Deep Learning for Real-Time Satellite Pose Estimation on Tensor Processing Units

Engineering Notes

Deep Learning for Real-Time Satellite Pose Estimation on Tensor Processing Units

Alessandro Lotti,* Dario Modenini,† and Paolo Tortora‡
University of Bologna, 47121 Forlì, Italy
and
Massimiliano Saponara§ and Maria A. Perino¶
Thales Alenia Space Italia, 10146 Turin, Italy

Nomenclature

E	=	mean pose error
e	=	error
$\boldsymbol{q}$	=	quaternion
$\boldsymbol{t}$	=	relative position

Subscripts

EST	=	estimated
GT	=	ground truth
q	=	quaternion
t	=	relative position

I. Introduction

VISION-BASED navigation is a key technology for the next generation of on-orbit servicing (OOS) and active debris removal missions. In these scenarios, guidance and control laws shall be fed with the relative chaser-to-target pose (i.e., position and attitude), which might be conveniently estimated from monocular images because these sensors are simple, light, and consume little power. Traditionally, image processing algorithms are divided into categories of 1) hand-crafted features [1,2] and 2) deep learning based [3–14]. However, the former are affected by low robustness against typical space imagery characteristics (such as low signal-to-noise-ratio, severe, and rapidly varying illumination conditions) and backgrounds. Neural networks (NNs) could overcome such weaknesses through proper training but often result in a high computational burden that is hardly compatible with typical onboard processing power.

*Ph.D. Candidate, Department of Industrial Engineering, Alma Mater Studiorum - Università di Bologna; alessandro.lotti4@unibo.it.

†Assistant Professor, Department of Industrial Engineering and Interdepartmental Center for Industrial Research in Aerospace, Alma Mater Studiorum - Università di Bologna; dario.modenini@unibo.it.

‡Full Professor, Department of Industrial Engineering and Interdepartmental Center for Industrial Research in Aerospace, Alma Mater Studiorum - Università di Bologna; paolo.tortora@unibo.it. Associate Fellow AIAA.

§Head of Guidance, Navigation and Control Department, Turin; massimiliano.saponara@thalesaleniaspace.com.

¶Director Space Economy Exploration and International Network; mariaantonietta.perino@thalesaleniaspace.com.

In recent years, deep learning has been shown to aid spacecraft monocular pose estimation at different levels; see, e.g., Refs. [15,16] for thorough surveys. Sharma et al. [3] addressed the problem as a classification task by discretizing the pose space. Later, Sharma and D'Amico [4] proposed a solution based on joint classification and regression. In this last work, Sharma and D'Amico presented the spacecraft pose estimation dataset (SPEED), providing synthetic and real high-resolution (1920 × 1200 pixels) images of the Tango spacecraft. The SPEED has been adopted as a benchmark in the first Satellite Pose Estimation Challenge (SPEC) [17] cohosted by the European Space Agency and Stanford University's Space Rendezvous Laboratory (SLAB). Three out the four top-scoring works adopted a three-stage approach, leveraging convolutional neural networks (CNNs) to detect the target on the image first and to regress the locations of some predefined key points later, which are then fed into off-the-shelf perspective-n-point solvers [5,6] for pose estimation. On the other hand, the authors of Ref. [7] proposed an end-to-end CNN based pipeline that, however, required a huge model to achieve competitive accuracy. Indeed, submissions were ranked solely on a pose regression error, regardless of their computational burden. Only the SLAB baseline [6] addressed this issue by adopting light networks that, however, led to a pose estimation error significantly higher than the best one.

Later works addressed the pose estimation accuracy–latency tradeoff. To this aim, Hu et al. [8] proposed a single-stage method leveraging a three-dimensional (3-D) loss to make pose estimation less sensitive to scale variations. Wang et al. [9] exploited Transformers for direct key point coordinates retrieval upon a CNN-based detection step, whereas Piazza et al. [10] adopted a light model for target detection. Carcagnì et al. [11] revisited the method in Ref. [5] by replacing the landmarks regression network backbone with a lighter variant. Similarly, Posso et al. [12] revisited the end-to-end approach proposed in Ref. [7] in a light manner, reaching an accuracy that was far from the top submissions in the SPEC.

These works have been tested on high-end desktop graphics processing unit only, and they are not optimized for low-power embedded devices that typically perform well on limited sets of operations as a consequence of hardware specialization. Black et al. [13] contributed remarkably in this respect by developing a light pipeline that achieves real-time inference on an Intel Joule 570x board consuming 3.7 W.

In this context, our work focuses on further improving the pose estimation accuracy–latency tradeoff on low-power devices from both the software and hardware points of view. First, we propose a pose estimation pipeline based on NNs optimized for embedded devices with an architecture that can be scaled to the available computational power. We investigate optimizations through TensorFlow Lite (TFLite) conversion and quantization. The TFLite is a machine learning (ML) library for on-device inference, whereas the quantization consists of converting NNs' weights to integers, yielding to a significant runtime advantage at deployment.

Second, we investigate high-performing ML coprocessors for low-power integer-only inference, namely, Edge Tensor Processing Units (TPUs), which are gaining the attention of the space community [18]. We thus test our models on a Coral Dev Board Mini equipped with both a TPU and a 1.5 GHz quadcore Advanced Reduced Instruction Set Computer Machines CPU.

We evaluate our algorithms on both the SPEED [4] and on a new dataset developed as part of this work, depicting a spacecraft from the Constellation of Small Satellites for the Mediterranean Basin Observation (COSMO), named the COSMO photorealistic dataset (CPD). Our results show how model optimization and edge processors can enable subdegree and centimeter-level real-time pose estimation compatible with typically available onboard power levels.
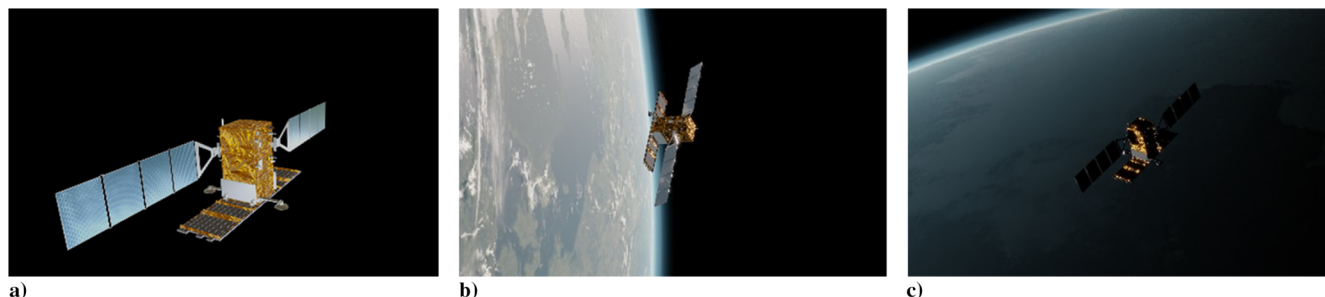
Fig. 1 Close-up preview of the COSMO SkyMed satellite (Fig. 1a), and sample images from CPD before postprocessing (Figs. 1b and 1c).

II. Photorealistic Dataset

Moving components, such as antennas and solar arrays, are common for large spacecrafts, i.e., the ones that would benefit the most from OOS. The need to validate pose estimation pipelines, even in this scenario, drove the design of a new dataset, named the COSMO photorealistic dataset, depicting a satellite from the SkyMed Earth-observation constellation in heterogeneous combinations of poses, solar arrays configurations, lighting conditions, and backgrounds. To this end, the 3-D computer graphics software, Blender, was selected because of its native support to physically based rendering (PBR). Most assumptions adopted for setting the spacecraft pose distribution and image postprocessing follow that of the SPEED [4] for ease of benchmarking, as will be detailed in the following.

A. Blender Scene

The Blender scene consists of three concentric highly polygonal spheres representing Earth, clouds, and the atmosphere plus a CAD model of the COSMO spacecraft. Similar to Ref. [7], the Earth was textured with a high-resolution map further augmented with an ocean mask[**] and topography data from the NASA's Blue Marble collection,[††] which also provided cloud texture. A third-party shader[‡‡] was applied to increase the realism of the clouds, providing at the same time the transparency, diffusion, and reflection. Atmospheric scattering was emulated, exploiting Blender's volumetric rendering in combination with a second shader, from the same package, which implements an exponential density model. A Blender sun lamp emulates the sun by providing collimated light at a blackbody temperature of 5778 K.

The main exterior features of COSMO spacecraft are the multilayer insulation (MLI) and the solar panels. A faithful representation of the MLI material was obtained thanks to a crumpled normal texture[§§] mapped to the spacecraft body for emulating the typical random reflections. The solar arrays have been equipped with a solar cell texture providing base color and displacement, whereas surface reflection has been obtained through Blender's glossy shader.

B. Pose Distribution

The distance is randomly selected from a standard normal distribution ($\mu = 36$ m and $\sigma = 10$ m) rejecting all the values above 70 m and below 36 m. The x-y offsets on the image plane are uncorrelated random values selected from a multivariate normal distribution, which are constrained to guarantee that the satellite almost always lies entirely in the image frame. The attitude is selected randomly from a uniform distribution of all rotations in the 3-D space.

Pose distribution and camera–satellite alignment are governed by the Starfish library [14]. Domain variation is provided through Blender Python interface by rotating the Earth and clouds beneath the satellite before each rendering. Geometrical constraints are prescribed to avoid

dark images. The sun–probe–Earth angle is required to be greater than 70 deg to exclude eclipse conditions. We also prescribe that at least one of the surfaces of the spacecraft main bus is illuminated by the sun and at the same time in view of the camera. This, in turn, is obtained by requiring that 1) the angle between the sun direction and the outward surface normal is smaller than 80 deg, and 2) that the angle between the position vector of the camera and the surface normal is smaller than 70 deg. Besides that, the lighting direction is randomized across the dataset to provide a comprehensive range of illumination conditions. An approximate sun-tracking rotation of solar panels is added through a Blender "locked track" constraint. It consists of setting the orientation of the solar arrays about the longitudinal axis to have them pointed toward the projection of the sun vector in the plane orthogonal to the rotation axis. This is a reasonable assumption for a spacecraft equipped with steerable solar panels.

C. Render Setup and Postprocessing

A total of 15,000 images has been rendered with the PBR Cycles engine through a pinhole camera model. The resolution has been set to 1920×1200 pixels. Postprocessing steps include a glare node (meant to replicate the bloom effect), grayscale conversion, and the addition of Gaussian noise and Gaussian blurring to emulate the shot noise and depth of field. Figure 1 depicts a close-up rendering of the satellite and two samples from the dataset before grayscale conversion and noise addition.

III. Methods

Our software pipeline is based on three stages, namely, 1) spacecraft detection, 2) landmarks' regression, and 3) pose estimation. For this work, we hold the assumption that the target is known by the chaser; i.e., a wireframe model is available. Because this information was not included in the SPEED [4] dataset, we reconstructed a 3-D model through multiview triangulation. We then retrieved training and testing labels (i.e., bounding boxes and landmarks' coordinates) for both datasets by projecting the 3-D satellites' key points, highlighted in Fig. 2, onto the image plane exploiting the known target pose.

A. Spacecraft Detection

Direct processing of high-resolution images would prevent real-time inference on low-power embedded devices due to the need for large NNs and a high memory footprint. The purpose of the detection network (DN) is therefore to identify a region of interest (ROI) by detecting the satellite on the image.

To this end, we employed a MobileDet Edge TPU optimized model [19] from the TensorFlow (TF) Detection Model Zoo[¶¶] with an input shape of 320×320 pixels and about 3.3 million parameters. The matching between the true and the estimated bounding boxes is assessed in terms of the intersection over union (IOU). To avoid distortion, the regressed bounding box is made square by enlarging the shortest side to match the longest one while keeping the box center fixed. The edges of the bounding box are further expanded by a factor of 1.15 to increase the chance that the satellite will lie within its

---

[**]Data available online at https://www.shadedrelief.com [retrieved 22 February 2021].

[††]Data available online at https://visibleearth.nasa.gov [retrieved 22 February 2021].

[‡‡]Data available online at https://gumroad.com/l/JlNTt [retrieved 22 February 2021].

[§§]Data available online at https://nasa3d.arc.nasa.gov/detail/Sentinel-6 [retrieved 26 February 2021].

[¶¶]Data available online at https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md [retrieved 25 October 2021].

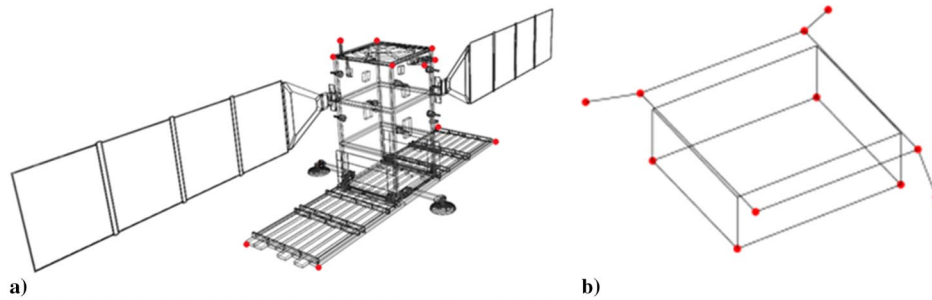**Fig. 2   Wireframe models: a) COSMO, and b) Tango.**

Table 1   Regression head structure

No.	Type	Activation function	Kernel size	Padding	No. of filters
1	Standard	RELU	1	Valid	128
2	Separable	RELU	3	Same	128
3	Standard	——	1	Valid	22
4	Standard	——	Previous layer resolution	Valid	22

Table 2   EfficientNet-Lite models specifications relative to Lite0 variant

Model	Input size, pixels	Width coefficient	Depth coefficient
Lite0	224×224	1.0	1.0
Lite1	240×240	1.0	1.1
Lite2	260×260	1.1	1.2
Lite3	280×280	1.2	1.4
Lite4	300×300	1.4	1.8

margins. The original image is eventually padded with black pixels whenever the ROI extends beyond its sides. In case the ROI is smaller than the second NN input shape, its borders are further enlarged to avoid excessive artifacts arising from direct resizing.

B.   Landmark Regression

The resulting ROI is fed into a second CNN, which is in charge of detecting a set of predefined key points. We propose a regression network (RN) architecture that can be scaled to different accuracy–latency tradeoffs. It consists of a fully convolutional model built on top of EfficientNet-Lite backbones,[***] which are obtained by removing operations not well supported by mobile accelerators from the original EfficientNet [20]. The regression head consists of a sequence of four convolutions (Table 1), which gradually reduce the feature map channels and dimensions through learned operations rather than pooling. The network returns a vector containing normalized coordinates of the landmarks in a predefined order.

EfficientNet-Lite backbones come in five variants that are obtained by scaling up the network's width, depth, and input resolution of a baseline model, as outlined in Table 2.

The target position and orientation are estimated through an EPnP solver [21] with random sample consensus exploiting the known two-dimensional/3-D correspondences. The resulting pose is eventually refined with a Levenberg–Marquardt optimization step.[†††]

IV.   Experiments and Results

Our models are first compared with state-of-the-art methods using the floating point, nonoptimized NNs configuration. We then illustrate the benefits obtained through TensorFlow Lite[‡‡‡] conversion first and quantization later. These modifications enable the deployment of the models on a low-power machine learning accelerator (namely, a TPU), which can further reduce the inference time.

A.   Training Setup

For the SPEED dataset, because the ground truth poses for the test sets have not been released at the time of this writing, only the 12,000 synthetic images from the SPEC training set were used. To prevent overfitting, a cross-validation framework has been adopted. To this aim,

both the SPEED and CPD have been divided into equally sized partitions containing 3000 images each. Given k partitions, each step of the cross-validation process consists of training the models on $k - 1$ partitions and tests it on the holdout one. The test results are then aggregated to compute average metrics. The DN is trained for 25,000 steps with a momentum optimizer starting from COCO's [22] pretrained weights, applying random crops as well as random brightness and contrast adjustments to avoid overfitting. RNs have been trained with an Adam optimizer [23] and a mean absolute error loss for 500 epochs on the SPEED and 375 epochs on the CPD. EfficientNet-Lite backbones were initialized with Imagenet [24] checkpoints. Applied data augmentations include random image rotations, bounding box enlargements and shifts, random brightness, and contrast adjustments.

The DN learning rate was gradually reduced according to a cosine decay law after a warmup phase, lasting 2000 steps, where it grew linearly from 0.015 to 0.45; whereas for the RNs, a linear scheduling was adopted, ranging from 2e-3 to 2e-5. The batch size was set to 200 for the DN and 250 for the RNs.

The trained networks were first deployed on the CPU of a Coral Dev Board Mini for performance assessment. Later, all networks were retrained by exploiting quantization aware training[§§§] (QAT), which emulates quantization along with NN parameter tuning, to reduce accuracy loss at conversion.

The results provided in the next paragraphs refer to the SPEED [4] unless otherwise stated.

B.   Comparison with State of the Art

To assess the accuracy of our methods, we adopt the same metric of the SPEC [17]. This is based on the sum of a normalized position $\bar{e}_t$ and rotation e_q errors averaged over all the N images:

$$e_{t_i} = |\boldsymbol{t}_{\text{GT}_i} - \boldsymbol{t}_{\text{EST}_i}|_2 \tag{1}$$

$$\bar{e}_t = \frac{1}{N} \sum_{i=1}^{N} \frac{e_{t_i}}{|\boldsymbol{t}_{\text{GT}_i}|_2} \tag{2}$$

$$e_q = \frac{1}{N} \sum_{i=1}^{N} 2\arccos(|\langle \boldsymbol{q}_{\text{GT}_i}, \boldsymbol{q}_{\text{EST}_i} \rangle|) \tag{3}$$

$$E = \bar{e}_t + e_q \tag{4}$$

Table 3[¶¶¶] displays the error metrics attained by all of our pipeline variants along with those of the best-ranked submissions of the SPEC

[***]Data available online at https://blog.tensorflow.org/2020/03/higher-accuracy-on-vision-models-with-efficientnet-lite.html [retrieved 30 March 2021].

[†††]Data available online at https://docs.opencv.org/3.4.12/d9/d0c/group__calib3d.html [retrieved 5 April 2021].

[‡‡‡]Data available online at https://tensorflow.org/lite [retrieved 9 November 2021].

[§§§]Data available online at https://tensorflow.org/model_optimization/guide/quantization/training?hl=en [retrieved 12 November 2021].

[¶¶¶]Gerard, K., "Segmentation-Driven Satellite Pose Estimation," Kelvins Day Presentation, 2019, https://indico.esa.int/event/319/attachments/3561/4754/pose_gerard_segmentation.pdf [retrieved 4 January 2021].

Table 3 Comparison with state of the art [6–8,13,17]

| Model	E	e_q, deg	e_t, m	Parameters, millions
UniAdelaide [5]	0.0094	0.41 ± 1.50	0.032 ± 0.095	176.2
Lite4 (ours)	0.0143	0.57 ± 0.75	0.040 ± 0.167	15.4
Lite3 (ours)	0.0152	0.61 ± 0.73	0.041 ± 0.106	10.5
Lite2 (ours)	0.0163	0.66 ± 0.80	0.044 ± 0.109	8.4
Lite1 (ours)	0.0166	0.67 ± 0.82	0.045 ± 0.124	7.7
Lite0 (ours)	0.0186	0.76 ± 1.06	0.053 ± 0.201	6.9
EPFL_cvlab (see footnote ¶¶¶)	0.0215	0.91 ± 1.29	0.073 ± 0.587	89.2
Black et al. [13]	0.0409	— —	— —	6.9
pedro_fairspace [7]	0.0571	2.49 ± 3.02	0.145 ± 0.239	≈500
SLAB_baseline [6]	0.0626	2.62 ± 2.90	0.209 ± 1.133	11.2

and the embedded method proposed in Ref. [13]. Our results clearly indicate how state-of-the art accuracy can be achieved with remarkably less NN parameters. We emphasize that our results referred to the whole SPEC training set through cross validation rather than to the competition test set.
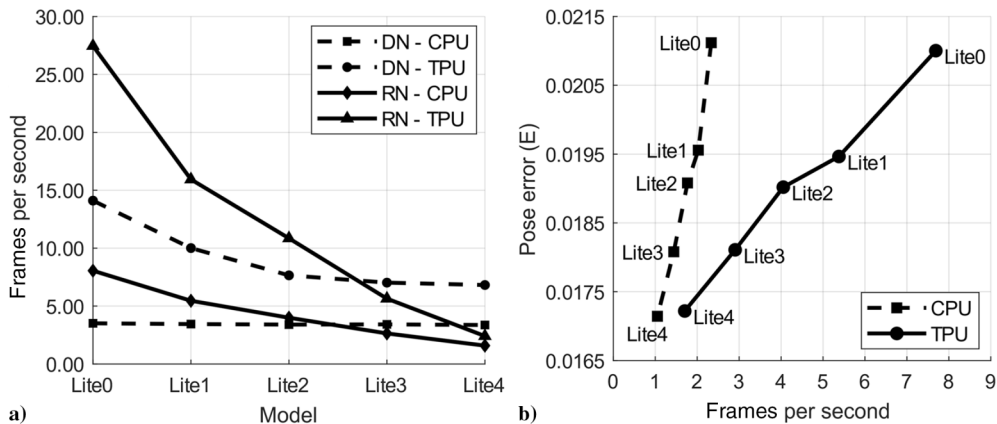
C. Conversion to TensorFlow Lite

Because TF is not optimized for on-device inference, we converted all NNs to TFLite. In this work, we exploited the 2.9.0 version of

Table 4 DN performances

DN	IOU mean	IOU median
TF	0.9332	0.9524
TFLite	0.9332	0.9523
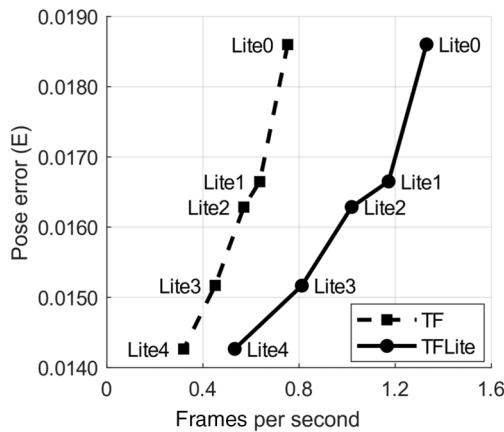
Fig. 3 TF vs TFLite pipelines performances' comparison on CPU.

tflite-runtime with XNNPACK, which is a ML library providing efficient implementations of the most common NN operators for ARM processors. Performance comparisons are reported in Table 4 and Fig. 3, highlighting latency reduction and persistence of accuracy. Note that the frames per second (FPS) data do not include image loading time from memory. Conversion from TF to TFlite allowed increasing the inference speed from 66 to 84% without affecting the accuracy.

D. Quantization: CPU Versus TPU Inference

Quantization, which is needed for deploying the networks on the TPU, allows compressing the file sizes up to the 75%, thereby significantly reducing latency. To limit the accuracy reduction arising from the conversion of floating point weights to integer values, all the models have been retained by applying QAT. A comparison between CPU and TPU performances is provided in Fig. 4, highlighting the superior FPS attainable by the latter, which is up to the 229% higher than with the CPU. In addition, the TPU allows reducing power consumption at inference: our tests revealed that the average absorption drops from the 3 W required by the CPU to 2.2 W for the TPU. This is also reflected by the board heating. Indeed, switching from the CPU to the TPU reduced the average temperature from 85 to 49°C. Notably, QAT was found to limit the pose error increase to only about 18% on average.

E. Performance on CPD

Finally, we tested our TPU pipelines on the CPD. In this case, the quantized DN achieved 0.9219 and 0.9368 mean and median IOUs, respectively. The mean pose error E is similar to that in the SPEED (see Table 5), even though few outliers are present. The worst results are obtained on images characterized by large self-occlusions, especially those where the solar panels hide a large portion of the synthetic aperture radar antenna. Overall, the results are promising, demonstrating the pipeline robustness to the variable geometrical configuration of the target arising from the steerable solar panels' orientation.

V. Conclusions

In this Note, a new photorealistic satellite dataset including steerable solar panels was introduced, and neural models enabling real-time spacecraft pose estimation from monocular images on

Table 5 TPU pipelines' performances on CPD

Quantized RNs	E	e_q, deg	e_t, m
Lite0	0.0193	0.852 ± 3.590	0.200 ± 0.500
Lite1	0.0170	0.748 ± 2.839	0.176 ± 0.583
Lite2	0.0163	0.708 ± 2.595	0.179 ± 0.619
Lite3	0.0158	0.686 ± 2.791	0.170 ± 0.402
Lite4	0.0138	0.598 ± 1.674	0.150 ± 0.373

Fig. 4 Quantized model inference, with CPU vs TPU: a) NNs' runtimes, and b) pipeline performances.

low-power embedded hardware were discussed. The main findings obtained from this work can be summarized as follows:

1) Converting the models to TFLite improved the overall inference speed by about 77% on average.

2) Network quantization, when coupled to quantization-aware training, can significantly improve the accuracy–latency tradeoff, leading on average to a further 80% speedup against an 18% increase of pose error.

3) While being light, the software developed in this paper demonstrate that the presence of movable components does not hinder the average pose estimation accuracy as long as their features are excluded from training.

4) Switching from a CPU to a TPU allows increasing the FPS by a factor of ≈ 2 to 3 (up to 7.7 FPS) while reducing the measured power consumption by 25% (from 3 W down to 2.2 W). As a result, the pipelines perform on par with the state of the art while using extremely lite networks. When evaluated on the CPD, the algorithms exhibit accuracies in line with the SPEED, although with higher sensitivity to occlusions, which deserve further investigation.

Future developments include testing the NNs on real imagery to investigate the domain gap.

Acknowledgment

References

[1] Sharma, S., Ventura, J., and D'Amico, S., "Robust Model-Based Monocular Pose Initialization for Noncooperative Spacecraft Rendezvous," *Journal of Spacecraft and Rockets*, Vol. 55, No. 6, 2018, pp. 1414–1429.
https://doi.org/10.2514/1.A34124

[2] Capuano, V., Cuciniello, G., Pesce, V., Opromolla, R., Sarno, S., Lavagna, M., Grassi, M., Corraro, F., Capuano, G., Tabacco, P., Meta, F., Battagliere, M., and Tuozzi, A., "VINAG: A Highly Integrated System for Autonomous On-Board Absolute and Relative Spacecraft Navigation," *Proceedings of the 4S Symposium*, European Space Agency, 2018, pp. 1–20, https://atpi.eventsair.com/QuickEventWebsitePortal/4s2018/4s/ExtraContent/ContentSubPage?page=1&subPage=1.

[3] Sharma, S., Beierle, C., and D'Amico, S., "Pose Estimation for Non-Cooperative Spacecraft Rendezvous Using Convolutional Neural Networks," *2018 IEEE Aerospace Conference*, IEEE, New York, 2018, pp. 1–12.
https://doi.org/10.1109/AERO.2018.8396425

[4] Sharma, S., and D'Amico, S., "Pose Estimation for Non-Cooperative Rendezvous Using Neural Networks," *2019 AAS/AIAA Astrodynamics Specialist Conference*, American Astronautical Soc. Paper 19-350, Springfield, VA, 2019, pp. 1–20.

[5] Chen, B., Cao, J., Parra, A., and Chin, T. J., "Satellite Pose Estimation with Deep Landmark Regression and Nonlinear Pose Refinement," *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, IEEE, New York, 2019, pp. 2816–2824.
https://doi.org/10.1109/ICCVW.2019.00343

[6] Park, T.H., Sharma, S., and D'Amico, S., "Towards Robust Learning Based Pose Estimation of Noncooperative Spacecraft," *2019 AAS/AIAA Astrodynamics Specialist Conference*, American Astronautical Soc. Paper 19-840, Springfield, VA, 2019, pp. 1–20.

[7] Proença, P. F., and Gao, Y., "Deep Learning for Spacecraft Pose Estimation from Photorealistic Rendering," *2020 IEEE International Conference on Robotics and Automation*, IEEE, New York, 2020, pp. 6007–6013.
https://doi.org/10.1109/ICRA40945.2020.9197244

[8] Hu, Y., Speierer, S., Jakob, W., Fua, P., and Salzmann, M., "Wide-Depth-Range 6D Object Pose Estimation in Space," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, New York, 2021, pp. 15870–15879.
https://doi.org/10.1109/CVPR46437.2021.01561

[9] Wang, Z., Zhang, Z., Sun, X., Li, Z., and Yu, Q., "Revisiting Monocular Satellite Pose Estimation with Transformer," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 58, No. 5, Oct. 2022,
pp. 4279–4294.
https://doi.org/10.1109/TAES.2022.3161605

[10] Piazza, M., Maestrini, M., and Di Lizia, P., "Monocular Relative Pose Estimation Pipeline for Uncooperative Resident Space Objects," *Journal of Aerospace Information Systems*, Vol. 19, No. 9, 2022, pp. 613–632.
https://doi.org/10.2514/1.I011064

[11] Carcagnì, P., Leo, M., Spagnolo, P., Mazzeo, P. L., and Distante, C., "A Lightweight Model for Satellite Pose Estimation," *Image Analysis and Processing–ICIAP 2022, Lecture Notes in Computer Science*, Vol. 13231, Springer, Cham, Switzerland, 2022, pp. 3–14.
https://doi.org/10.1007/978-3-031-06427-2_1

[12] Posso, J., Bois, G., and Savaria, Y., "Mobile-URSONet: An Embeddable Neural Network for Onboard Spacecraft Pose Estimation," *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, New York, 2022, pp. 794–798.

[13] Black, K., Shankar, S., Fonseka, D., Deutsch, J., Dhir, A., and Akella, M. R., "Real-Time, Flight-Ready, Non-Cooperative Spacecraft Pose Estimation Using Monocular Imagery," *31st AAS/AIAA Space Flight Mechanics Meeting*, American Astronautical Soc. Paper 21-283, Springfield, VA, 2021, pp. 1–16.

[14] Schubert, C., Black, K., Fonseka, D., Dhir, A., Deutsch, J., Dhamani, N., Martin, G., and Akella, M., "A Pipeline for Vision-Based On-Orbit Proximity Operations Using Deep Learning and Synthetic Imagery," *2021 IEEE Aerospace Conference*, IEEE, New York, 2021, pp. 1–15.
https://doi.org/10.1109/AERO50100.2021.9438232

[15] Pasqualetto Cassinis, L., Fonod, R., and Gill, E. "Review of the Robustness and Applicability of Monocular Pose Estimation Systems for Relative Navigation with an Uncooperative Spacecraft," *Progress in Aerospace Sciences*, Vol. 110, Oct. 2019, Paper 100548.
https://doi.org/10.1016/j.paerosci.2019.05.008

[16] Song, J., Rondao, D., and Aouf, N. "Deep Learning-Based Spacecraft Relative Navigation Methods: A Survey," *Acta Astronautica*, Vol. 191, Feb. 2022, pp. 22–40.
https://doi.org/10.1016/j.actaastro.2021.10.025

[17] Kisantal, M., Sharma, S., Park, T. H., Izzo, D., Märtens, M., and D'Amico, S., "Satellite Pose Estimation Challenge: Dataset, Competition Design, and Results," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 56, No. 5, 2020, pp. 4083–4098.
https://doi.org/10.1109/TAES.2020.2989063

[18] Goodwill, J., Crum, G., MacKinnon, J., Brewer, C., Monaghan, M., Wise, T., and Wilson, C., "NASA SpaceCube Edge TPU SmallSat Card for Autonomous Operations and Onboard Science-Data Analysis," *Proceedings of the Small Satellite Conference*, No. SSC21-VII-08, Univ. of Utah, 2021, https://digitalcommons.usu.edu/smallsat/.

[19] Xiong, Y., Liu, H., Gupta, S., Akin, B., Bender, G., Wang, Y., Kindermans, P.-J., Tan, M., Singh, V., and Chen, B., "MobileDets: Searching for Object Detection Architectures for Mobile Accelerators," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, New York, 2021, pp. 3824–3833.
https://doi.org/10.1109/CVPR46437.2021.00382

[20] Tan, M., and Le, Q., "EfficientNnet: Rethinking Model Scaling for Convolutional Neural Networks," *International Conference on Machine Learning*, Univ. of Cambridge, Cambridge, 2019, pp. 6105–6114, http://proceedings.mlr.press/v97/.

[21] Lepetit, V., Moreno-Noguer, F., and Fua, P., "EPnP: An Accurate $O(n)$ Solution to the PnP Problem," *International Journal of Computer Vision*, Vol. 81, No. 2, 2009, pp. 155–166.
https://doi.org/10.1007/s11263-008-0152-6

[22] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L., "Microsoft COCO: Common Objects in Context," *Computer Vision—ECCV 2014, Lecture Notes in Computer Science*, Vol. 13231, Springer, New York, 2014, pp. 740–755.
https://doi.org/10.1007/978-3-319-10602-1_48

[23] Kingma, P. M., and Ba, J., "Adam: A Method for Stochastic Optimization," *3rd International Conference on Learning Representations*, San Diego, CA, 2015, https://arxiv.org/abs/1412.6980.

[24] Krizhevsky, A., Sutskever, I., and Hinton, G. E., "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems*, Vol. 25, No. 2, 2012, pp. 1106–1114.
https://doi.org/10.1145/3065386

I. I. Hussein
Associate Editor