It's All About Attitude: Monocular Spacecraft Pose Estimation

Andrew Constantinescu[1] and Loïc J. Azzalini[1]

***Abstract*— Satellite servicing tasks and debris removal missions are just a few of many active maintenance operations in orbit that require highly accurate maneuvering. For this reason, there exists a need for robust full pose estimation of spacecrafts given on-board perception sensors. This paper presents a method for directly regressing the 6D pose of a satellite from a single grayscale image. We experiment with three different attitude descriptors for regressing the orientation and compare their results. The Spacecraft PosE Estimation Dataset (SPEED) provided in the Kelvin's Satellite Pose Estimation Challenge is used to train and evaluate the different approaches. Our code is available at `https://github.com/ljazzal/spacecraft-pose-estimation`**

I. INTRODUCTION

Launching spacecrafts in orbit is an expensive endeavour and accessing them once they are in space is challenging, if not impossible. There has been a lot of interest in recent years to add active maintenance to the satellite life cycle, potentially expanding the life expectancy of space assets (*e.g.*, OSAM-1 [1]). Whether it is refueling, repairs or even removal, the maintenance of space vehicles and their orbital environment is seen as a crucial undertaking. Typically, the target satellite is known beforehand and a detailed 3D model of the vehicle is available for close-proximity operation. However, in the case of a space debris or defunct satellite, exact models might not be available, warranting the use of generalizable techniques that only rely on scene information. Perception algorithms such as object detection and pose estimation could provide an estimate of the state of the target object.

The Kelvins Satellite Pose Estimation Challenge [2] was designed for the purpose of evaluating monocular vision-based approaches as a means for pose estimation. The challenge is based on the Prisma project which consists of two satellites, Tango and Mango, that orbit the Earth together in close proximity. In 2019, a synthetic image dataset was created with 3D renderings of the Tango spacecraft to mimic observations of Tango from the Mango satellite [3]. This dataset contains images of varying lighting and distances from the observing satellite, with either a black space or Earth background. The training dataset consists of images of size $(1920, 1200)$, along with a translation vector and a unit quaternion vector representing the ground truth pose. The camera intrinsics are the same for all the images and are provided as part of the dataset. Given a single grayscale image from a monocular camera on-board Mango, our goal is to fully determine the relative position and orientation of the Tango satellite. In this study, we restrict the amount of usable information about the target satellite to allow generalization to other space objects. This vision problem accepts both a classical and/or deep learning treatment and consists of a relatively new extension of the field.

[1]The authors are with the University of Toronto Institute for Aerospace Studies, Toronto, Canada `{first name}.{last name}@mail.utoronto.ca`

II. RELATED WORK

Solutions to the pose estimation problem have successfully been demonstrated throughout the computer vision and robotics literature, from 3D bounding box estimation in self-driving vehicle applications [4]–[6] to the full 6DOF object pose estimation required for robotic grasping tasks [7], [8]. These fields offer a rich history of transferable solutions and techniques that can be applied to spaceborne applications. In particular, these methods can help address the unique challenges presented by the space environment in terms of scarce scene information, lighting conditions and domain adaptation.

A. 3D bounding box estimation

In the self-driving literature, the application itself allows for the reduction from 6DOF to 3DOF estimation through geometric constraints. Mousavian *et al.* [5] use a 2D bounding box and mean estimates of the 3D bounding box dimensions to regress the orientation and dimensions of the final 3D model; they assume that the 3D bounding box fits tightly within the 2D bounding box. These solutions typically make use of *MultiBin* architectures where the orientation and dimensions are regressed in different branches (but share a common backbone). Given the large domain of the orientation target, orientation regression is turned into a classification task by discretizing each orientation DOF, with each bin centered about a mean orientation angle, leaving the residual to be learned. These two techniques are popular in the self-driving literature [4] and while they cannot directly be applied to this problem due to restricted information about the target object, they provide some inspiration for the derivation of suitable geometric priors.

B. 6D object pose estimation

Robotic grasping tasks use accurate 3D models of the target objects to guide the learning of equivalent constraints. The pose estimation is carried out by matching feature points between the model and the scene. Several works combine detection, segmentation and pose estimation from a single RGB image into an end-to-end learning framework [7]–[9]. While segmentation is effective in cluttered environments, it becomes trivial in a space environment where most features belong to the object of interest. Therefore, we ignore the

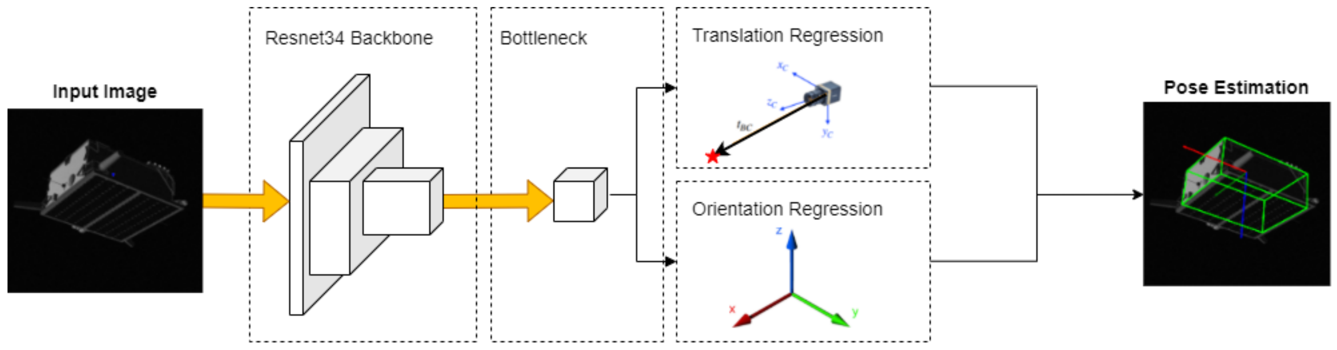


Fig. 1. **Network Architecture**. *Yellow arrow:* Convolution layer. *Black arrow:* Fully-connected layer.

segmentation component of these works and focus on their ability to regress directly from the image to the 6D pose [7], [9]. Both works decouple the translation and orientation due to their distinct influence on the scene appearance, allowing the network to learn the dependencies between each pose component and the visual features.

C. 6D pose regression

Both of the fields mentioned above can leverage the additional information provided at the input to reduce the scope of the estimation problem. Given the lack of such information in the present study, our work can be compared to end-to-end camera pose regressors [10], whereby the relative pose of a camera is determined from scene information captured in a single RGB image. The main difficulty that comes with this approach is the derivation of a geometric loss function that adequately captures both position and orientation errors, which are being regressed separately. Given that a simple linear weighted sum of the errors is known to provide limited accuracy, [11] propose an optimal weighting scheme to balance their direct unit quaternion and translation regression.

D. Spacecraft pose estimation

The Kelvins Challenge attracted several research groups with diverse backgrounds in computer vision and space engineering and has allowed a great transfer of knowledge from the aforementioned areas to the problem of spacecraft pose estimation. [12] use multi-view triangulation to construct a 3D model of the target satellite and learn the position of pre-defined landmarks (*i.e.*, hand-crafted features of the target). In that sense, their approach is similar to [8], where a 3D model is assumed to be known *a priori*. They then solve the 2D-3D correspondence problem via robust geometric optimization. Hybrid approaches combining neural network keypoint regression and Perspective-n-point (PnP) pose solvers appeared to be a winning strategy, as top-scoring teams followed a similar approach [12], [13]. End-to-end deep learning approaches generally do not deliver as high an accuracy as they lack the geometric information that correspondence-based approaches afford. Nonetheless, [14] achieved competitive results with a deep learning framework that directly regresses the translation and solves for the orientation via soft classification. The latter allows them to fit a Gaussian mixture model to a discretized target domain, similar to the *MultiBin* approach [5]. A similar method was used in [13] to estimate the orientation of the object by classifying pre-defined uniformly distributed random rotations.

E. Contributions

While similar studies do consider directly regressing the pose via translation and rotation branches, they quickly dismiss the idea for its lack of geometrical grounding. Yet, direct regression has been shown to be successful in other fields [7], [9], [10] and we believe its application to the problem under study can be pushed further. Furthermore, unit quaternions, being free of singularities, have become a popular means to describe attitude in computer vision. However, there exist many alternative ways to describe a 3D rotation which, through their characteristics, could prove advantageous for the problem at hand [15]. Our contributions consist of directly regressing the relative translation and orientation of the target satellite without prior 3D information of the object and comparing the performance of three attitude descriptors: Euler angles, axis-angle and unit-quaternions.

III. APPROACH

A. Direct regression network

As mentioned already, the two most common methods of estimating 6D pose are: (i) predicting the 2D location of 3D keypoints from the object model and optimizing their correspondence using a PnP solver, or (ii) regressing the pose of a target object directly from the image. In this work, we will adopt the latter approach and make use of a deep convolutional neural network (CNN) to regress the translation and orientation vectors in separate branches. A pre-trained ResNet-34 is used as the common backbone as depicted in Figure 1. In both pose estimation approaches, the first step is normally to detect the object whose pose we are interested in. Given that this dataset only deals with a single target, we do not need to carry out this step, greatly simplifying our implementation. Instead, to mimic the output of an object detector (*e.g.*, YOLOv3) upstream from our network, we work out what a suitable 2D detection box would look like in the following section.

B. Data augmentation

The training set from the SPEED dataset consists of images taken from a monocular camera with intrinsic parameters K and the ground truth, represented as a three-dimensional translation vector along with a four-dimension orientation vector expressed in unit quaternions. Additionally, we choose to construct 3D wireframe bounding boxes and 2D detection boxes given the provided ground truth data. This will be used in later stages of our implementation such as in data transformations and translation regression. A detailed CAD model of the Tango satellite is not available to us, so we approximate the geometry of the satellite as a rectangular prism with dimensions $570mm \times 740mm \times 295mm$. We combine the ground truth data with these approximate dimensions to determine the 8 vertices required to define the satellite's wireframe in each image. We define the 2D detection box by only two points, which we obtain by taking the minimum x, minimum y and the maximum x, maximum y components of all the wireframe coordinate points.

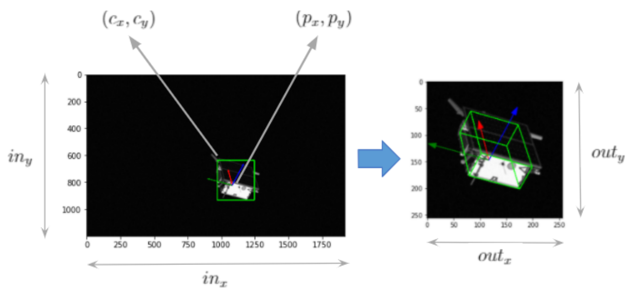


Fig. 2. *ResCrop* Transformation

We implement a rescaling and cropping transformation on the training images from the dataset which we refer to as the *ResCrop* transformation and illustrate in Figure 2. The purpose of this transformation is to eliminate as much background information as possible while also scaling the image to correspond to the required square-shaped $(256, 256)$ input size of the convolutional network. First, we take the two points used in defining the 2D detection box to compute its side-lengths (s_x, s_y). We take the maximum of the two lengths and apply a safety factor f_S to determine the side-length of the crop. The safety factor is applied for images such as Figure 3, where the 2D detection box does a poor job at capturing the entire satellite.

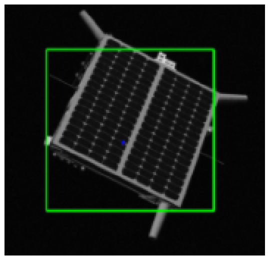


Fig. 3. Poor detection box on the satellite.

This factor helps us ensure that we do not unintentionally crop pixels belonging to the satellite. If the satellite spans a large portion of the image, the resulting crop will be minimal. In contrast, if the satellite in the original image is small, the crop will help remove a significant amount of irrelevant background data. The crop is performed about the centre of the bounding box, which typically does not correspond to the centre of the original image. For this reason, the principal point of the camera intrinsics needs to be adjusted. The final step of the transformation is to rescale the cropped image to a size of $(256, 256)$. Both cropping and rescaling operations are effectively resizing the image in a manner that impacts the angle of view and its magnification. The original camera intrinsics matrix K is shown in (1). We scale the focal length and principal point accordingly in our new camera intrinsics matrix K_{rc} as shown in (2).

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

$$K_{rc} = \begin{bmatrix} \frac{out_x}{in_x} f_x & 0 & \frac{out_x}{in_x}[\frac{d}{2} + (c_x - p_x)] \\ 0 & \frac{out_y}{in_y} f_y & \frac{out_y}{in_y}[\frac{d}{2} + (c_y - p_y)] \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

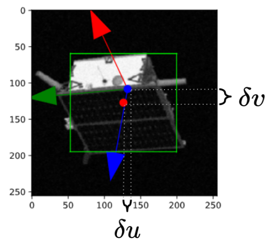
In these matrices, (in_x, in_y) represents the dimensions of the original image as $(1920, 1200)$, (out_x, out_y) represents the dimensions of the output image as $(256, 256)$, (c_x, c_y) represents the principal point of the input image, (p_x, p_y) represents the center point of the cropping operation and d represents the side-length of the cropped image. A sample image undergoing a *ResCrop* transformation is shown in Figure 2. Pseudocode for the *ResCrop* transformation is shown in Algorithm 1. In the pseudocode below, we define the *crop* function to output *img* cropped about the centre-point $(top, left)$ with side-lengths (d, d). Similarly, we define the *resize* function to output *img* rescaled with side-lengths (out_x, out_y).

Algorithm 1 *ResCrop* Transform

```
1: $d = f_S \times max(s_x, s_y)$
2: $top = c_y - d/2$
3: $left = c_x - d/2$
4: $img = crop(img, top, left, d, d)$
5: $img = resize(img, out_x, out_y)$
```

C. Translation

We have assumed that a 2D detection box is generated upstream from the pose estimation network. This is a suitable assumption given how readily available object detection algorithms have become in recent years. Similar to [7], [9], we use the geometry of the 2D box to constrain the regression of the 3DOF in translation whose domain is unbounded. In this dataset, the origin of the satellite body frame coincides with the center of the bottom plate (hidden in Figure 4).



$$\begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} \frac{u+\delta u - c'_x t_z}{f'_x} \\ \frac{v+\delta v - c'_y t_z}{f'_y} \end{bmatrix}$$

Fig. 4. Constraining the translation regression

We notice that while the center of the bounding box (red point) is offset from the actual body frame origin (blue point), it makes for a suitable initial guess. We decide to express t_x and t_y in terms of the offset δv and δu by relating them through the camera intrinsics K_{rc} (in Figure 4, the prime denotes the intrinsic parameters from K_{rc}). This approach significantly restricts the regression of the translation vector, without being trivial as the distance to the camera frame t_z still needs to be regressed in full. A normalized L2 loss is then used for comparison to the ground truth as expressed in (3).

$$L_t = \mathbf{d}(\mathbf{t}, \hat{\mathbf{t}}) = \frac{\|\mathbf{t} - \hat{\mathbf{t}}\|_2}{\|\mathbf{t}\|_2} \tag{3}$$

D. Attitude descriptors

Orientation regression is significantly harder than translation due to the varying properties in different attitude descriptions [15]. The main trade-off is choosing between a representation that comes with singularities or another that consists of constraints. For this reason, we decide to experiment with three different parameterizations of varying characteristics. Specifically we explore unit quaternions to match the provided ground truth data for orientation, we explore Euler angles to avoid imposing constraints, and finally we explore the axis-angle representation.

1) Unit quaternions: While unit quaternions do not exhibit any singularities, they consist of four real numbers that are constrained to have a magnitude of one on a 4D hypersphere. Coming up with a suitable metric to regress unit quaternions is challenging due to the definition of distance on a sphere being more complex than on a plane. By assuming small changes in orientation at each iteration, the distance can be approximated as shown by the loss function in (4), where the angle brackets represent the dot product operator between the vectors $\mathbf{q}$ and $\hat{\mathbf{q}}$.

$$L_q = \mathbf{d}(\mathbf{q}, \hat{\mathbf{q}}) = 1 - \langle \mathbf{q}, \hat{\mathbf{q}} \rangle^2 \tag{4}$$

A unit quaternion can equivalently be expressed as a Direction Cosine Matrix (DCM) with (5).

$$\mathrm{C}_q = \begin{bmatrix} 2q_0^2 - 1 + 2q_1^2 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 \\ 2q_1q_2 - 2q_0q_3 & 2q_0^2 - 1 + 2q_2^2 & 2q_2q_3 + 2q_0q_1 \\ 2q_1q_3 + 2q_0q_2 & 2q_2q_3 - 2q_0q_1 & 2q_0^2 - 1 + 2q_3^2 \end{bmatrix} \tag{5}$$

2) Euler angles: Euler angles consist of exactly three parameters to express 3D rotations, but suffer from singularities. The Euler angles can be extracted from a DCM, such as the popular 3-2-1 sequence, as expressed in (6).

$$\begin{gathered}
\mathbf{C}^{321} = \begin{bmatrix} c_1c_2 & s_1c_2 & -s_2 \\ -s_1c_3 + c_1s_2s_3 & c_1c_3 + s_1s_2s_3 & c_2s_3 \\ s_1s_3 + c_1s_2c_3 & -c_1s_3 + s_1s_2c_3 & c_2c_3 \end{bmatrix} \\
c_i = cos\theta_i, \quad s_i = sin\theta_i \\
\theta_1 = tan^{-1}\left(\frac{\mathbf{C}^{321}_{Euler}[0][1]}{\mathbf{C}^{321}_{Euler}[0][0]}\right) \\
\theta_2 = -sin^{-1}\left(\mathbf{C}^{321}_{Euler}[0][2]\right) \\
\theta_3 = tan^{-1}\left(\frac{\mathbf{C}^{321}_{Euler}[1][2]}{\mathbf{C}^{321}_{Euler}[2][2]}\right)
\end{gathered} \tag{6}$$

We then use the ground truth, which is now represented using Euler angles, to compute the orientation loss with the three regressed parameters as per the pseudocode Algorithm 2.

Algorithm 2 Euler Angle Loss: $L(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})$

```
1: Correct the estimate to be in bounds $[-\pi, \pi]$
2: $\hat{\boldsymbol{\theta}} \leftarrow (\hat{\boldsymbol{\theta}} + \pi)\%(2\pi) - \pi$
3: $loss \leftarrow \boldsymbol{\theta} - \hat{\boldsymbol{\theta}}$
4: Correct the loss to be in bounds $[0, \pi]$
5: if $loss > \pi$ then
6:     $loss \leftarrow 2\pi - loss$
7: end if
```

Due to the fact that the regressed parameters are unbounded, yet Euler angles consist of a wrap-around in increments of 2π, we must ensure that the loss function is representative of the true distance between angles. This is captured in the loss function by first correcting the estimated parameters to be in the bounds of $[-\pi, \pi]$. If we simply take the difference of the ground truth and the corrected parameters, we are guaranteed to obtain a result in the bounds of $[0, 2\pi]$. However, we know that the two Euler angles can be separated by a maximum of π radians, so we apply the final correction to the loss function in line 6 of the pseudocode. The total orientation loss of this parameterization is the summation of the angle loss of each of the three dimensions, as per (7).

$$L_{Euler} = \mathbf{d}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = \sum_{i=1}^{3} L(\theta_i, \hat{\theta}_i) \tag{7}$$

3) Euler axis-angle: The axis-angle parameterization of a rotation consists of a unit vector ν representing the axis of rotation and a scalar quantity θ representing the magnitude of rotation about the axis ν. With this 4-parameter representation, we avoid singularities but have to deal with the unit constraint on the axis of rotation. We handle this by using the special orthogonal rotation group to define the matrix $\hat{\mathbf{R}}$ as per (8), which is simply the exponential map applied to the rotation vector $\theta\nu$ output by the network.

$$\hat{\mathbf{R}} = exp(\theta[\nu]^{\times}) \tag{8}$$

We ensure that $\nu \in [-1, 1]$ and $\theta \in [0, \pi)$ by using a $\tanh$ activation function and scaling the output by π. This will ensure that the constructed $\hat{\mathbf{R}}$ matrix belongs to the $SO(3)$ group and we have a valid axis-angle parameterization [6]. The special orthogonal rotation group is defined by $SO(3) = \{R : R \in \mathbb{R}^{3\times3}, R^T R = I_3, det(R) = 1\}$, whose constraints we use to define the loss function given by (9).

$$L_{axis\text{-}angle} = \mathbf{d}(\mathbf{R}, \hat{\mathbf{R}}) = \cos^{-1}\left(\frac{tr(\mathbf{R}^T\hat{\mathbf{R}}) - 1}{2}\right) \quad (9)$$

Given that the ground truth is a unit quaternion $q = q_0 + \mathbf{x}$, where $\mathbf{x} = [q_1, q_2, q_3]^T$, we obtain the ground truth rotation matrix $\mathbf{R}$ in (9) via the unit quaternion to DCM conversion shown in (5).

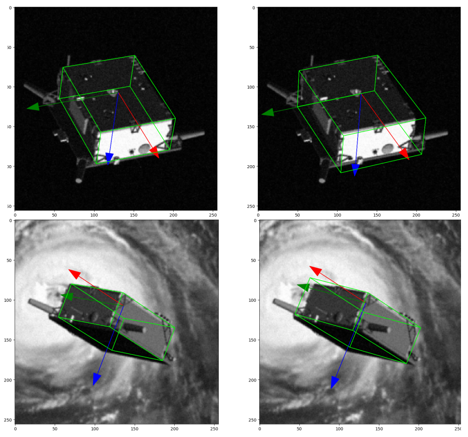


Fig. 5. Examples of accurate pose recovery with the ground truth on the left and the prediction on the right.

IV. EXPERIMENTAL EVALUATION

A. Implementation and Training Details

All of our experiments were conducted on Stanford's benchmark dataset SPEED [3]. SPEED consists of a training set of 12000 synthetic images and 5 real images along with a testing set of 2998 synthetic images and 300 real images. *Real* is used to denote images of a scale model of the Tango spacecraft taken by the Space Rendezvous Laboratory at Stanford, as opposed to the synthetic renderings that make up the majority of the dataset [3]. Due to the fact that there are very few real training images and ground truth data is provided only for training images, we conducted our own experimental comparison of the three attitude descriptors by splitting the synthetic training images into a training set and test set. However we were able to make use of the testing set through a submission to the Kelvin's challenge as explained in section IV-C.

All networks were trained either on a NVIDIA GeForce GTX 1050 or through one of the several GPUs available through Google Colab. We trained on 5000 shuffled images from the 12000 synthetic images provided in the dataset due to the limitations in computational hardware available to us. Initially, we used SGD with a learning rate of 0.0001, a weight decay of 0.01, a momentum of 0.9 and a batch size of 16 images. We found that by making the learning rate variable, we were able to slightly improve on our results. We initialized the training with a learning rate of 0.1 for the translation branch and 0.001 for the orientation branch but then we scale them both by a factor of 0.5 every 25 epochs, until convergence. We found that 100 epochs are sufficient for the network to converge.

B. Test Set Evaluation

Figure 5 shows examples of the output of our network. Our different implementations were benchmarked according to the scoring metrics used by the original competition. The equations (10)-(13) are used to evaluate the predicted translation and orientation against the ground truth. The quantitative results are recorded in Table I, while a qualitative comparison of the performance of the different attitude descriptors is depicted in Figure 6.

$$score^{(i)}_{position} = \frac{\|r^{(i)}_{gt} - r^{(i)}_{est}\|_2}{\|r^{(i)}_{gt}\|_2} \quad (10)$$

$$score^{(i)}_{orientation} = 2\cos^{-1}\left(\left|\left\langle q^{(i)}_{est}, q^{(i)}_{gt}\right\rangle\right|\right) \quad (11)$$

$$score^{(i)}_{pose} = score^{(i)}_{position} + score^{(i)}_{orientation} \quad (12)$$

$$score = \frac{1}{N}\sum_{i=1}^{N} score^{(i)}_{pose} \quad (13)$$

All three methods used the same translation regression scheme, suggesting that discrepancies observed in the top-left plot are caused by the balancing of the translation and orientation errors which affects training performance.

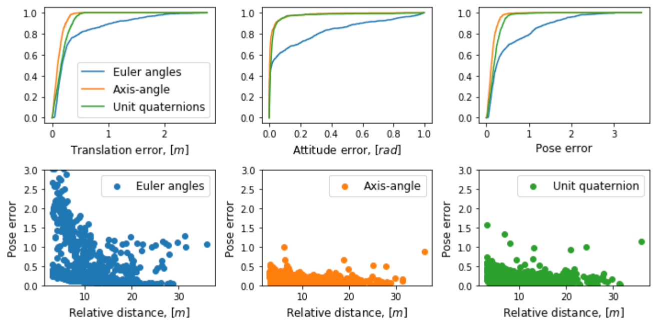


Fig. 6. **Attitude descriptor comparison**. *Top*: Cumulative histograms of translation, orientation and total errors. The axis-angle method performed best as seen by the sharpest slope in all three plots. *Bottom*: Plots of the pose error against the relative distance to the target, showing the effectiveness of the *ResCrop* transform at standardizing the regression process.

The cumulative histogram captures the error spread in the steepness of its slope, suggesting that the axis-angle outperforms the other two methods. This can also be observed in the more contained relationship between pose error and distance to the target (bottom center), whose adverse effect was mostly cancelled by the *ResCrop* transform. Nonetheless, a slight downward trend can be observed for small distances in all three bottom plots; this effect is due to (10), which penalizes position errors more heavily when the target is close. This trend has also been observed in similar works [13]. The mean and standard deviation of each error are provided below.

TABLE I

COMPARISON OF ATTITUDE DESCRIPTION ON THE TEST SET

Attitude descriptor	**score**$_{\text{position}}$	**score**$_{\text{orientation}}$	**score**$_{\text{pose}}$
Euler angles	0.376 ± 0.485	0.600 ± 0.647	0.976 ± 0.951
Axis-angle	0.119 ± 0.0871	0.191 ± 0.230	0.310 ± 0.239
Unit quaternions	0.180 ± 0.123	0.263 ± 0.246	0.443 ± 0.280

The main advantage of the axis-angle method is the handling of constraints through the $SO(3)$ group definition and the accurate error metric it supports via (9). In comparison, it is difficult to represent an accurate and tractable geodesic error metric for unit quaternions, leaving us with the approximation in (4). While a $\tanh$ activation improves the results by bounding the target domain, a normalization step is still required to guarantee that the unit constraint is met. The normalization step, however, is not reflected in the optimization process and as a result, we do not penalize for illegal rotation descriptions. A similar comment can be made about the limitations of the Euler angle loss metric (7) in reflecting true rotation error. While singularities could easily be handled, the lack of constraints (*i.e.*, as a result of the periodicity of the Euler angles) affects our ability to penalize illegal rotations. Given more time, we could have addressed these limitations by introducing appropriate regularization terms to each loss function.

C. Kelvin's Challenge

The last piece of evaluation conducted on our implementation is submitting to the Kelvin's Pose Estimation Challenge [2] to compare our results to the other teams that have participated. The original challenge took place from February 1st to July 1st, 2019 however a post-mortem version of the challenge was created with a separate leaderboard to continue benchmarking [16]. Our methods were evaluated on 2998 synthetic images and 300 *real* images, both of which were unseen during training. Our best total score is 0.8072, placing us in 10th for the post-mortem challenge and 11th in comparison to the submissions from the original challenge (Table II) [17].

Out of the top four, three teams used PnP solvers to achieve results under 0.1 for the synthetic dataset, suggesting that solving for 2D-3D correspondences is a more rigorous approach. We strongly believe that additional computational power would allow us to train on the entire dataset, potentially allowing for a better generalization to the test set.

TABLE II

SAMPLE RESULTS FROM THE ORIGINAL 48-TEAM COMPETITION

Team	E_{syn}	E_{real}
1. UniAdelaide [12]	**0.0094**	0.3752
2. EPFL_cvlab	0.0215	**0.1140**
3. pedro_fairspace [14]	0.0571	0.1555
4. stanford_slab [13]	0.0626	0.3951
⋮		
10. VSI_Feeney	0.4658	1.5993
11. **astrotrons (ours)**	0.8072	1.4003

V. CONCLUSIONS

This paper proposes a method for directly regressing the full 6D pose of a spacecraft given a single image. We split the regression into position and orientation branches and explore three different attitude descriptors. Experiments reveal that the axis-angle parameterization for orientation leads to the overall best results. We attribute this result to the fact that the loss function for this representation captures the true error most accurately and thus, is able to train a better performing network. In future iterations, we could imagine incorporating object detection (instead of using ground truth information) and learning the intrinsic parameters of K_{rc} (instead of preprocessing the data). The next steps for this project also include adaptation to different domains. It would be desirable to provide solutions capable of estimating the pose of various satellites, space debris or other objects for which a model may be unknown. This type of generalization is important when bridging the gap from synthetic images to images being taken in outer space.

REFERENCES

[1] "Nasa's exploration & in-space services." [Online]. Available: https://nexis.gsfc.nasa.gov/OSAM-1.html

[2] "European space agency, kelvins - esa's advanced concepts competition website." [Online]. Available: https://kelvins.esa.int/satellite-pose-estimation-challenge/

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

[4] C. Sahin, G. Garcia-Hernando, J. Sock, and T.-K. Kim, "A review on object pose recovery: From 3d bounding box detectors to full 6d pose estimators," *Image and Vision Computing*, vol. 96, p. 103898, 2020.

[5] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3d bounding box estimation using deep learning and geometry," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[6] S. Mahendran, H. Ali, and R. Vidal, "3d pose regression using convolutional neural networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017.

[7] T.-T. Do, M. Cai, T. Pham, and I. Reid, "Deep-6dpose: Recovering 6d object pose from a single rgb image," 2018.

[8] Y. Hu, J. Hugonot, P. Fua, and M. Salzmann, "Segmentation-driven 6d object pose estimation," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[9] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *Robotics: Science and Systems XIV*, 2018.

[10] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.

[11] A. Kendall and R. Cipolla, "Geometric loss functions for camera pose regression with deep learning," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[12] B. Chen, J. Cao, A. Parra, and T.-J. Chin, "Satellite pose estimation with deep landmark regression and nonlinear pose refinement," *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2019.

[13] T. H. Park, S. Sharma, and S. D'Amico, "Towards robust learning-based pose estimation of noncooperative spacecraft," *ArXiv*, vol. abs/1909.00392, 2019.

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

[15] D. Q. Huynh, "Metrics for 3d rotations: Comparison and analysis," *Journal of Mathematical Imaging and Vision*, vol. 35, no. 2, p. 155–164, 2009.

[16] "Leaderboard." [Online]. Available: https://kelvins.esa.int/pose-estimation-challenge-post-mortem/leaderboard/

[17] M. Kisantal, S. Sharma, T. H. Park, D. Izzo, M. Martens, and S. Damico, "Satellite pose estimation challenge: Dataset, competition design, and results," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 5, p. 4083–4098, 2020.