7 show the ground truth and estimated values of position and orientation. The graphs depict that the proposed architecture is able to closely follow the relative pose values, albeit with a noise due to varying illumination.

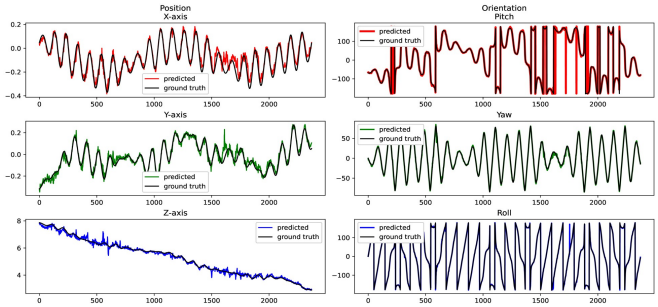


(a) ROE1 synthetic dataset.

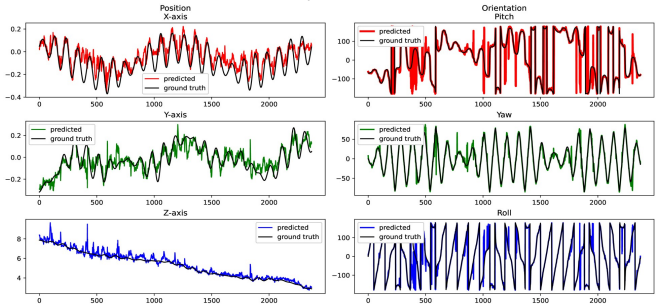


(b) ROE1 lightbox dataset.

Fig. 6: Position and Orientation plots for trajectory ROE1



(a) ROE2 synthetic dataset.



(b) ROE2 lightbox dataset.

Fig. 7: Position and Orientation plots for trajectory ROE2

B. Pose Tracking

The pose tracking values after filtering the initial pose estimates of SHIRT dataset show an improvement in accuracy and success rate. The mean error values of raw estimate and filtered pose is shown in table VII. The plots in fig. 8 and 9 compare the position and orientation errors of estimated and filtered data. Lastly, table VIII shows the improvement in accuracy of the pose values after applying the filter.

TABLE VII: SHIRT dataset initial pose estimate and filtered position and orientation errors

Image type	Estimated		Filtered	
	Position Error (m)	Orientation Error (°)	Position Error (m)	Orientation Error (°)
ROE1 synthetic	0.1979	5.863	0.1737	4.861
ROE1 *lightbox*	0.3939	27.37	0.3058	22.23
ROE2 synthetic	0.1006	3.483	0.0955	3.323
ROE2 *lightbox*	0.2358	12.905	0.2139	12.151

TABLE VIII: SHIRT dataset initial pose estimate and filtered pose success rate (in percentage)

Image type		Lateral Accuracy	Range Accuracy	Roll Accuracy	Yaw/Pitch Accuracy
ROE1 synthetic	**estimated**	100	97.47	94.47	81.44
	filtered	100	99.96	91.68	89.26
ROE1 *lightbox*	**estimated**	99.16	76.97	67.14	26.15
	filtered	100	85.82	71.57	36.37
ROE2 synthetic	**estimated**	96.65	89.38	65.74	82.87
	filtered	97.01	89.16	68.28	83
ROE2 *lightbox*	**estimated**	84.52	69.63	43.90	43.42
	filtered	84.63	70.89	44.66	44.2

VIII. Conclusion

This paper proposes a novel general purpose Deep Learning (DL) based direct pose estimation architecture, which utilizes the feature capturing abilities of the DenseNet network. As highlighted in this paper, the proposed architecture is able to give better pose estimate accuracy than state-of-the-art direct pose estimation methods as shown in table IV. For SPEED+ and SHIRT datasets also, the RNS accuracy requirements are met at a good percentage level. This it achieves without any modification in hyperparameters or network architecture for different satellites. Thus, the same model can be trained and deployed for estimating the pose of several satellites, while also not requiring any 3D model of the satellite. The parameter size of the model is also comparatively small. Integration of the pose estimation module with the adaptive navigation filter helps in increasing the robustness of the relative pose, although there is room for improvement. Besides, no other direct DL based pose estimation works have added the navigation filter to their network. Future works include improving the performance of the DL architecture across the domain gap and creating challenging datasets in the lab to further test the network for more satellites.

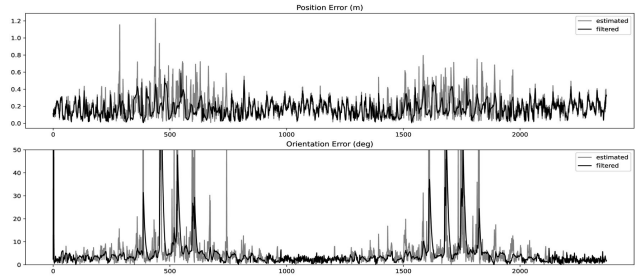
References

[1] M. Kisantal, S. Sharma, T. H. Park, D. Izzo, M. Märtens, and S. D'Amico, "Satellite pose estimation challenge: Dataset, competition design, and results," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 5, 2020.

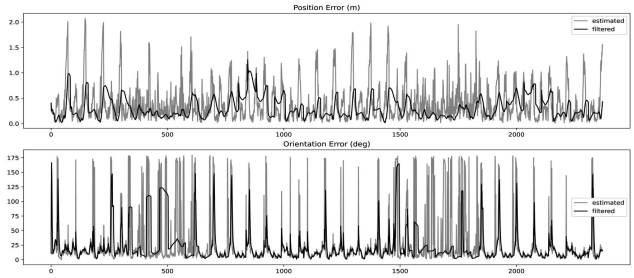
[2] S. Sharma, T. H. Park, and S. D'Amico, "Spacecraft pose estimation dataset (speed)," Stanford Digital Repository, 2019, available at https://purl.stanford.edu/dz692fn7184.

[3] P. F. Proença and Y. Gao, "Deep learning for spacecraft pose estimation from photorealistic rendering," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020.

[4] Y. Hu, S. Speierer, W. Jakob, P. Fua, and M. Salzmann, "Wide-depth-range 6d object pose estimation in space," in *Proceedings of the*

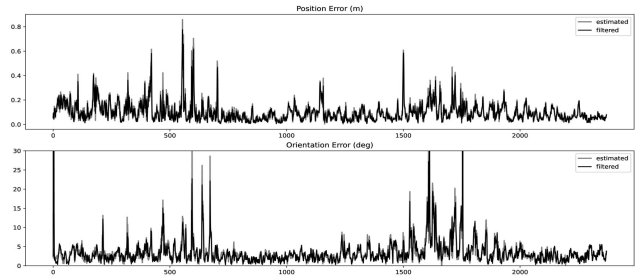


(a) ROE1 synthetic dataset.

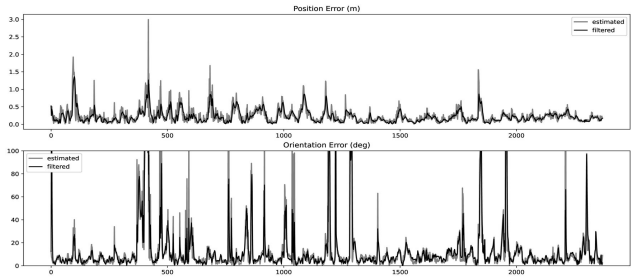


(b) ROE1 lightbox dataset.

Fig. 8: Plots for position error (top) and orientation error (bottom) for ROE1 trajectory. Initial estimate vs filtered pose.



(a) ROE2 synthetic dataset.



(b) ROE2 lightbox dataset.

Fig. 9: Plots for position error (top) and orientation error (bottom) for ROE2 trajectory. Initial estimate vs filtered pose.

IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021.

[5] T. H. Park, M. Märtens, G. Lecuyer, D. Izzo, and S. D'Amico, "Speed+: Next-generation dataset for spacecraft pose estimation across domain gap," in *2022 IEEE Aerospace Conference (AERO)*. IEEE, 2022.

[6] T. H. Park and S. D'Amico, "Shirt: Satellite hardware-in-the-loop rendezvous trajectories dataset," Stanford Digital Repository, 2022, available at https://purl.stanford.edu/zq716br5462.

[7] A. Rathinam, V. Gaudilliere, M. A. Mohamed Ali, M. Ortiz Del Castillo, L. Pauly, and D. Aouada, "SPARK 2022 Dataset : Spacecraft Detection and Trajectory Estimation," Jun. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6599762

[8] J. Song, D. Rondao, and N. Aouf, "Deep learning-based spacecraft relative navigation methods: A survey," *Acta Astronautica*, vol. 191, 2022.

[9] L. Pauly, W. Rharbaoui, C. Shneider, A. Rathinam, V. Gaudillière, and D. Aouada, "A survey on deep learning-based monocular spacecraft pose estimation: Current state, limitations and prospects," *Acta Astronautica*, vol. 212, 2023.

[10] S. Sharma and S. D'Amico, "Pose estimation for non-cooperative rendezvous using neural networks," *arXiv preprint arXiv:1906.09868*, 2019.

[11] S. Sharma and S. D'Amico, "Neural network-based pose estimation for noncooperative spacecraft rendezvous," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 6, 2020.

[12] L. Deng, H. Suo, Y. Jia, and C. Huang, "Pose estimation method for non-cooperative target based on deep learning," *Aerospace*, vol. 9, no. 12, 2022.

[13] D. Hirano, H. Kato, and T. Saito, "Deep learning based pose estimation in space," in *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS), Madrid, Spain*, 2018.

[14] R. Arakawa, Y. Matsushita, T. Hanada, Y. Yoshimura, and S. Nagasaki, "Attitude estimation of space objects using imaging observations and deep learning," in *Proc. AMOS*, 2019.

[15] T. Phisannupawong, P. Kamsing, P. Torteeka, S. Channumsin, U. Sawangwit, W. Hematulin, T. Jarawan, T. Somjit, S. Yooyen, D. Delahaye *et al.*, "Vision-based spacecraft pose estimation via a deep convolutional neural network for noncooperative docking operations," *Aerospace*, vol. 7, no. 9, 2020.

[16] 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*, 2019.

[17] T. H. Park and S. D'Amico, "Robust multi-task learning and online refinement for spacecraft pose estimation across domain gap," *Advances in Space Research*, 2023.

[18] Z. Wang, M. Chen, Y. Guo, Z. Li, and Q. Yu, "Bridging the domain gap in satellite pose estimation: a self-training approach based on geometrical constraints," *IEEE Transactions on Aerospace and Electronic Systems*, 2023.

[19] J. I. B. Pérez-Villar, Á. García-Martín, and J. Bescós, "Spacecraft pose estimation based on unsupervised domain adaptation and on a 3d-guided loss combination," in *European Conference on Computer Vision*. Springer, 2022.

[20] L. Pasqualetto Cassinis, R. Fonod, E. Gill, I. Ahrns, and J. Gil Fernandez, "Cnn-based pose estimation system for close-proximity operations around uncooperative spacecraft," in *AIAA Scitech 2020 Forum*, 2020.

[21] T. H. Park and S. D'Amico, "Adaptive neural-network-based unscented kalman filter for robust pose tracking of noncooperative spacecraft," *Journal of Guidance, Control, and Dynamics*, vol. 46, no. 9, 2023.

[22] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[23] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017.

[24] B. J. Naasz, R. D. Burns, S. Z. Queen, J. Van Eepoel, J. Hannah, and E. Skelton, "The hst sm4 relative navigation sensor system: overview and preliminary testing results from the flight robotics lab," *The Journal of the Astronautical Sciences*, vol. 57, no. 1-2, 2009.

[25] J. Posso, G. Bois, and Y. Savaria, "Mobile-ursonet: an embeddable neural network for onboard spacecraft pose estimation," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2022.

[26] A. Garcia, M. A. Musallam, V. Gaudilliere, E. Ghorbel, K. Al Ismaeil, M. Perez, and D. Aouada, "Lspnet: A 2d localization-oriented spacecraft pose estimation neural network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.