AI-Assisted Near-Field Monocular Monostatic Pose Estimation of Spacecraft

Daigo Kobayashi, Alexander Burton, and Carolin Frueh
School of Aeronautics and Astronautics, Purdue University

Abstract

Accurately estimating the pose of human-made spacecraft is indispensable for various space applications such as in-orbit servicing, active debris removal, and health monitoring. However, achieving precise attitude estimation remains a challenging task. Resolved imaging via optical sensors can be achieved in the near field. Among the various methods, monostatic monocular imaging is the most practical, yet it presents significant challenges for pose estimation since only a single viewing direction per image is available. One part of that challenge is the fact that pose includes both the attitude of and distance to the object, as both are fully coupled. Existing research has addressed this issue by focusing on attitude estimation in the near field, leveraging machine learning techniques at various stages of the process.

In this paper, the pose of a spacecraft is estimated from a single image. In the first step, keypoint detection in a single image is performed. It is assumed that the keypoint locations in the spacecraft's body-fixed frame are fully known, which corresponds to perfect knowledge of the observed spacecraft a priori, for example, via a CAD model. Four deep neural network models are tested and compared against each other from two different detection classes: regression and heatmap. Keypoint detection is the precursor to the subsequent pose estimation. Next, the position and attitude of the spacecraft relative to the observer are estimated. The baseline method implemented is the Efficient Perspective-n-Point (EPnP) algorithm. The EPnP has been explored in performance, and implementation has been optimized for our given dataset. The second method used is a particle swarm optimizer (PSO), which has not been applied to the monostatic spacecraft pose estimation problem before. Although the keypoint detection methods rely on machine learning, neither EPnP nor PSO are learning-based. The PSO is shown to produce more accurate pose estimates than the EPnP and is also expected to be more durable against imaging errors.

1. INTRODUCTION

Rendezvous and proximity operations such as in-orbit servicing, debris removal, and health monitoring require the autonomous and real-time estimation of a target object's six-degree-of-freedom (6-DoF) pose. This estimation may be performed by using onboard sensors on a chaser satellite. Among the possible choices for the sensors are Light Detection and Ranging (LiDAR), Range Detection and Ranging (RADAR), and stereo vision sensors. However, due to the low Size-Weight-Power-Cost (SwaP-C) requirements in spacecraft, a monocular monostatic camera often becomes the only feasible option in such operations. The advantage of such a setup is that resolved optical images can be obtained because of the relatively short distance between the chaser and the observed spacecraft in contrast in the near-field, rather than non-resolved far-field observations, e.g., from the ground. It must be highlighted that the attitude estimate is coupled with the estimated relative distance between the target and the observer, even within the resolved near-field domain. In this paper, we use the term *pose* as an umbrella term.

Numerous studies have been dedicated to addressing the 6D pose estimation of a target object using image data. Traditional approaches [1, 2, 3, 4] rely on image processing techniques to identify the salient features of the target and then solve the Perspective-n-Point (PnP) problem to estimate the pose by matching the keypoints in an image with known keypoints on a known shape-albedo model of the spacecraft. More recent studies predominantly use deep neural network models to achieve more robust pose estimation. A prevailing strategy involves one deep neural network model to detect a bounding box, thereby cropping out the region of interest, followed by another deep neural network model to identify keypoints. Subsequently, the PnP problem is solved based on the detected keypoints [5, 6, 7].

Unfortunately, the earlier methods suffer from the limited accuracy of realistic imaging, such as in the presence of image noise, varying contrast, and diverse backgrounds as commonly encountered in spaceborne images. Thus, efforts

have been made to address the domain gap problem arising from dissimilarities between synthetic training datasets and real spaceborne images. Notably, the winning teams in the 2021 Satellite Pose Estimation Competition [8, 9] utilized adversarial learning techniques to enhance robustness when dealing with out-of-distribution datasets, with the aim to generalize the model's performance across diverse conditions and improve its applicability in practical scenarios.

In this paper, particle swarm optimization is used to accurately estimate the known spacecraft pose from a set of detected keypoint locations in an image. The results are compared and benchmarked against the EPnP algorithm [10], which has also been fully implemented and run on the same dataset under the exact same conditions.

Before finding the pose estimate, keypoints have to be detected in each image. In this paper, four different keypoint detection models are tested and compared. The first model is ResNet [11], one of the most popular regression models in computer vision. The second and third models are HRNet [12] and HRFormer [13], which are widely-used heatmap models. The fourth model is RTMPose [14], which is expected to outperform the heatmap models.

The organization of this paper is as follows: Section 2 describes the detailed steps of the pose estimation process for a known target. This section encompasses the overview of the method in Section 2.1 and explains the details of the dataset in Section 2.2. Additionally, the object detection method is detailed in Section 2.3, followed by the keypoint detection method in Section 2.4, and the pose estimation methods in Section 2.5. To conclude the investigation, Section 2.6 presents a comparative analysis of the results obtained from the EPnP and the PSO algorithms. Finally, the paper ends in Section 3, with some concluding remarks.

2. POSE ESTIMATION FOR KNOWN TARGET

2.1 Overview

The pose estimation process for a known spacecraft is implemented in three steps, as shown in Figure 1. First, the bounding box of the target object is estimated by an object detection network, YOLOv8 [15]. The detected bounding box is used to crop the image after adding a 10-pixel margin. Subsequently, the keypoints of the object are located by using a keypoint detection network model, RTMPose [14]. Finally, the object's pose in the image is estimated by solving the Perspective-n-Point (PnP) problem, given the keypoint locations in both the image and body-fixed frames. The PnP problem is solved via two different methods, the Particle Swarm Optimizer (PSO) and the EPnP algorithm [16], where the latter serves as a baseline.

2.2 Dataset

The image dataset is generated by a fast rendering tool developed in-house by the Space Information Dynamics Group at Purdue University. The test object featured in this paper is the Transiting Exoplanet Survey Satellite (TESS) [17] satellite with a size of just under 4 meters. The TESS satellite is almost symmetric in a box-wing configuration, providing a good baseline for testing and exploring the features of the algorithm.

First, images of TESS are rendered by a ray tracing algorithm with random rotations and illumination conditions. Subsequently, images of the Earth from the Himawari-8 dataset [18] are superimposed onto the background of TESS images to increase realism. In total, 5,000 images are generated and split into 4,000, 900, and 100 images for training, validation, and testing respectively. These images are primarily meant to train, validate, and test the keypoint detection method. However, the validation dataset also provided the input for testing the PSO and EPnP algorithms.

The images are rendered using a perspective camera model instead of an orthographic camera model. A perspective camera scales the object's size and position depending on the depth, whereas an orthographic camera renders objects at the same scale regardless of the depth. The perspective camera model gives more realistic images while helping the EPnP solver give more accurate results because the keypoint locations preserve the depth information.

For this paper, all images have been generated assuming that the observer satellite features a camera focal length is 21.9 meters with a field of view of 25×25 degrees. The phase angle of the sunlight is constrained to below 90 degrees, avoiding extremely dark conditions. The object range is set between 18 and 40 meters so that the object is always resolved while fully viewed in an image, even at the closest possible distance. Fig.2 shows a few examples of the rendered satellite images of TESS.

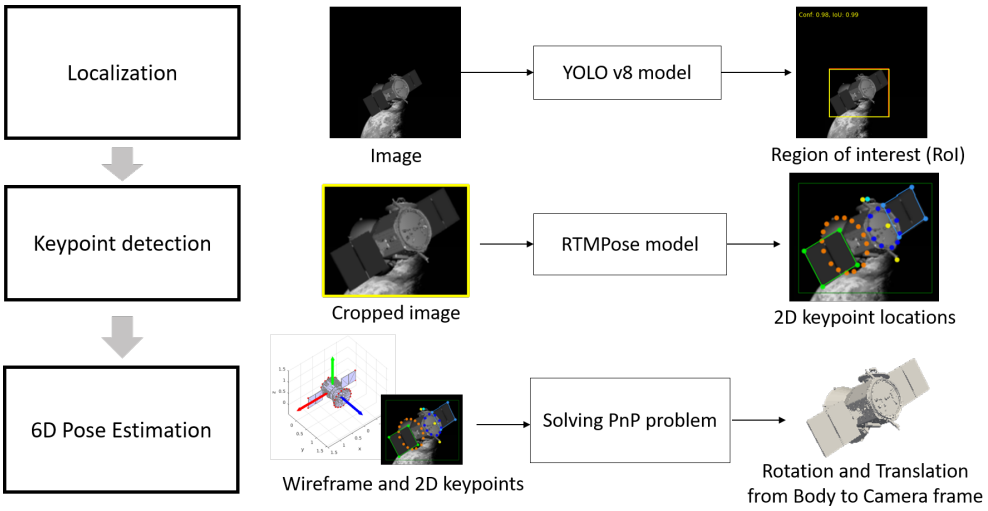

Figure 1: Overview of pose estimation method for a known target.

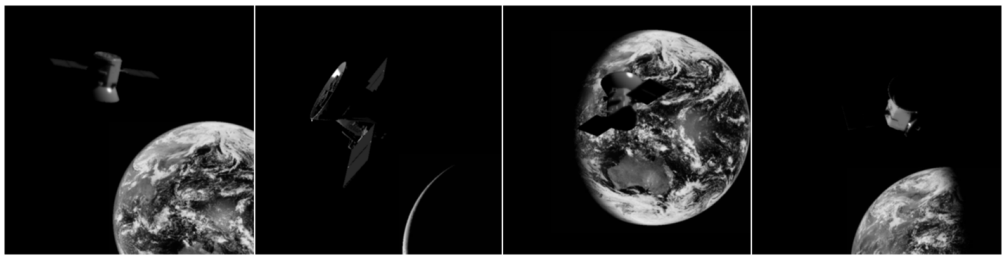

Figure 2: Examples of TESS images used for training YOLOv8 model.

2.3 Object Detection

In each image, a bounding box fully encapsulating all object features in the image is estimated using YOLOv8 [15], a powerful deep neural network model for single-shot object detection. YOLOv8 has an optimal balance between accuracy and speed and is suitable for real-time object detection by a satellite onboard sensor. There are five different sizes of the model, ranging from nano to extra-large model. In this study, the nano model (YOLOv8-n) pre-trained on the COCO dataset is adapted and trained on the TESS training dataset for 200 epochs with a batch size of 16.

The trained model achieves the mAP50 of 0.995 and the mAP50-95 of 0.981 on the validation dataset. The left panel in Figure 3 shows examples of the ground truth (shown in red) and detected bounding boxes (shown in yellow). Even when the spacecraft is overlapped with Earth or hardly visible by shadow, the model successfully detects the bounding boxes. The top-right panel in Figure 3 shows the impact of RMS image contrast and the intersection-over-union (IoU) of the detected bounding boxes. Although there is a slight degradation in the performance for very limited contrast, the model achieves over 0.85 of IoU for most images. The bottom-right panel in Figure 3 shows the distribution of the IoU values on the validation dataset. The result shows that the model gives the IoU of over 0.8 for 99.9% of images in the dataset. All of these results show that the YOLOv8-n performs sufficiently well for detecting the TESS, even with the smallest number of hyperparameters.

The predicted bounding boxes are enlarged by adding a 10-pixel margin to the width and height to ensure that the boxes completely encapsulate the object. The images are cropped by the bounding boxes, and keypoints are detected from the cropped images in the next step.

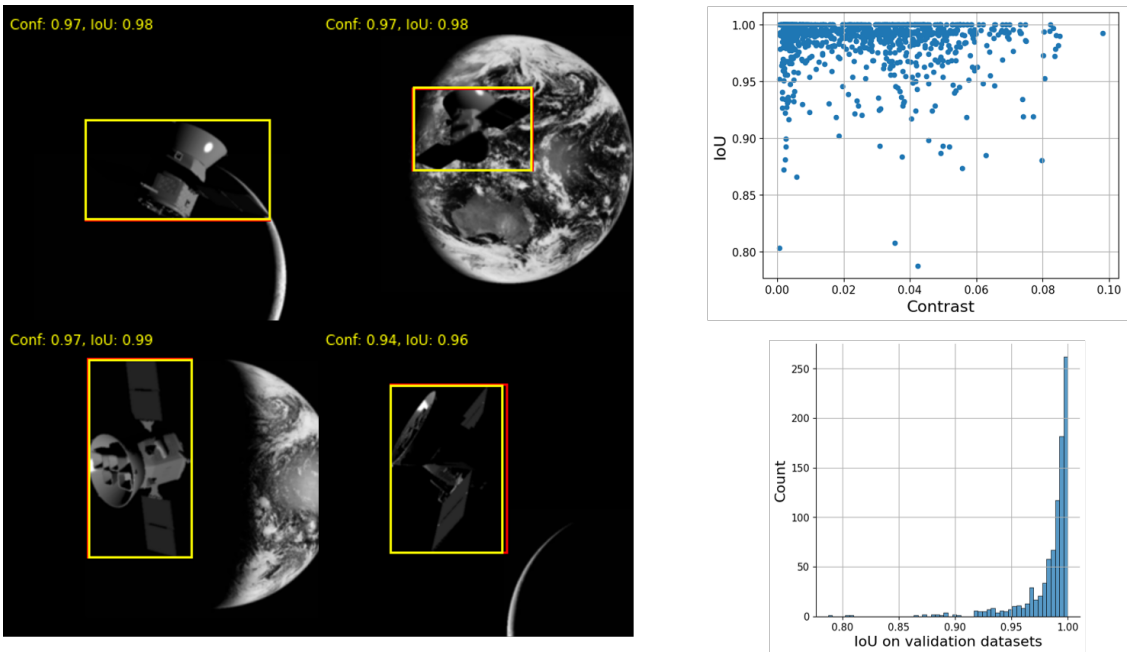

Figure 3: Results of object detection by YOLOv8 model. Left panel: truth (red) and prediction (yellow) of bounding boxes. Top-right panel: the relationship between IoU and contrast of 900 validation images. Bottom-right panel: distribution of IoU values of 900 results.

2.4 Keypoint Detection

This study uses a deep neural network model to detect the keypoints of TESS in the cropped images. The keypoint locations in the body-fixed frame are predefined as shown in Figure 4, comprising the a priori perfect knowledge of the target spacecraft. There are 38 keypoints in total: 16 points on TEES' aperture, 10 points on the bus, 8 points on the solar panels, 1 point on the parabolic antenna, and 3 points on the rod antennas. The keypoint locations in the body-fixed frame are assumed to be perfectly known.

Several deep neural network models are tested and compared. Two families of network models are commonly used for detecting keypoints: regression models and heatmap models. Regression models directly learn regression from the image to the keypoint locations using a relatively simple network architecture. This low complexity enables the regression model to train and infer in a very short time. In contrast, heatmap models learn a heatmap of semantic features of images, followed by an extraction of keypoints from the heatmap. Most heatmap models employ multiple parallel convolution layers to extract multi-resolution features, achieving better accuracy than the regression model. The accuracy of the keypoint detection highly depends on the resolution of the heatmap, but the more fine-grained heatmap leads to a longer inference time. This problem is tackled by some recent models. One such model is RTMPose [14], which achieves the sub-pixel localization of keypoints by applying the SimCC-based classification method [19] to the heatmap model.

In this study, four keypoint detection models are tested. The first model is ResNet [11], which is one of the most popular regression models in computer vision. The second and third models are HRNet [12] and HRFormer [13], which are widely-used heatmap models. The fourth model is RTMPose [14], which is expected to outperform the heatmap models. All the models are trained for 210 epochs with a batch size of 64 images and optimized with Adam. The learning rate is linearly increased from 5e-7 to 5e-4 for the first 500 iterations, and it is decayed by 0.1 once the epoch reaches 170 or 200. The training is implemented on NVIDIA A10 Tensor Core GPU.

The trained models are tested on the validation dataset of 900 images and evaluated based on the RMS error of the

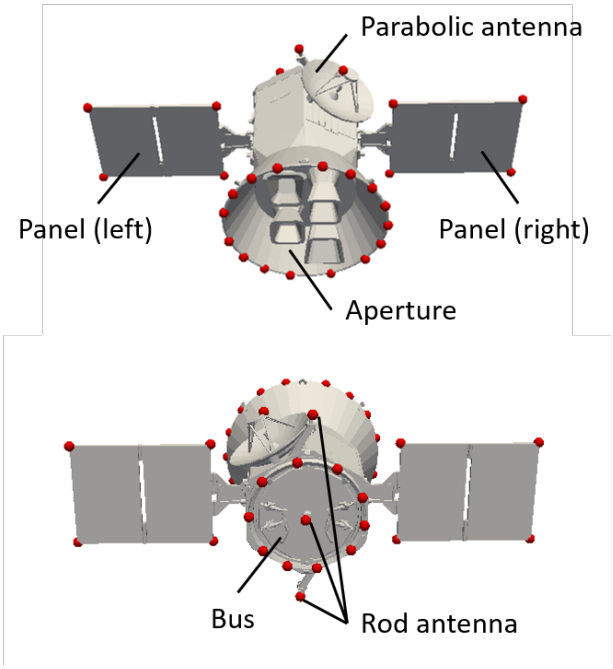

Figure 4: Keypoint locations and name of each component of TESS model.

keypoints, defined by:

$$E_{\text{keypoint}} = \frac{1}{mn} \sum_{j=1}^{m} \sqrt{\sum_{i=1}^{n} \frac{d_{i,j}^2}{s_j^2}} \times 100\ \% \tag{1}$$

where m is the number of images in the validation dataset, and n is the number of keypoints of objects in the image. The value $d_{i,j}$ is the Euclidean distance between prediction and ground truth of i-th keypoint locations in the j-th image. The value s_j is the object scale, given by a square root of the area of the bounding box of the object in the j-th image.

The mean detection errors of keypoints detected by the four models are illustrated in Figure 5. The keypoint errors are averaged in several groups, each identified by a component name defined in Figure 4. Detecting keypoints on solar panels is challenging for all the methods, especially for the ResNet, which gives more than 8% of the detection error on average on solar panels. However, the error of the detected keypoints by RTMPose is only around 1%, even on the solar panels. The RTMPose is also favored in terms of the training and inference speed. The training time of the RTMPose is 0.245 minutes per epoch, while that of the ResNet, HRNet, and HRFormer are 0.222, 0.778, and 0.890 minutes per epoch, respectively.

Figure 6 shows several images with keypoints detected by RTMPose. Each panel shows an image with true keypoints on the left and the one with detected keypoints on the right. The keypoint errors on frames 4443, 4730, and 4708 are 57.0, 42.3, and 35.1%, and these are the worst three results over 900 images in the validation dataset. The first three frames have particularly large errors because the keypoints on one solar panel are mistaken for the ones on the other. The flipping of the two solar panels occurs because the important cues for distinguishing these two panels, such as the keypoints on the parabolic antenna and rod antenna, are almost completely hidden in those three images. However, except for the three images where the occlusion of those keypoints occurs, the keypoint detection errors are below 4% for the validation dataset.

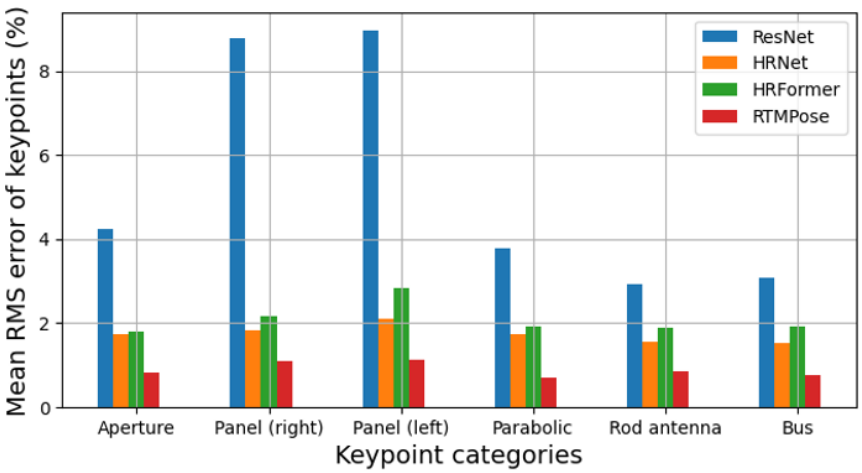

Figure 5: Comparison of four keypoint detection network models: ResNet, HRNet, HRFormer, and RTMPose in terms of category-wise mean RMS error.

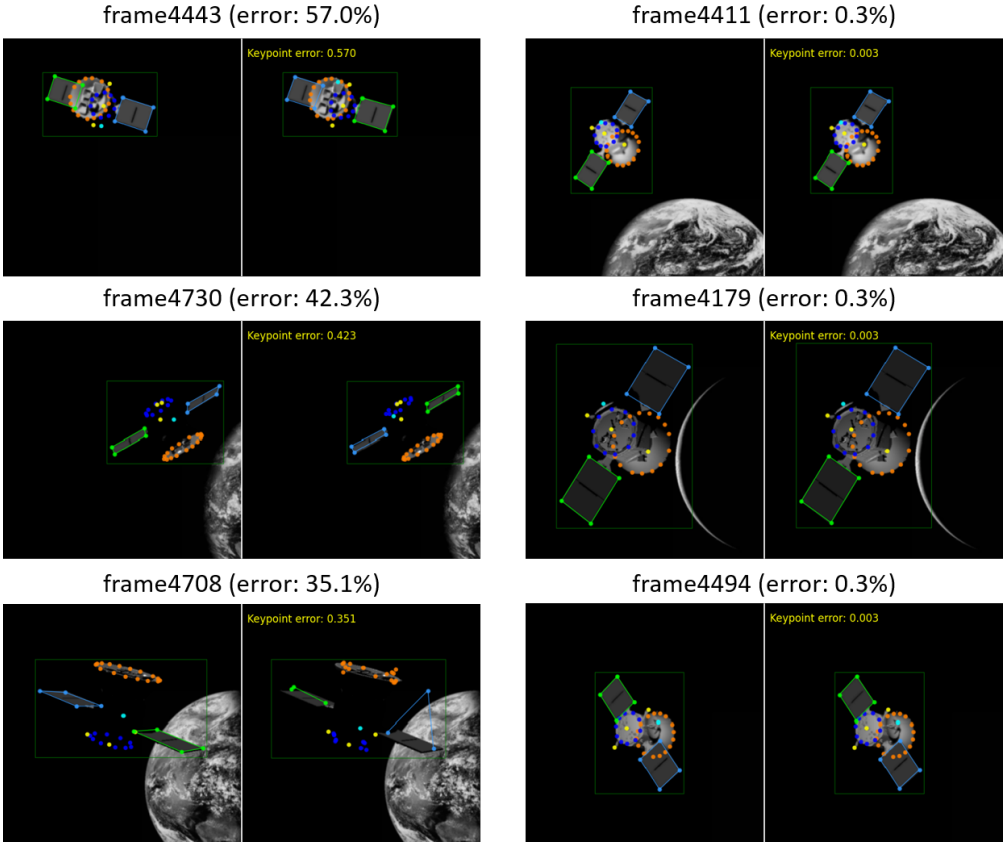

Figure 6: Results of the keypoint detection by RTMPose model. The first column shows the worst three results, while the second column shows the best three results in 900 validation images. Each panel shows the true keypoints on the left and the predicted keypoints on the right. Frame 4443, 4730, and 4708 have errors over 30% because keypoint locations on the solar panels are flipped. The flip occurs only in the three images, and all the other results achieve much smaller keypoint errors (<4%).

2.5 Pose Estimation

Once the keypoints of the TESS are detected in an image, as shown in Figure 6, they can be used to estimate the pose of the satellite or its position and orientation in space. The estimate is generated by solving the Perspective-n-Point (PnP) problem. The goal of the PnP problem is to find the position and orientation of a camera—or, equivalently, of the points relative to the camera—given the same set of n points both in three-dimensional space and in a two-dimensional image. The camera intrinsics, such as pixel density and focal length, are also assumed to be known.

In this study, the n points correspond to the 38 TESS keypoints. The PnP problem is solved by finding the transformations between a coordinate frame fixed in the satellite and the image plane of the camera. Let B be a body-fixed coordinate system in the TESS satellite as shown in Figure 7 and C be a coordinate system fixed in the camera as in Figure 8. The transformation from body-fixed coordinates in B to camera-fixed coordinates in C is:

$$\boldsymbol{r}^C = \left[\begin{array}{c|c} {}^B\bar{R}^C & {}^C\boldsymbol{t}^B \end{array} \right] \begin{bmatrix} \boldsymbol{r}^B \\ 1 \end{bmatrix}, \tag{2}$$

where $\boldsymbol{r}^{B/C} = \begin{bmatrix} x^{B/C} & y^{B/C} & z^{B/C} \end{bmatrix}^T$ is the vector written in B and C coordinates, respectively. The matrix ${}^B\bar{R}^C$ is the direction cosine matrix (DCM) from the B frame to the C frame, and ${}^C\boldsymbol{t}^B$ is the translation vector from the origin of the camera frame to the origin of the body frame. A vector in the camera frame can be written in terms of homogeneous image coordinates as follows:

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} {}^f\!/_{\rho_u} & 0 & u_0 \\ 0 & {}^f\!/_{\rho_v} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{r}^C = \bar{A}\boldsymbol{r}^C, \tag{3}$$

where $\tilde{u}$, $\tilde{v}$, and $\tilde{w}$ are homogeneous image coordinates, ρ_u and ρ_v are the horizontal and vertical pixel densities, f is the camera focal length, and u_0 and v_0 are the horizontal and vertical coordinates of the image centroid. To get the image coordinates in terms of pixels, $\tilde{u}$ and $\tilde{v}$ must be divided by $\tilde{w}$.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{\tilde{w}} \begin{bmatrix} \tilde{u} \\ \tilde{v} \end{bmatrix} \tag{4}$$

In the PnP problem applied to TESS, the pixel coordinates u_i and v_i and the body-frame coordinates $\boldsymbol{r}_i^B$ of the 38 keypoints are known. Knowing the pose of the satellite is equivalent to knowing both the DCM ${}^B\bar{R}^C$ and translation vector ${}^C\boldsymbol{t}^B$ in Eq. (3). Their values can be found by either solving for them directly or searching for the $\boldsymbol{r}_i^C$ values that produce the observed u_i and v_i. Once the camera-frame and body-frame keypoint positions are known, it is easy to find the transformation between the two coordinate systems.

Two different methods of solving the problem, the efficient PnP (EPnP) algorithm and a particle swarm optimizer (PSO), are discussed below. The EPnP uses a non-iterative method to find the camera-frame positions of the TESS keypoints. In contrast, the PSO uses an iterative method to directly search for the DCM and translation vector that best fits the input image. In the following two sections, the EPnP algorithm and PSO are outlined. Then, both methods are applied to the 900-image validation dataset, and their relative performances are discussed. Although the validation dataset is reused to test the two pose estimation methods, neither is a machine learning method.

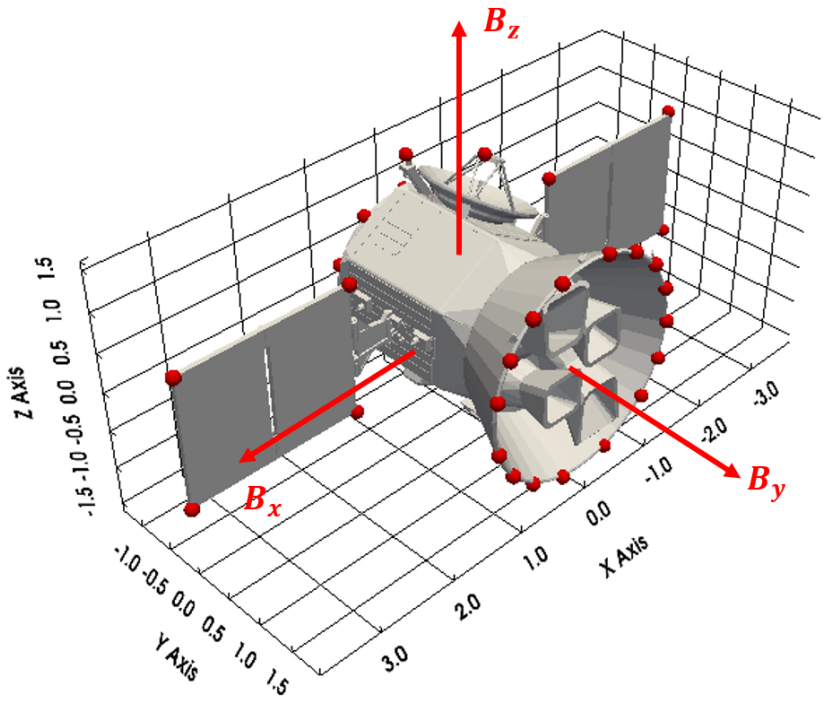

Figure 7: Illustrates the body-fixed *B* frame unit vector directions on the TESS model.

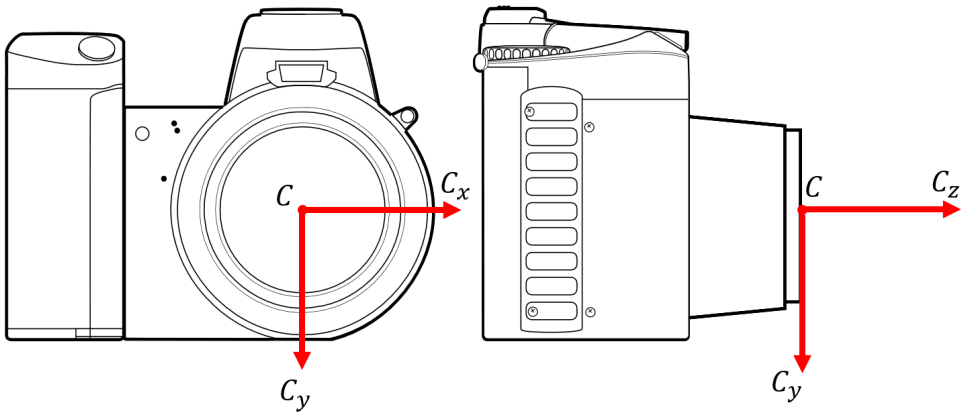

Figure 8: Illustrates the camera-fixed *C* frame unit vector directions using a generic camera image [20].

2.5.1 Efficient PnP (EPnP) Algorithm

The EPnP algorithm introduced by Lepetit et al. [10] is a non-iterative method of solving the PnP problem using an algebraic solution. The outline of the algorithm in this section closely follows the original paper with different notation in some places. The algorithm reduces the PnP problem from one of estimating the camera frame position of n keypoints to one of estimating the position of only four control points $\boldsymbol{c}_j^C = \begin{bmatrix} a_j^C & b_j^C & c_j^C \end{bmatrix}^T$. For TESS $n = 38$. The control points are defined such that:

$$\boldsymbol{r}_i^B = \sum_{j=1}^{4} \alpha_{ij} \boldsymbol{c}_j^B \qquad\qquad \boldsymbol{r}_i^C = \sum_{j=1}^{4} \alpha_{ij} \boldsymbol{c}_j^C, \tag{5}$$

where i indicates the i-th of n keypoints, and α_{ij} is a known set of weights that satisfy the following condition.

$$\sum_{j=1}^{4} \alpha_{ij} = 1 \tag{6}$$

After some manipulation, it can be shown that substituting Eq. (5) into Eq. (3) results in the following pair of linear equations for each keypoint.

$$\sum_{j=1}^{4} \frac{f}{\rho_u} \alpha_{ij} a_j^C - \alpha_{ij}(u_i - u_0) c_j^C = 0 \tag{7}$$

$$\sum_{j=1}^{4} \frac{f}{\rho_v} \alpha_{ij} b_j^C - \alpha_{ij}(v_i - v_0) c_j^C = 0 \tag{8}$$

These summations result in a system of $2n$ linear equations once all keypoints are taken into account. This system can be written as:

$$\bar{M}\boldsymbol{x} = \boldsymbol{0} \tag{9}$$

where

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{c}_1^C \\ \boldsymbol{c}_2^C \\ \boldsymbol{c}_3^C \\ \boldsymbol{c}_4^C \end{bmatrix} \tag{10}$$

The value of $\boldsymbol{x}$ is found by solving for the null space of $\bar{M}$, and the control points are then used to find the DCM and translation vector from the body-fixed frame to the camera-fixed frame.

The EPnP algorithm has been widely used for solving PnP problems because of its fast computation with almost no loss of accuracy compared to iterative methods. The most expensive part of the algorithm is the computation of $\bar{M}^T\bar{M}$ to estimate the null space of $\bar{M}$, which can be done in $O(n)$ complexity for $n \geq 4$. However, the estimation of the null space is very sensitive to outliers [21], and it may require additional steps to reject outliers where the 3D-2D correspondences of keypoints are corrupted by noise [16].

2.5.2 Particle Swarm Optimization

A particle swarm optimizer (PSO) searches the solution space of a problem using a method based on the swarming behavior of social animals, such as flocks of birds. It was first introduced by Eberhart and Kennedy [22] in 1995, and

its properties have continued to be studied in the decades since [23, 24]. As a proof of concept, the implementation in this study largely follows Eberhart and Kennedy's original paper.

A PSO works by generating particles in the problem's solution space. For the pose estimation problem, the solution space is the DCM ${}^{B}\bar{R}^{C}$ and translation vector ${}^{C}\boldsymbol{t}^{B}$ from Eq. (2). The position of a particle i in solution space is described using the vector $\hat{\boldsymbol{y}}_i$:

$$\hat{\boldsymbol{y}}_i = \begin{bmatrix} \hat{\boldsymbol{y}}_{t,i} \\ \hat{\boldsymbol{y}}_{R,i} \end{bmatrix} \tag{11}$$

where $\hat{\boldsymbol{y}}_t$ is the translation vector estimate and $\hat{\boldsymbol{y}}_R$ is the attitude estimate.

$$\hat{\boldsymbol{y}}_{t,i} = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \qquad\qquad \hat{\boldsymbol{y}}_{R,i} = \begin{bmatrix} \phi_i \\ \theta_i \\ \psi_i \end{bmatrix} \tag{12}$$

where ϕ_i, θ_i, and ψ_i are the Euler angles in a 321 Euler angle sequence corresponding to the estimated DCM. An Euler angle representation of the attitude space is used because it allows each particle to move through attitude space as if it were a Cartesian space while guaranteeing that the resulting DCM is a valid rotation matrix.

This study uses the mean position of the detected keypoints in the image to provide an initial guess of the unit vector from the camera to TESS. The $\hat{\boldsymbol{y}}_{t,i}$ vector is generated by multiplying the unit vector by a random range and then perturbing the result randomly by a small amount. To create the validation images, the TESS was simulated as between 18 and 40 meters from the camera, so the randomly generated ranges were set to be between 10 and 50 meters.

Each particle has a velocity vector in solution space $\boldsymbol{v}_i$:

$$\boldsymbol{v}_i = \begin{bmatrix} \boldsymbol{v}_{t,i} \\ \boldsymbol{v}_{R,i} \end{bmatrix} \tag{13}$$

where $\boldsymbol{v}_{t,i}$ is the velocity associated with $\hat{\boldsymbol{y}}_{t,i}$ and $\boldsymbol{v}_{R,i}$ is the velocity associated with $\hat{\boldsymbol{y}}_{R,i}$. Each velocity is constrained not to exceed a certain maximum $\boldsymbol{v}_{max,t}$ or $\boldsymbol{v}_{max,R}$.

Each particle is randomly assigned a position and velocity when the optimizer is initialized. After being created, each particle evaluates the cost function to be minimized at its location. For this study, the cost function is based on the location of the estimate's projected keypoints in the image of TESS:

$$J = \frac{1}{n}\sum_{j=1}^{n} d_j, \tag{14}$$

where $n = 38$ is the number of keypoints, and d_j is the Euclidean distance between the detected pixel coordinates of the j-th keypoint and its estimated location based on $\hat{\boldsymbol{y}}_i$. The formulation of Eq. (14) encourages the PSO to find a pose estimate that minimizes the average d_j value, which encourages solutions that minimize the RMS error of the estimate's projected keypoint locations.

The value of J at each particle's initial location is stored as its personal best $\hat{\boldsymbol{y}}_{b,i}$ with the associated cost $p_{b,i}$. The lowest $p_{b,i}$ with its estimate is stored as the global best p_g and $\hat{\boldsymbol{y}}_g$. Particles are also assigned neighbors based on their starting $\hat{\boldsymbol{y}}_R$ vectors. For each particle i, the n_{neigh} other particles with $\hat{\boldsymbol{y}}_R$ closest to $\hat{\boldsymbol{y}}_{R,i}$ are chosen as neighbors. The p_b of a particle and its neighbors is stored as its local best $p_{l,i}$ corresponding to $\hat{\boldsymbol{y}}_l$.

Next, each particle is accelerated randomly towards the personal, local, and global bests according to the following expression:

$$\boldsymbol{v}_i^{+} = \begin{bmatrix} \boldsymbol{v}_{t,i}^{+} \\ \boldsymbol{v}_{R,i} \end{bmatrix} = \boldsymbol{v}_i^{-} + \alpha\left[R_b(\hat{\boldsymbol{y}}_{b,i} - \hat{\boldsymbol{y}}_i) + w_l R_l(\hat{\boldsymbol{y}}_{l,i} - \hat{\boldsymbol{y}}_i) + w_g R_g(\hat{\boldsymbol{y}}_g - \hat{\boldsymbol{y}}_i)\right] \tag{15}$$

where α is an acceleration constant, $\boldsymbol{v}_i^{-/+}$ is the velocity of particle i before and after acceleration, the R_x terms are random numbers between 0 and 1. The w_x terms are relative weights given to the local and global bests. If the magnitude of either $\boldsymbol{v}_{t,i}^{+}$ or $\boldsymbol{v}_{R,i}^{+}$ is greater than the maximum allowed magnitudes, then its magnitude is reduced to the maximum.

Finally, the positions of the particles are updated.

$$\hat{\boldsymbol{y}}_i^{+} = \begin{bmatrix} \hat{\boldsymbol{y}}_{t,i}^{+} \\ \hat{\boldsymbol{y}}_{R,i}^{+} \end{bmatrix} = \hat{\boldsymbol{y}}_i^{-} + \boldsymbol{v}_i \tag{16}$$

Note that if any of the components of $\hat{\boldsymbol{y}}_{R,i}^{+}$ are outside of the $[0, 2\pi]$ range, $\pm 2\pi$ is added to the component to place it inside the desired range.

The cost function Eq. (14) is reevaluated at the new $\hat{\boldsymbol{y}}_i^{+}$ locations. If the cost associated with any particle's new location is less than its current $p_{b,i}$, then the $p_{b,i}$ and $\hat{\boldsymbol{y}}_{b,i}$ are updated accordingly. Similarly, the local and global bests are updated if necessary. The velocity of the particles is updated again using Eq. (15), and the process repeats for a set number of iterations.

For this study, 250 particles are initialized, each particle is assigned $n_{neigh} = 5$ neighbors, and the PSO is run for 250 total iterations.

2.6 Results

The EPnP algorithm and PSO are used to estimate TESS's pose using the images in the validation dataset. For these tests, the positions of the keypoints in each image are predicted using the keypoint detection neural network. Due to symmetries in TESS's shape, the keypoint predictions are incorrectly placed in three of the 900 images in the validation dataset, as discussed in Section 2.4. In those three cases, the left solar panel's keypoints are placed over the right solar panel and vice versa in the image. Pose estimates made using these three images include anomalously large errors, so they are excluded from the dataset when evaluating the accuracy of the EPnP and PSO estimates.

The pose estimates are evaluated using three parameters: normalized translation error, relative angle to the true attitude, and RMS error in their projected keypoints. The normalized translation error is calculated by taking the norm of the difference between the estimated translation vector ${}^{C}\hat{\boldsymbol{t}}^{B}$ and the true translation vector ${}^{C}\boldsymbol{t}^{B}$, then dividing by the true distance between the camera and TESS.

$$E_T = \frac{||{}^{C}\hat{\boldsymbol{t}}^{B} - {}^{C}\boldsymbol{t}^{B}||}{||{}^{C}\boldsymbol{t}^{B}||} \tag{17}$$

The normalized position estimates for the 897 images are shown as a histogram in Figure 9. The blue bins correspond to the EPnP estimate errors, and the orange bins correspond to the POS estimate errors. Although the maximum and minimum position estimate errors are approximately the same for both methods, as seen in Table 1, the distribution of the two sets of errors is quite different. Specifically, EPnP errors have a larger standard deviation of $8.726 \cdot 10^{-3}$ compared to the PSO errors' standard deviation of $4.807 \cdot 10^{-3}$. This difference in distribution is also demonstrated by the EPnP algorithm's higher mean error of $1.327 \cdot 10^{-2}$ compared to the PSO's mean error of $5.293 \cdot 10^{-3}$.

In contrast to the position errors, the distribution of relative angle E_R values for the EPnP algorithm and PSO are quite similar, as shown in Figure 10 and Table 1. The relative angle between the estimated attitude and the true attitude $\boldsymbol{q}$ is the minimum rotation angle needed to align the true and estimated attitudes and is calculated as follows:

Table 1: Compares the error statistics for the EPnP and PSO pose estimates.

	Normalized Translation Error		**Relative Angle (deg)**		**RMS (%)**	
	EPnP	PSO	EPnP	PSO	EPnP	PSO
Min	$1.099 \cdot 10^{-4}$	$2.169 \cdot 10^{-4}$	0.03613	0.07150	0.2375	0.2105
Max	$4.706 \cdot 10^{-2}$	$3.637 \cdot 10^{-2}$	13.1872	11.5121	4.5441	2.2680
Mean	$1.327 \cdot 10^{-2}$	$5.293 \cdot 10^{-3}$	1.1600	1.0986	0.8420	0.4379
St. Dev.	$8.726 \cdot 10^{-3}$	$4.807 \cdot 10^{-3}$	1.2072	0.8584	0.4219	0.1527

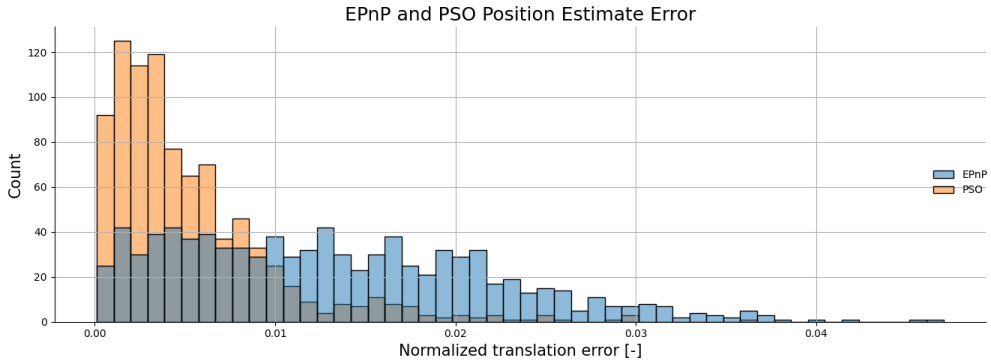

Figure 9: The normalized errors in the position estimates obtained using the EPnP and PSO methods across 897 images.

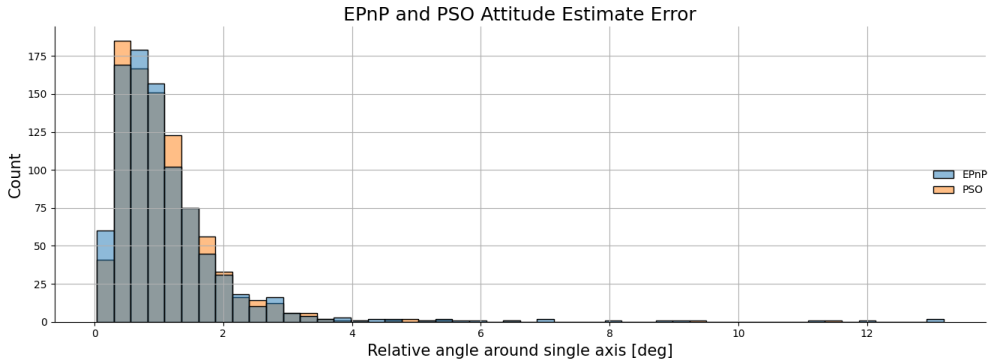

Figure 10: The normalized errors in the attitude estimates obtained using the EPnP and PSO methods across 897 images.

$$E_R = 2\cos^{-1}(\hat{\boldsymbol{q}} \cdot \boldsymbol{q}), \tag{18}$$

where $\boldsymbol{q}$ is the true attitude represented as a quaternion and $\hat{\boldsymbol{q}}$ is the estimated attitude. The relative angle distributions for the EPnP and PSO estimates are similar, although the EPnP E_R values have a higher standard deviation and slightly higher mean than the PSO E_R distribution.

The RMS error E_{RMS} for each pose estimate is calculated as follows:

$$E_{\text{RMS}} = \frac{1}{n}\sqrt{\sum_{j=1}^{n}\frac{d_j^2}{s^2}} \times 100\ \% \tag{19}$$

where $n = 38$ is the number of keypoints, d_j is the Euclidean distance between the projected j-th keypoint based on the pose estimate and the predicted location of the same keypoint, and s is the object scale calculated as for Eq. (1). The resulting E_{RMS} values are shown in Figure 11. As seen from the plot and Table 1, the PSO pose estimates have a lower keypoint mean RMS error and are more tightly distributed than the EPnP RMS errors. This difference is not surprising since the cost PSO function in Eq. (14) penalizes differences between the estimate's projected keypoints and the keypoints detected in the image.

However, the translation, relative angle, and keypoint RMS errors do not tell the whole story. While the PSO estimates have better mean E_T and E_{RMS} values than the estimates obtained using the EPnP algorithm, the EPnP algorithm is still able to locate TESS accurately enough for many purposes and uses far less computing power than the PSO.

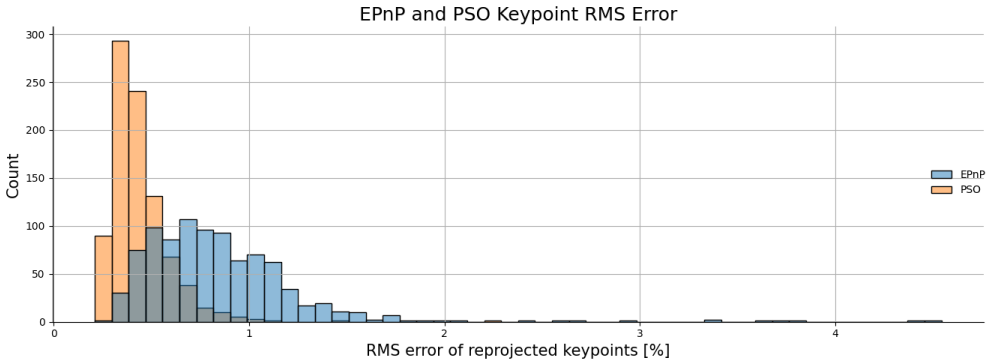

Figure 11: The normalized errors in the projected keypoints for the pose estimates obtained using the EPnP and PSO methods across 897 images.

On average, the EPnP took only 0.0182 seconds to generate a pose estimate for a given set of detected keypoints to the PSO's approximately 5-second computation time. Therefore, the EPnP algorithm is more practical in applications where real-time estimates are needed. Although the PSO is slower than EPnP, its estimates can be improved by running the optimizer with a lower maximum particle velocity over more iterations if more accuracy is required. The PSO is also expected to be more robust to imaging errors than the EPnP algorithm.

3. CONCLUSIONS

This paper explores the monocular monostatic pose estimation method for a space object with the knowledge of 3D geometry from a single near-field resolved optical image.

Machine learning models are used for object detection, object classification, and keypoint detection. The dataset for evaluating the models is generated by a fast in-house photo-realistic renderer modeling a space-based observer camera with a focal length of 21.9 meters. The camera-to-object distances are modeled to be between 18 and 40 meters.

First, the object's bounding box is detected by the YOLOv8n model, and the 2D keypoint locations are detected by the RTMPose model in a real-time manner. The performance of the RTMPose model is shown to outperform other regression-based and heatmap-based keypoint detection models commonly used in pose estimation. Subsequently, the relative position and attitude of the target object are estimated by solving the Perspective-n-Point (PnP) problem using non-AI methods. This study employs a particle swarm optimizer (PSO) to solve the PnP problem. In the investigated examples, the PSO algorithm achieved higher accuracy than the most well-known PnP solver, the Efficient PnP (EPnP) algorithm, in pose estimation from the single monocular monostatic images. Moreover, the PSO is expected to be more robust to imaging errors than the EPnP algorithm. The proposed technique may be applied to proximity operations where the accuracy of the pose estimate is especially important.

4. ACKNOWLEDGEMENTS

This research has been made possible through the generous support of the Boeing Company's Enterprise Technology office via SSOW-BRT-Z0722-5045. The authors also thank Liam Robinson for generating the datasets utilized in this study.

REFERENCES

[1] Shijie Zhang and Xibin Cao. Closed-form solution of monocular vision-based relative pose determination for rvd spacecrafts. *Aircraft Engineering and Aerospace Technology*, 77(3):192–198, 2005.

[2] Antoine Petit, Eric Marchand, and Keyvan Kanani. Vision-based space autonomous rendezvous: A case study. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 619–624. IEEE, 2011.

[3] Simone D'Amico, Mathias Benn, and John L. Jørgensen. Pose estimation of an uncooperative spacecraft from actual space imagery. *International Journal of Space Science and Engineering*, 2(2):171, 2014.
[4] Alessio A Grompone. *Vision-based 3D motion estimation for on-orbit proximity satellite tracking and navigation*. PhD thesis, Monterey, California: Naval Postgraduate School, 2015.
[5] Tae Ha Park, Sumant Sharma, and Simone D'Amico. Towards robust learning-based pose estimation of noncooperative spacecraft. *arXiv*, (arXiv:1909.00392), 2019.
[6] Massimo Piazza. *Deep Learning-Based Monocular Relative Pose Estimation of Uncooperative Spacecraft*. Master's thesis, Politecnico Milano, 2020.
[7] Siddarth Kaki, Jacob Deutsch, Kevin Black, Asher Cura-Portillo, Brandon A. Jones, and Maruthi R. Akella. Real-time image-based relative pose estimation and filtering for spacecraft applications. *Journal of Aerospace Information Systems*, 20(6):290–307, 2023.
[8] Tae Ha Park, Marcus Märtens, Gurvan Lecuyer, Dario Izzo, and Simone D'Amico. SPEED+: Next-generation dataset for spacecraft pose estimation across domain gap. In *2022 IEEE Aerospace Conference (AERO)*, pages 1–15. ISSN: 1095-323X.
[9] Tae Ha Park, Marcus Märtens, Mohsi Jawaid, Zi Wang, Bo Chen, Tat-Jun Chin, Dario Izzo, and Simone D'Amico. Satellite pose estimation competition 2021: Results and analyses. *Acta Astronautica*, 2023.
[10] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. EPnP: An accurate o(n) solution to the PnP problem. *International Journal of Computer Vision*, 81(2):155–166, 2009.
[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE, 2016.
[12] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition. (arXiv:1908.07919), 2020.
[13] Yuhui Yuan, Rao Fu, Lang Huang, Weihong Lin, Chao Zhang, Xilin Chen, and Jingdong Wang. HRFormer: High-resolution transformer for dense prediction. (arXiv:2110.09408), 2021.
[14] Tao Jiang, Peng Lu, Li Zhang, Ningsheng Ma, Rui Han, Chengqi Lyu, Yining Li, and Kai Chen. RTMPose: Real-time multi-person pose estimation based on mmpose. *arXiv*, 2023.
[15] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics YOLOv8, 2023.
[16] Luis Ferraz Colomina, Xavier Binefa, and Francesc Moreno-Noguer. Leveraging feature uncertainty in the PnP problem. In *Proceedings of the BMVC 2014 British Machine Vision Conference*, pages 1–13, 2014.
[17] Keivan G. Stassun, Ryan J. Oelkers, Joshua Pepper, Martin Paegert, Nathan De Lee, Guillermo Torres, David W. Latham, Sté phane Charpinet, Courtney D. Dressing, Daniel Huber, Stephen R. Kane, Sébastien Lépine, Andrew Mann, Philip S. Muirhead, Bárbara Rojas-Ayala, Roberto Silvotti, Scott W. Fleming, Al Levine, and Peter Plavchan. The TESS input catalog and candidate target list. *The Astronomical Journal*, 156(3):102, Aug 2018.
[18] Kotaro Bessho, Kenji Date, Masahiro Hayashi, Akio Ikeda, Takahito Imai, Hidekazu Inoue, Yukihiro Kumagai, Takuya Miyakawa, Hidehiko Murata, Tomoo Ohno, Arata Okuyama, Ryo Oyama, Yukio Sasaki, Yoshio Shimazu, Kazuki Shimoji, Yasuhiko Sumida, Masuo Suzuki, Hidetaka Taniguchi, Hiroaki Tsuchiyama, Daisaku Uesawa, Hironobu Yokota, and Ryo Yoshida. An Introduction to Himawari-8/9— Japan's New-Generation Geostationary Meteorological Satellites. *Journal of the Meteorological Society of Japan. Ser. II*, 94(2):151–183, 2016.
[19] Yanjie Li, Sen Yang, Peidong Liu, Shoukui Zhang, Yunxiao Wang, Zhicheng Wang, Wankou Yang, and Shu-Tao Xia. SimCC: a simple coordinate classification perspective for human pose estimation. *arXiv*, (arXiv:2107.03332).
[20] Orthographic vector drawing of camera. `https://publicdomainvectors.org/en/free-clipart/Orthographic-vector-drawing-of-camera/20378.html`. Accessed 07/25/2023.
[21] Simon Hadfield, Karel Lebeda, and Richard Bowden. HARD-PnP: PnP optimization using a hybrid approximate representation. *IEEE transactions on pattern analysis and machine intelligence*, 41(3):768–774, 2018.
[22] Russell Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the sixth international symposium on micro machine and human science*, pages 39–43. Ieee, 1995.
[23] Yuhui Shi. Particle swarm optimization. *IEEE connections*, 2(1):8–13, 2004.
[24] Dongshu Wang, Dapei Tan, and Lei Liu. Particle swarm optimization algorithm: an overview. *Soft computing*, 22:387–408, 2018.