End-to-end Neural Estimation of Spacecraft Pose with Intermediate Detection of Keypoints

Antoine Legrand[1,2,3], Renaud Detry[2], and Christophe De Vleeschouwer[1]

[1] Université Catholique de Louvain, Louvain-la-Neuve, Belgium
{antoine.legrand, christophe.devleeschouwer}@uclouvain.be
[2] Katholieke Universiteit Leuven, Leuven, Belgium
renaud.detry@kuleuven.be
[3] Aerospacelab, Mont-Saint-Guibert, Belgium

Abstract. State-of-the-art methods for estimating the pose of spacecrafts in Earth-orbit images rely on a convolutional neural network either to directly regress the spacecraft's 6D pose parameters, or to localize pre-defined keypoints that are then used to compute pose through a Perspective-n-Point solver. We study an alternative solution that uses a convolutional network to predict keypoint locations, which are in turn used by a second network to infer the spacecraft's 6D pose. This formulation retains the performance advantages of keypoint-based methods, while affording end-to-end training and faster processing. Our paper is the first to evaluate the applicability of such a method to the space domain. On the SPEED dataset, our approach achieves a mean rotation error of 4.69° and a mean translation error of 1.59% with a throughput of 31 fps. We show that computational complexity can be reduced at the cost of a minor loss in accuracy.

Keywords: Spacecraft Pose Estimation · End-to-end · Keypoints

1 Introduction

Space agencies and the private sector are showing a growing interest and demand for missions that involve proximity operations, such as on-orbit servicing (repairing, refuelling, inspection) or space debris mitigation. While some of those missions conduct proximity operations via tele-operation, the consensus is that autonomous operations are safer and cheaper, which in turn motivates the development of guidance systems that allow a spacecraft to navigate by itself at close range of its target. A key component of this capability is to estimate on-board the 6D pose, i.e. position and orientation, of the target spacecraft.

In terrestrial applications, 6D pose estimation is often achieved with the help of a Lidar or depth camera. Unfortunately, the cost, mass and power requirements of space-grade Lidars are obstacles to their integration in an orbital probe. As a result, there is a strong interest in solutions that rely solely on cameras, and in particular on a single monocular camera.

To deploy monocular pose estimation on orbit, multiple challenges must be overcome. Orbital lighting conditions are very different from those encountered

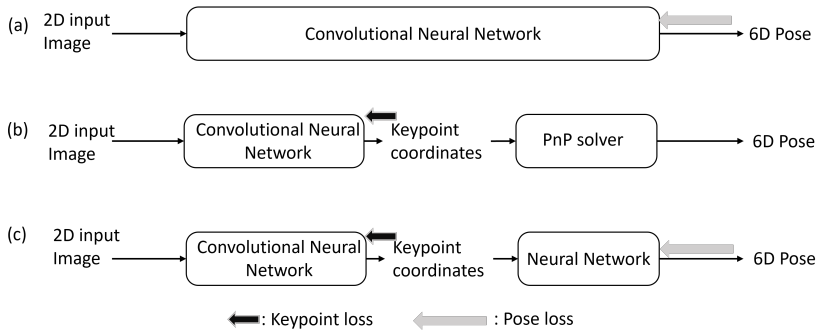

Fig. 1. Overview of the existing methods to estimate a spacecraft's pose. **(a)** Direct methods use a convolutional neural network to directly predict the target 6D pose. **(b)** Keypoints-based methods exploit a convolutional neural network to predict pre-defined keypoint coordinates which are then used to recover the pose through a Perspective-n-Point (PnP) solver. **(c)** Our solution combines both methods as it relies on a first, convolutional, neural network to extract the coordinates of pre-defined keypoints and a neural network to predict pose from those coordinates. The solution is therefore both keypoints-based and end-to-end trainable.

on Earth: the lack of atmospheric diffusion causes extreme contrast between exposed and shadowed surfaces. This problem is further exacerbated by specularity: the absence of dust contamination maintains the specularity of clean metal surfaces. In addition, spacecraft design often includes near symmetrical shapes, which requires us to resolve symmetry ambiguities from relatively subtle features. Finally, pose estimation must be carried out using the limited resources offered by space-grade hardware [5,8,11].

Traditional pose estimators have struggled to overcome the challenges listed above [30]. Recently, as for terrestrial applications, significant improvements have been brought by the use of deep convolutional neural networks [28]. In space robotics, CNN pose estimators generally adopt one of the two following strategies. The first strategy, depicted in Figure 1(a), trains a model that directly regresses the 6D pose of the target from a camera image [22,28]. The mapping from image data to 6D poses is subject to no constraints other than the model's loss function, and the inductive bias of classical CNN layers. By contrast, the second type of model, depicted in Figure 1(b) uses CNNs to predict image keypoints, often set in pre-defined locations on the object [3,16,21,9,2], and adopts an iterative solver such as RANSAC-PnP to turn the keypoints into pose parameters. Despite its attractive end-to-end form, the direct-regression approach (a) generally under-performs compared to the second, keypoint-based, solution (b) [17].

In this paper, we consider a third strategy, depicted in Figure 1(c). Here, a neural network replaces the iterative solver. This network estimates the target's pose from the keypoints predicted by the upstream CNN. This approach enables the end-to-end training of both networks from a loss of direct interest

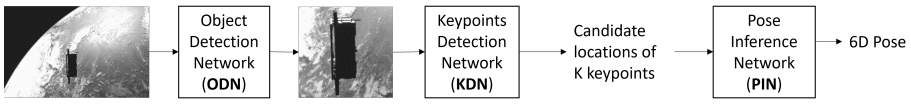

Fig. 2. Framework overview. The high-resolution input image is processed by an Object Detection Network to detect a Region-Of-Interest. The region is fed into a Keypoint Detection Network that regresses multiple candidate locations of K pre-defined keypoints. Finally, a Pose Inference Network computes the spacecraft 6D pose from those candidate locations.

for the spacecraft pose estimation task instead of a surrogate loss. Furthermore, as the iterative solving required by RANSAC-PnP is replaced by an inference through a network that can be parallelized, the proposed method is faster than its PnP-based counterpart. A similar strategy has been tested successfully on terrestrial images of natural objects in [14]. Our work is the first to explore its applicability to images of satellites and spacecrafts orbiting the Earth. We study the relationship between model complexity and pose accuracy, and we validate our work on the standard SPEED [29] dataset.

The rest of the paper is organized as follows. Section 2 presents our method. Section 3 introduces our validation set-up, and presents the results of our experiments. Section 4 concludes.

2 Method

Figure 2 depicts our spacecraft pose estimation framework. The input image goes through an Object Detection Network (ODN) that outputs a square region forming a minimal bounding box around the object. Then, this region is processed by a Keypoint Detection Network (KDN) that predicts multiple candidate pixel coordinates for K pre-defined keypoints. Finally, we map keypoint coordinates to the spacecraft's 6D pose with a neural network that we refer to as the Pose Inference Network (PIN).

This section discusses the KDN (Section 2.1) and PIN (Section 2.2), as well as the approach we followed to train these two networks jointly (Section 2.3). We assume that object detection is carried out with one of the many solutions previously discussed in the literature [3,21,1,9,23] and discuss it no further. In Section 3, we limit the scope of our evaluation to the KDN and PIN, and use ground truth bounding boxes.

2.1 Keypoint Detection Network

The Keypoint Detection Network used in this paper was originally proposed by Hu *et al.* [15] to predict the locations of K pre-defined keypoints in natural and terrestrial images. As depicted in Figure 3, this network consists of a backbone and two heads. The first head performs a segmentation task that separates the spacecraft from the background (empty space or Earth's disk) while the second

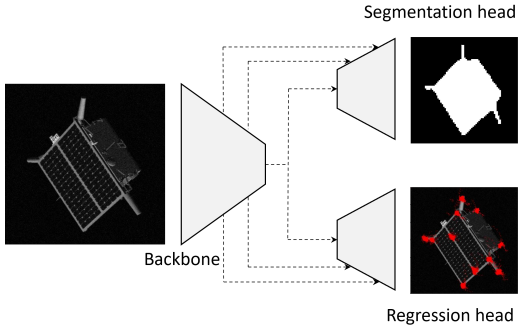

Fig. 3. Architecture of the Keypoint Detection Network [15]. It is made of a backbone and two decoding heads. The first head performs a segmentation of the spacecraft while the second head regresses candidate locations of some pre-defined keypoints. Dashed lines represent skip connections [25].

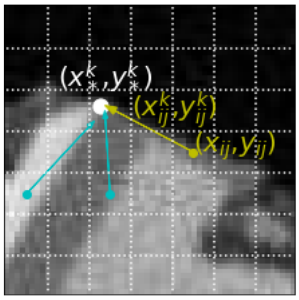

Fig. 4. Illustration of some offsets predicted by the KDN. (x^k_*,y^k_*) are the ground-truth coordinates of the keypoint k. (x_{ij},y_{ij}) are the coordinates of the center of a patch. This patch predicts an offset (x^k_{ij}, y^k_{ij}) pointing toward keypoint k.

head predicts one composite field, f, for each 3D keypoint. Each cell of these composite fields encodes a 2D vector that points to the location of the associated keypoint, and a parameter that models the confidence associated to this offset.

The width/height of the maps produced by either head of the network are an eighth of the width/height of the input image. The segmentation stream outputs a map that predicts, for each cell, whether the corresponding 8×8 patch in the input image belongs to the foreground or the background. The fields $f \in R^{K\times 3\times W\times H}$ regressed by the second head can be represented over the output map as:

$$f^k_{ij} = [x^k_{ij}, y^k_{ij}, c^k_{ij}] \tag{1}$$

where i and j index the output feature cell over the feature map height H and width W respectively, and $1 \leq k \leq K$ denotes the keypoint index. As depicted in Figure 4, x and y denotes the coordinates of the vector pointing from the

center of the cell to the keypoint location while c is the confidence of cell in its prediction.

In summary, the KDN fulfills three tasks: (a) Segmentation of the spacecraft, (b) Offsets prediction on predefined keypoint locations, (c) Confidence estimation on the accuracy of these predictions. Following Hu *et al.* [15], we model these three tasks with three loss functions respectively denoted by $\mathcal{L}_{seg}$, $\mathcal{L}_{kpts}$ and $\mathcal{L}_{conf}$, and we train the KDN with a loss written as

$$\mathcal{L}_{KDN} = \beta_{seg}\mathcal{L}_{seg} + \beta_{kpts}\mathcal{L}_{kpts} + \beta_{conf}\mathcal{L}_{conf}, \tag{2}$$

where β_{seg}, β_{kpts} and β_{conf} are numerical parameters that weight the contribution of each task in the KDN.

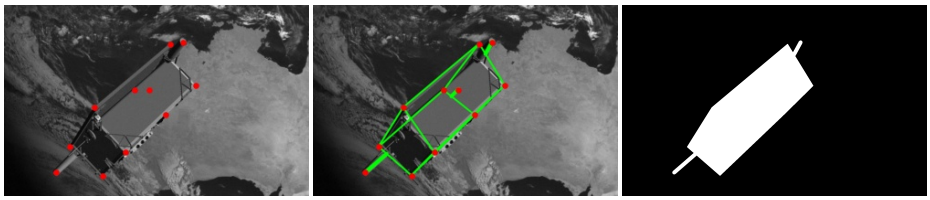

Fig. 5. Computation of pseudo ground-truth segmentation masks. **left**: Keypoint coordinates may be computed by projecting the spacecraft model on the image plane according to its relative pose. **middle**: Body Edges are directly computed from the body keypoints while antennas may be approximated as a straight line segment between their corresponding keypoint and their basis on the spacecraft body. **right**: The approximate segmentation mask can be recovered by filling in the body parts.

We define the segmentation loss $\mathcal{L}_{seg}$ as the Binary Cross Entropy between the predicted segmentation map and the ground-truth segmentation mask [15]. As SPEED [29] does not contain segmentation masks, we define approximate segmentation masks from the known keypoint coordinates as depicted in Figure 5.

The keypoint loss $\mathcal{L}_{kpts}$ is the term associated to the mean error between the predicted keypoint coordinates and their ground truth, computed only in offset map locations that belong to the foreground, $\mathcal{M}$. Let Δ_{ij}^{k} denote the prediction error of the k^{th} keypoint from cell (i, j)'s location, with

$$\Delta_{ij}^{k} = \sqrt{(x_{ij} + x_{ij}^{k} - x_{*}^{k})^2 + (y_{ij} + y_{i}^{k} - y_{*}^{k})^2}, \tag{3}$$

where $(x_{ij},\ y_{ij})$ are coordinates of the center of the patch of the input image that corresponds to cell (i, j) in the output map, and (x_{*}^{k}, y_{*}^{k}) are the coordinates of keypoint k in the input image. We define the keypoint loss as the sum of all prediction errors [15], with

$$\mathcal{L}_{kpts} = \frac{1}{K}\frac{1}{|\mathcal{M}|}\sum_{(i,j)\in\mathcal{M}}\sum_{k=1}^{K}\Delta_{ij}^{k}. \tag{4}$$

Finally, we design the confidence loss $\mathcal{L}_{conf}$ to control the training of the confidence map prediction. As in previous work [15], this is done by adopting a MSE loss to ensure that the confidence map fits a decreasing exponential of the offset approximation error. Formally,

$$\mathcal{L}_{conf} = \frac{1}{K}\frac{1}{|\mathcal{M}|}\sum_{(i,j)\in\mathcal{M}}\sum_{k=1}^{K}(c_{ij}^{k} - e^{-\tau\Delta_{i,j}^{k}})^2. \tag{5}$$

This network was evaluated on SPEED [10], in conjunction with a RANSAC-PnP pose solver, during SPEC2019 [17] where it achieved the second best performance. Unlike the SPEC2019 winner [3], the network is inherently resilient to occlusions because it can predict the location of keypoints hidden or located outside the image, via the predictions made by patches that are in the frame. This relaxes significantly the constrains on the rest of the system and therefore motivates its selection as a baseline for this paper.

2.2 Pose Inference Network

This section presents how the 2D keypoint location candidates are turned into a 3D pose, using a neural network trained to convert 2D cues into 3D pose. This gives the opportunity to make the pose prediction system end-to-end, but requires a specific training for each target satellite and each camera intrinsic parameters.

The Keypoint Detection Network computes $W \times H$ predictions of the location of each keypoint. Among those predictions, those made by background cells are highly uncertain. Hence, the second network only processes the predictions made by foreground cells. Furthermore, since the input resolution of the second network is fixed, we sample m predictions per keypoint out of the foreground predictions.

Because each prediction associates a location on the 2D image with a 3D reference keypoint, those predictions are referred as 2D-3D correspondences. The 2D locations are predicted by the Keypoint Detection Network while the 3D keypoints are learned by the Pose Inference Network and implicitly encoded in its weights. In the text below, we refer to the set of correspondences associated to a single keypoint as a *cluster*. This term alludes to the typical clustered nature of predictions around the keypoint's ground truth position.

To infer the spacecraft pose from those clusters, we use the network proposed by Hu *et al.* [14], as depicted in Figure 6, where each prediction, under the form of a 4 dimensional vector containing the patch center coordinates and the predicted offsets ($[x_{ij}, y_{ij}, \hat{x}_{ij}^{k}, \hat{y}_{ij}^{k}]$), is supplied as input to a MLP unit, in charge of extracting a high-dimensional representation of the prediction. The representations corresponding to a same keypoint are then pooled to obtain a representation of the whole cluster. The spacecraft pose is computed through a MLP fed by the representations of all clusters.

In order to train the network, we make use of the 3D reconstruction loss [32,18,14] that captures the 3D-error made on all the keypoints between the estimated

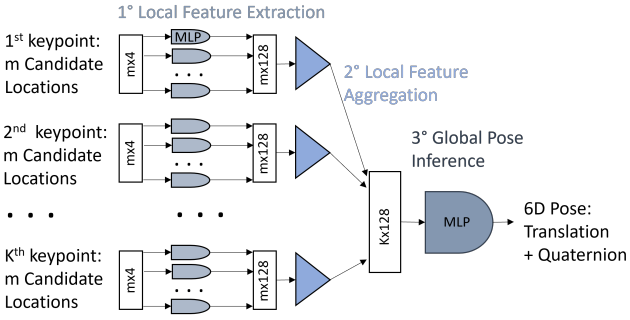

Fig. 6. Architecture of the Pose Inference Network [14]. Local features are extracted from every candidate location through a MLP which weights are shared across all candidates and all keypoints. Those local features are then aggregated per keypoint into a single representation. Finally, the final pose is inferred through a MLP fed by those keypoint representations.

spacecraft pose and the ground-truth:

$$\mathcal{L}_{PIN} = \frac{1}{K}\sum_{i=1}^{K} ||(\hat{\mathbf{R}}\mathbf{p}_i + \hat{\mathbf{t}}) - (\mathbf{R}\mathbf{p}_i + \mathbf{t})||_2, \tag{6}$$

where $\hat{\mathbf{R}}$ and $\mathbf{R}$ are the estimated and true rotation matrices, $\hat{\mathbf{t}}$ and $\mathbf{t}$ are the estimated and true translation vectors, $\mathbf{p}_i$ is the 3D coordinates of the i^{th} keypoint in the spacecraft 3D model and K is the number of keypoints.

2.3 Joint Training of KDN and PIN

Training a concatenation of the KDN and PIN with the PIN loss only is unlikely to converge, because the output of the PIN can only make sense if its input consists of keypoint predictions clustered around their ground truth. We therefore train the model as follows: We first train the KDN alone, using the combined loss described in section 2.1. Next, we train a concatenation of the KDN and PIN with all KDN weights frozen, using the 3D reconstruction loss described in section 2.2. Finally, we unfreeze the KDN weights and fine-tune both networks jointly. Since both aim at complementary but different goals, the total loss must represent both of them. We opt for a weighted sum of $\mathcal{L}_{KDN}$ and $\mathcal{L}_{PIN}$:

$$\mathcal{L}_{Tot} = \mathcal{L}_{KDN} + \beta_{pose}\mathcal{L}_{PIN} \tag{7}$$

3 Experiments

This section considers the on-orbit pose estimation task. Section 3.1 discusses the dataset and implementation details considered in the following. Section 3.2

assesses the proposed framework performance on the SPEED [29] dataset and compares its performance with previous works. Section 3.3 studies how complexity reduction affects the pose estimation accuracy. Finally, Section 3.4 presents some ablation studies of the Keypoint Detection Network and the Pose Inference Network.

3.1 Dataset and Implementation Details

In this section, we evaluate our framework's performance on SPEED [29] and compare it to previous works. SPEED was used in the Satellite Pose Estimation Challenge (SPEC) [17] that was organized in 2019 by the Advanced Concepts Team (ACT) of the European Space Agency (ESA) and the Space Rendezvous Laboratory (SLAB) of Stanford University. It consists of 15,000 synthetic images and 305 real images. An OpenGL-based rendering pipeline was used to produce the 1920x1200, grayscale, synthetic images depicting the target spacecraft at distances between 3 and 40.5 meters. On half of them, random Earth images were added. The synthetic images were post-processed with Gaussian blur and Gaussian noise. The dataset is split into 12,000 images for training and 3,000 for testing. For each training image, the ground-truth pose label is provided under the from of a translation vector and a unit quaternion that represents the relative rotation between the camera and the target spacecraft [17]. The real images were produced in the Testbed for Rendezvous and Optical Navigation (TRON) facility of SLAB [20].

80% of the training set was used to train the model while the model accuracy was evaluated on the remaining 20%. Training was carried on a NVIDIA Tesla A100 while inference was performed on a NVIDIA GeForce RTX 2080 Ti to provide a fair comparison with previous works.

Two assumptions are adopted through the whole section.

- We have at our disposal a perfect Object Detection Network to crop the input image from its initial resolution of 1920x1200 pixels to the KDN input resolution, i.e. 608x608 pixels. In practice, we follow the process used in [21] where the crop is taken as the smallest square containing all the keypoints enlarged by 20% in inference or by a random percentage (from 0 up to 50%) during training. In addition, both horizontal and vertical shifts of at most 20% of the crop size are applied during training.
- A wireframe model of the spacecraft is provided. Here, as SPEED does not contain the spacecraft model, we used the one recovered by Chen *et al* [3]. In this model, the keypoints are defined as the 8 corners of the spacecraft body combined with the top of the 3 antennas as depicted in Figure 5 (a).

The KDN backbone and decoding streams were pre-trained on Linemod [15,12] and then trained for 200 epochs using SGD with a momentum of 0.9 and a weight decay of $1e^{-4}$. After a grid search, the initial learning rate was set to $5e^{-3}$ and divided by 10 at [50, 75, 90] % of the training. We used brightness, contrast and noise data augmentation as explained in section 3.4. After a grid search,

Table 1. Our framework performance compared to previous works. Our two-network solution achieves slightly better performance than the solution combining the first network with a PnP strategy, at a higher throughput. It is also close to, but not yet on par with, state-of-the art solutions. Values from [15,21,4,16] are copied from their original paper, or from [16] (indicated by the 1 in superscript).

		Ours		Previous works			
Metric		DarkNet-PIN	DarkNet-PnP	Segdriven [15]	KRN [21]	DLR [3]	WDR [16]
Complexity [M]	↓	72.1	71.2	89.2[1]	5.64	176.2[1]	51.5
Throughput [fps]	↑	31	23	12[1]	∼70	0.7[1,*]	18-35*
score_{ESA} [/]	↓	0.098	0.112	0.02 [17]	0.073	0.012[1]	0.016-0.018
mean E_T, N [%]	↓	1.589	2.561	-	1.9	-	-
median E_T, N [%]	↓	1.048	0.820	-	-	-	-
mean E_T [m]	↓	0.201	0.267	-	0.211	0.0359	-
median E_T [m]	↓	0.089	0.078	-	0.124	0.0147	-
mean E_R [°]	↓	4.687	4.957	-	3.097	0.728	-
median E_R [°]	↓	3.272	1.402	-	2.568	0.521	-

the loss hyperparameters were fixed to $\beta_{seg} = 1$, $\beta_{kpts} = 4$, $\beta_{conf} = 1$, $\tau = 5$ and $\beta_{pose} = 1$. Regarding the strategy used to select the $K \times m$ correspondences out of the KDN predictions, we observed that selecting the correspondences as the m most confident predictions per keypoint did not provide any improvement compared to randomly sampling them from the predicted foreground mask. We therefore adopted this second, simpler, strategy.

3.2 Comparisons with a PnP-based Solution and Prior Works

Table 1 ranks competing solutions according to three metrics: complexity, frames per second, and pose accuracy. We define complexity as the number of parameters in a model. We opted for this definition because of its generality and hardware-independence, by contrast to floating point operations which depend on software implementation and hardware configuration. In addition to this definition of complexity, we provide an empirical measure of complexity given by the processing rate (FPS) of each model on a GeForce RTX 2080 Ti. The third metric provided in Table 1 reflects pose accuracy. We follow the definition proposed by ESA in SPEC2019, which is defined as a sum of normalized translation and rotation errors [17]:

$$score_{ESA} = \frac{1}{N}\sum_{i=1}^{N}\left[E_{T,N}^{(i)} + E_{R}^{(i)}\right] \tag{8}$$

The normalized translation error on image i, $E_{T,N}^{(i)}$ is defined by the following equations where $t_*^{(i)}$ and $t^{(i)}$ are the ground-truth and estimated translations, respectively.

$$E_{T,N}^{(i)} = \frac{E_T^{(i)}}{\left|t_*^{(i)}\right|_2} \quad \text{where} \quad E_T^{(i)} = \left|t_*^{(i)} - t^{(i)}\right|_2 \tag{9}$$

For each image i, the rotation error, in radians, between the predicted and ground-truth quaternions is computed as:

$$E_R^{(i)} = 2\arccos\left(\left|\left\langle q_*^{(i)}, q^{(i)}\right\rangle\right|\right) \tag{10}$$

where $q_*^{(i)}$ and $q^{(i)}$ are the ground-truth and estimated quaternions, respectively.

As highlighted by Table 1, our solution, which combines a Keypoint Detection Network with a Pose Inference Network, achieves a pose estimation accuracy similar to the one obtained when using a more classical RANSAC-PnP solver applied on the keypoints detected by the same KDN. Furthermore, our solution runs 34% faster than the RANSAC-PnP approach. However, even if both solutions lead to a decent accuracy, they remain beyond the state-of-the-art methods. Those preliminary results are promising, and demonstrate the relevance of studying architectures composed of two networks when dealing with space images[4]. Paths for improvements include joint end-to-end training of both networks, but also end-to-end learning of keypoints, for example using an approach similar to D2-Net [6].

3.3 Complexity Reduction

Although SPEC2019 showed that it was possible to accurately estimate the pose of a spacecraft using a conventional camera and neural networks, it also highlighted the fact that those methods are often too complicated to run on space-grade hardware [17]. This section therefore evaluates the complexity-accuracy trade-off of our framework. Because the PIN accounts for only 1.3% of the total size of our model, we limit this section to a discussion of the complexity-accuracy trade-off of the KDN.

Backbone Figure 7 summarizes the complexity-accuracy trade-off obtained by 6 different backbones (DarkNet-53 [24], ShuffleNet v2 [19], EfficientNet [31], MobileNet v2 [27], MobileNet v3 small [13] and MobileNet v3 large [13]). All of them were pre-trained on ImageNet [26], except DarkNet-53 which was pre-trained on Linemod [12], and then trained on SPEED using the training strategy

[4] The accuracy of our solution must still be measured on spaceborne images.

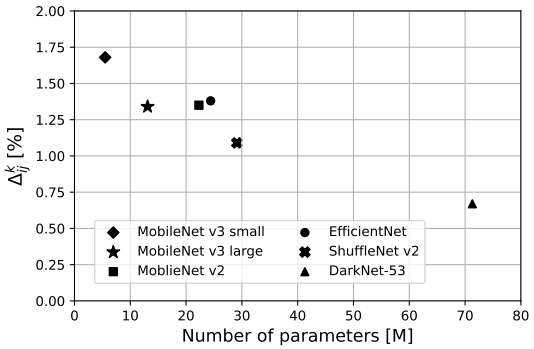

Fig. 7. Complexity/accuracy trade-off for 6 different KDN backbones. The common trend is that larger networks achieve lower errors on keypoint locations.

described in Section 3.1. Figure 7 reveals that the large version of MobileNet v3 offers the best trade-off among the selected architectures.

MobileNet v3 small, large and ShuffleNet offer interesting trade-offs. MobileNet v3 large offers a trade that contrast sharply with DarkNet, and it positions itself as a compromise between MobileNet v3 small and ShuffleNet. We therefore chose to use it as an additional baseline.

Segmentation and Regression Heads. To further reduce the KDN complexity, we may simplify the 2 decoding heads as it becomes relevant when using most simple backbones. Each head is made of 3 stages that process the features map at different spatial resolutions. Each stage is made of 3 blocks that implement depth-wise convolutions. To simplify the network, we simply decrease the number of blocks in each stage.

Figure 8 summarizes the complexity-accuracy trade-off that occurs in a KDN made of a large MobileNet v3 with a regression stream made of 3 stages composed of 1,2 or 3 blocks and a segmentation stream made of 3 stages composed of 1,2 or 3 blocks.

We draw the following conclusions:

- Adopting a single block per stage in the segmentation head reduces by 30% the number of total parameters without impacting accuracy.
- Reducing the number of parameters in the regression head by a factor 0.65 inflates the keypoint estimation error, $\overline{\Delta_{ij}^k}$, by a factor 1.3, a trade that many end-users are likely to consider worthy.

As a result, we decided to use a large MobileNet v3 with regression and segmentation streams made of 3 stages composed of 2 and 1 blocks, respectively. Such a network achieves a mean error of 2.05% on the keypoint locations while using only 7.8 millions parameters. Compared to our first baseline, based on DarkNet, which uses 71.2 millions parameters to reach a 0.67% accuracy, it represents a solution that may fit on space-grade hardware while achieving a

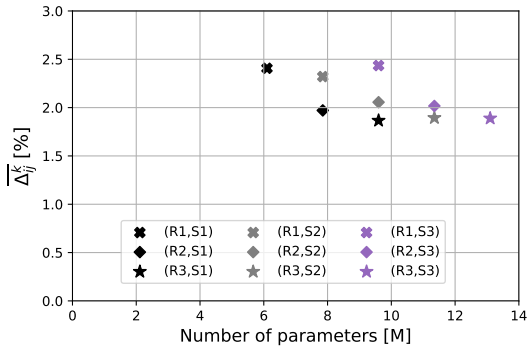

Fig. 8. Complexity/accuracy trade-off for a KDN based on MobileNet v3 large with 9 different heads architectures. Each setting is labelled (RX,SY) where X is the number of blocks per stage in the regression head and Y is the number of blocks per stage in the segmentation one. Reducing the complexity of the regression head increases the error on the keypoints. The complexity of the segmentation head has a negligible impact on the predictions accuracy.

Table 2. Average error on the keypoint locations ($\mathbb{E}\left[\Delta_{ij}^{k}\right]$) in percent of the image size for different data augmentation techniques

Bright./Contr.	Noise	Rotation	Background	$\mathbb{E}\left[\Delta_{ij}^{k}\right]$ [%]
				1.33
✓				0.62
	✓			1.18
		✓		1.34
			✓	1.66
✓	✓			0.67

sufficient accuracy on the keypoint locations. This results in an ESA-score twice larger than the one obtained with DarkNet in Table 1.

3.4 Ablation study

This section studies the impact of the data augmentation techniques used to train the KDN and evaluates the impact of the PIN architecture on the complexity-accuracy trade-off.

KDN Data Augmentation. Table 2 summarizes the mean error on the keypoint predictions normalized by the image size obtained by the KDN when trained with different data augmentation techniques. Each data augmentation is applied randomly on each image with a 50% probability.

Table 3. Pose Estimation Accuracy for 2 global pose inference MLPs of different complexities.

MLP layers	mean E_T [%]	mean E_R [°]	score$_{ESA}$ [/]	parameters [M]
$1408 \rightarrow 512 \rightarrow 256 \rightarrow 7$	1.59	4.69	0.098	**0.86**
$1408 \rightarrow 1024 \rightarrow 512 \rightarrow 7$	**1.37**	**3.87**	**0.081**	1.97

Table 4. Pose Estimation Accuracy achieved by the Pose Inference Network when fed with the predictions emitted by the ground-truth foreground cells or with the predictions made by the foreground cells predicted by the KDN.

Candidates sampling strategy	mean E_T [%]	mean E_R [°]	score$_{ESA}$ [/]
PIN fed with foreground ground-truth	2.30	6.60	0.138
PIN fed with predicted mask	**1.59**	**4.69**	**0.098**

- **Brightness/Contrast.** We make use of brightness/contrast data augmentation to account for the wide diversity of illumination scenarios encountered in orbit. In practice, we follow the method proposed by Park *et al.* [21]. The contrast is modified by a random factor in between 0.5 and 2 while the brightness is randomly increased by at most 10%. Such a data augmentation provides a reduction of 0.7% on the mean error on the keypoint locations, $\overline{\Delta_{ij}^k}$.
- **Gaussian Noise.** Adding Gaussian noise which variance is randomly chosen below 10% of the image range decreases by 0.14% the average error.
- **Rotation.** Random rotations of [90,180,270]° did not provide any improvements compared to unaugmented training.
- **Background.** We tested a random background data augmentation technique. Using the segmentation mask recovered in 2.1, we paste the masked original image on an image randomly taken from the PASCAL-VOC dataset [7]. Unfortunately, this data augmentation did not provide any improvement, even worse, it decreased the keypoint detection accuracy. The reason of such a decrease probably comes from the raw cut-and-paste strategy used in the data augmentation. In particular, since the background and spacecraft brightness's are not similar, the network may have learned to detect changes of brightness which do not exist on the original image.
- As both **Brightness/Contrast and Noise** data augmentation improve the KDN performance, both were combined, leading to an error reduction of 0.66%.

PIN Architecture. In this section, we discuss some elements of the Pose Inference Network that impact the accuracy of our system.

Despite its appealing performance, the Pose Inference Network is not yet on par with existing solutions. However, as summarized in Table 3, adding more parameters in the global pose inference MLP improves its ability to predict

accurate pose labels. We may therefore expect that a smarter architecture may lead to more accurate pose estimates.

Table 4 compares the accuracy of a PIN network fed with candidate locations extracted from either the segmentation map predicted by the KDN or the foreground mask. Using the ground-truth foreground mask leads to an ESA-score increased by 50% compared to the KDN segmentation output. Since the misclassifications happen mainly in some specific parts of the image such as the spacecraft edges, antennas or darker areas, we believe that there is a link between the nature of the different patches and the accuracy of their predictions.

The PIN could therefore be enhanced by associating a keypoint-specific level of confidence to the prediction of the 2D candidates, e.g. to take into account the visibility of each keypoint in the 2D image. Furthermore, the Pose Inference Network could be modified to learn by itself the 3D keypoints according to its ability to accurately predict them.

4 Conclusions

This paper has investigated a spacecraft pose estimation framework adopting a first, convolutional, neural network to predict candidate locations of some pre-defined keypoints, from which a fixed number of candidates are randomly sampled from a predicted foreground mask to feed a second neural network that infers the spacecraft 6D pose.

The method was tested on SPEED [29] where it achieves a mean rotation error of 4.69°, a mean translation error of 1.59% and an ESA-score of 0.098 for a throughput of 31 fps. This is promising in terms of accuracy, although behind the performance of state-of-the-art solutions. Furthermore, our work reveals that the complexity of the network initially recommended in [14] can be significantly reduced to run on space-grade hardware while preserving a reasonable accuracy. Finally, we highlighted that the Pose Inference Network can be improved either by exploring smarter architectures or by feeding it with the confidence associated to the keypoint candidates.

Acknowledgments Special thanks go to Mikko Viitala and Jonathan Denies for the supervision of this work within Aerospacelab. The research was funded by Aerospacelab and the Walloon Region through the Win4Doc program. Christophe De Vleeschouwer is a Research Director of the Fonds de la Recherche Scientifique - FNRS. Computational resources have been provided by the supercomputing facilities of the Université catholique de Louvain (CISM/UCL) and the Consortium des Équipements de Calcul Intensif en Fédération Wallonie Bruxelles (CÉCI) funded by the Fond de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under convention 2.5020.11 and by the Walloon Region.

References

1. Black, K., Shankar, S., Fonseka, D., Deutsch, J., Dhir, A., Akella, M.R.: Real-time, flight-ready, non-cooperative spacecraft pose estimation using monocular imagery. arXiv preprint arXiv:2101.09553 (2021)

2. Carcagnì, P., Leo, M., Spagnolo, P., Mazzeo, P.L., Distante, C.: A lightweight model for satellite pose estimation. In: International Conference on Image Analysis and Processing. pp. 3–14. Springer (2022)
3. Chen, B., Cao, J., Parra, A., Chin, T.J.: Satellite pose estimation with deep landmark regression and nonlinear pose refinement. In: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops. pp. 0–0 (2019)
4. Chen, B., Parra, A., Cao, J., Li, N., Chin, T.J.: End-to-end learnable geometric vision by backpropagating pnp optimization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8100–8109 (2020)
5. Cosmas, K., Kenichi, A.: Utilization of fpga for onboard inference of landmark localization in cnn-based spacecraft pose estimation. Aerospace **7**(11), 159 (2020)
6. Dusmanu, M., Rocco, I., Pajdla, T., Pollefeys, M., Sivic, J., Torii, A., Sattler, T.: D2-net: A trainable cnn for joint description and detection of local features. In: Proceedings of the ieee/cvf conference on computer vision and pattern recognition. pp. 8092–8101 (2019)
7. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International journal of computer vision **88**(2), 303–338 (2010)
8. Furano, G., Meoni, G., Dunne, A., Moloney, D., Ferlet-Cavrois, V., Tavoularis, A., Byrne, J., Buckley, L., Psarakis, M., Voss, K.O., Fanucci, L.: Towards the use of artificial intelligence on the edge in space systems: Challenges and opportunities. IEEE Aerospace and Electronic Systems Magazine **35**(12), 44–56 (2020). https://doi.org/10.1109/MAES.2020.3008468
9. Garcia, A., Musallam, M.A., Gaudilliere, V., Ghorbel, E., Al Ismaeil, K., Perez, M., Aouada, D.: Lspnet: A 2d localization-oriented spacecraft pose estimation neural network. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2048–2056 (2021)
10. Gerard, K.: Segmentation-driven satellite pose estimation. Kelvins Day Presentation, URL: `https://indico.esa.int/event/319/attachments/3561/4754/pose_gerard_segmentation.pdf` (2019)
11. Goodwill, J., Crum, G., MacKinnon, J., Brewer, C., Monaghan, M., Wise, T., Wilson, C.: Nasa spacecube edge tpu smallsat card for autonomous operations and onboard science-data analysis. In: Proceedings of the Small Satellite Conference. No. SSC21-VII-08, AIAA (2021)
12. Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., Navab, N.: Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In: Asian conference on computer vision. pp. 548–562. Springer (2012)
13. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al.: Searching for mobilenetv3. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 1314–1324 (2019)
14. Hu, Y., Fua, P., Wang, W., Salzmann, M.: Single-stage 6d object pose estimation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 2930–2939 (2020)
15. Hu, Y., Hugonot, J., Fua, P., Salzmann, M.: Segmentation-driven 6d object pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3385–3394 (2019)
16. Hu, Y., Speierer, S., Jakob, W., Fua, P., Salzmann, M.: Wide-depth-range 6d object pose estimation in space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15870–15879 (2021)

17. Kisantal, M., Sharma, S., Park, T.H., Izzo, D., Märtens, M., D'Amico, S.: Satellite pose estimation challenge: Dataset, competition design, and results. IEEE Transactions on Aerospace and Electronic Systems **56**(5), 4083–4098 (2020)
18. Li, Y., Wang, G., Ji, X., Xiang, Y., Fox, D.: Deepim: Deep iterative matching for 6d pose estimation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 683–698 (2018)
19. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: Proceedings of the European conference on computer vision (ECCV). pp. 116–131 (2018)
20. Park, T.H., Bosse, J., D'Amico, S.: Robotic testbed for rendezvous and optical navigation: Multi-source calibration and machine learning use cases. arXiv preprint arXiv:2108.05529 (2021)
21. Park, T.H., Sharma, S., D'Amico, S.: Towards robust learning-based pose estimation of noncooperative spacecraft. 2019 AAS/AIAA Astrodynamics Specialist Conference (2019)
22. Proença, P.F., Gao, Y.: Deep learning for spacecraft pose estimation from photorealistic rendering. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). pp. 6007–6013. IEEE (2020)
23. Rathinam, A., Gao, Y.: On-orbit relative navigation near a known target using monocular vision and convolutional neural networks for pose estimation. In: i-SAIRAS (2020)
24. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767 (2018)
25. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)
26. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision **115**(3), 211–252 (2015)
27. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4510–4520 (2018)
28. Sharma, S., Beierle, C., D'Amico, S.: Pose estimation for non-cooperative spacecraft rendezvous using convolutional neural networks. In: 2018 IEEE Aerospace Conference. pp. 1–12. IEEE (2018)
29. Sharma, S., D'Amico, S.: Pose estimation for non-cooperative rendezvous using neural networks. arXiv preprint arXiv:1906.09868 (2019)
30. Sharma, S., Ventura, J., D'Amico, S.: Robust model-based monocular pose initialization for noncooperative spacecraft rendezvous. Journal of Spacecraft and Rockets **55**(6), 1414–1429 (2018)
31. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International conference on machine learning. pp. 6105–6114. PMLR (2019)
32. Xiang, Y., Schmidt, T., Narayanan, V., Fox, D.: Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. arXiv preprint arXiv:1711.00199 (2017)