POLITECNICO
MILANO 1863

School of Industrial and Information Engineering
Department of Aerospace Science and Technology

Master of Science in
Space Engineering

Deep Learning-Based Monocular Relative Pose Estimation of Uncooperative Spacecraft

Advisor:
Pierluigi Di Lizia

Candidate:
Massimo Piazza
(ID: 919920)

Co-advisor:
Michele Maestrini

November 25, 2020

Abstract

The aim of this dissertation is to present the design and validation of a deep learning-based pipeline for estimating the pose of an uncooperative target spacecraft, from a single grayscale monocular image.

The possibility of enabling autonomous vision-based relative navigation, in close proximity to a non-cooperative space object, has recently gained remarkable interest in the space industry. In particular, such a technology would be especially appealing in an on-orbit servicing scenario as well as for Active Debris Removal (ADR) missions.

The use of a simple camera, compared to more complex sensors such as a LiDAR, has numerous advantages in terms of lower mass, volume and power requirements. This would clearly translate into a substantially cheaper chaser spacecraft, at the expense of increased complexity of the image processing algorithms.

The Relative Pose Estimation Pipeline (RPEP) proposed in this work leverages state-of-the art Convolutional Neural Network (CNN) architectures to detect the features of the target spacecraft from a single monocular image. Specifically, the overall pipeline is composed of three main subsystems. The input image is first of all processed using an object detection CNN that localizes the portion of the image enclosing our target, i.e. the Bounding Box. This is followed by a second CNN that regresses the location of semantic keypoints of the spacecraft. Eventually, a geometric optimization algorithm exploits the detected keypoint locations to solve for the final relative pose, based on the knowledge of camera intrinsics and of a wireframe model of the target satellite.

The Spacecraft PosE Estimation Dataset (SPEED), a collection of 15300 images of the Tango spacecraft released by the Space rendezvous LABoratory (SLAB), has been used for training the Artificial Neural Networks employed in our pipeline, as well as for evaluating performance and estimation uncertainty.

The RPEP presented in this dissertation guarantees on SPEED a centimeter-level position accuracy and degree-level attitude accuracy, along with considerable robustness to changes in lighting conditions and in the background. In addition, our architecture also showed to generalize well to actual images, despite having exclusively been exposed to synthetic imagery during the training of CNNs.

Abstract (Italian version)

L'obiettivo di questa tesi è di illustrare il processo di sviluppo e validazione di una pipeline basata su tecniche di deep learning, per la stima della posa di un satellite non-cooperativo, a partire da una semplice immagine monoculare in scala di grigi.

La possibilità di implementare un sistema autonomo di navigazione relativa basato su input visivi, in prossimità di un oggetto spaziale non-cooperativo, ha recentemente suscitato un interesse degno di nota nell'industria spaziale. In particolare, una simile tecnologia diverrebbe di particolare importanza in scenari quali ad esempio la manutenzione di un satellite già in orbita o in missioni per la rimozione attiva dei detriti spaziali (ADR).

L'utilizzo di una fotocamera, rispetto a sensori di maggiore complessità come i LiDAR, presenta diversi vantaggi in termini di riduzione di massa, volume e potenza richiesta. Tutto ciò si traduce chiaramente in un notevole risparmio economico per il satellite "chaser", che tuttavia si contrappone ad una maggiore complessità degli algoritmi impiegati per processare la sequenza di immagini raccolte da un sensore visivo.

La pipeline per la stima della posa relativa qui proposta è in grado di identificare i punti caratteristici del satellite "target" da una singola immagine monoculare, sfruttando dei Convolutional Neural Networks (CNNs) che rappresentano l'attuale stato dell'arte nel campo della computer vision. Nello specifico, l'architettura complessiva è costituita da tre sottosistemi principali. Il primo step è affidato ad un CNN che identifica la porzione di immagine che racchiude al suo interno il target di interesse, ossia, identificando la cosiddetta Bounding Box. A questo punto, segue un secondo CNN addestrato a rilevare nella Bounding Box la posizione di alcuni punti caratteristici del satellite. Infine, un algoritmo di ottimizzazione geometrica sfrutta i punti appena identificati per convergere alla posa che meglio riflette tale posizionamento, tutto ciò basandosi sulla conoscenza delle caratteristiche del nostro sensore visivo e del modello 3D del satellite target.

Lo Spacecraft PosE Estimation Dataset (SPEED), una raccolta di 15300 immagini del satellite Tango rilasciata dallo Space rendezvous LABoratory (SLAB), è stata utilizzata per l'addestramento delle reti neurali impiegate nella nostra pipeline, ma anche per valutare la performance e l'incertezza della stima che ne risulta.

L'architettura proposta in questa tesi ha dimostrato un livello di precisione centimetrica per quanto riguarda la posizione relativa ed un errore di assetto nell'ordine del grado, insieme ad una considerevole robustezza a variazioni delle condizioni di illuminazione e dello sfondo dell'immagine. Inoltre, è stato provato che simili prestazioni sono garantite anche su immagini reali, benché l'addestramento dei CNN sia stato eseguito processando esclusivamente immagini sintetiche.

Acknowledgements

First and foremost, I would like to express my gratitude to my advisor Pierluigi Di Lizia and to my co-advisor Michele Maestrini, for their constant support, trust and interest for the work I carried out during the last four months. I would describe this time as the most motivating period of my academic career, yet one of the most challenging. It was both a privilege and an opportunity to rely on your knowledge and willingness to help me out.

I would also like to thank OHB Sweden for providing me with real data from the Prisma mission, that will allow me to further research the topics here presented, also after my graduation.

Among the people who should be credited for helping me get to where I am now, graduating as a Space Engineer, I must surely mention all my fellow classmates, starting from the ones who have been there since the very first day. Francesca, Gregorio, Gustavo, Matteo, thank you all. And thank you to all the ones that followed: Giorgio, Cesare, Valerio, Chiara, the rest of the Padova-crew, Marzio, the MARS-PENGUIN team and all the ones I did not explicitly mention, you are part of this too.

Thank you Milan for making me feel home since the very beginning of this journey which is now coming to an end, for all the experiences, opportunities and new people I came across during these years, both inside and outside university. I will miss all of this and all of you.

And last but not least, I clearly have to thank my family for their constant presence, albeit away from my hometown, and for their immense support during the five years that brought me here.

We become the people we decide to surround ourselves with, again, thank you all.

I could either watch it happen
or be a part of it.

Contents

List of Figures

List of Tables

Acronyms

AB Anchor Box 18, 19
ADAM ADAptive Momentum 6, 15, 16, 34
ADR Active Debris Removal i, ii, 2
ANN Artificial Neural Network i, 11, 15, 17, 54
AP Average Precision 19, 20, 22, 28, 31, 34, 35, 60

BB Bounding Box i, ii, 16–20, 22, 28–30, 33, 36–38, 41, 42
BGD Batch Gradient Descent 15

CNN Convolutional Neural Network i, ii, 4–6, 10–14, 16–18, 20, 30, 32, 33, 35, 52, 60, 62
CPU Central Processing Unit 53

EPnP Efficient Perspective-n-Point 4, 6, 24, 25, 28, 37–39, 43, 53, 54, 58, 60
ESA European Space Agency 5, 7, 9, 44, 60

FLOP FLoating-Point Operation 28, 34
FN False Negative 20
FoV Field of View 7, 33, 41, 42
FP False Positive 20
FPS Frames Per Second 28, 54

GDM Gradient Descent Method 15, 25
GNM Gauss-Newton Method 4, 24, 25
GPU Graphics Processing Unit 14, 34, 54
GT Ground Truth 9, 22, 30, 34, 40, 41, 52, 56

HRNet High-Resolution Network vi, viii, 6, 21, 33, 34, 60

IoU Intersection over Union vi, 18–20, 22, 31, 60

LiDAR Light Detection And Ranging i, ii, 1
LMM Levenberg-Marquardt Method 7, 25, 28, 37, 53, 54, 60
LRN Landmark Regression Network 10, 28, 29, 33–37, 43, 52–56, 58, 60

mAP mean Average Precision 20

MEV-1 Mission Extension Vehicle-1 2
ML Machine Learning 7, 11, 12, 14, 45, 61
MNPE Median Normalized Pose Error 43, 46, 50, 53, 54, 61

NRM Newton-Raphson Method 24

OKS Object Keypoint Similarity vi, 22, 35, 60

PnP Perspective-n-Point 7, 10, 22, 24, 25, 60

RANSAC RANdom SAmple Consensus 7
ReLU Rectified Linear Unit 13
RMSProp Root Mean Square Propagation 15, 16
RoI Region of Interest vi, viii, 6, 7, 28–34, 37–39, 41, 45, 50, 56, 58, 60, 61
RPEP Relative Pose Estimation Pipeline i, viii, 10, 27, 28, 36, 40, 44, 46, 49, 60, 61

S/C SpaceCraft 1, 2, 5, 16, 22, 28, 29, 33, 37, 38, 40, 46, 52, 60, 62
SGD Stochastic Gradient Descent 15, 30
SLAB Space rendezvous LABoratory i, ii, vi, 5–7, 9, 29, 31, 41, 42, 44, 46, 50–52, 54, 60, 61
SLN Spacecraft Localization Network vi, viii, 10, 28, 31–33, 36, 37, 52–56, 60
SPEED Spacecraft PosE Estimation Dataset i, ii, vi, viii, 2, 7, 8, 10, 29, 34, 40, 44, 45, 60, 61
SSD Single Shot Detector 17
SVD Sharma-Ventura-D'Amico vi, 3, 4
SVM Support Vector Machine 17

TP True Positive 19, 20
TRON Testbed for Rendezvous and Optical Navigation 7, 8

WGE Weak Gradient Elimination 3

YOLO You Only Look Once viii, 5, 6, 17–19, 28, 31, 33, 60

1 Chapter

Introduction

1.1 Problem statement & motivation

The problem that will be tackled in this dissertation is that of estimating the relative pose of an uncooperative spacecraft (S/C) from a single grayscale monocular image, in a close-proximity operations scenario. The term "uncooperative" is here referred to a situation in which the target S/C is not equipped with supportive means (e.g. light-emitting markers) nor is capable of establishing a communication link. The satellite is modeled as a rigid body, which means that its six-dimensional pose space is defined in terms of 3 translation components and 3 attitude components, relative to the chaser S/C.

For estimating the pose of an uncooperative spacecraft relative to another satellite, two main approaches are possible. The motion of a space object might be in principle estimated using ground-based radar facilities. However, such an estimate would be affected by significant uncertainty and its availability would depend on the target's visibility from ground. This is clearly unsuitable for close-proximity operations between two spacecrafts. The second approach consists in estimating the pose of the target directly onboard the chaser S/C, by exclusively relying on the sensors available on the latter. This currently represents the only strategy that is suitable for close-proximity operations.

A possible choice to achieve onboard pose estimation may be the use of LiDAR and/or stereo camera sensors, which, nevertheless, can be extremely expensive and represent a substantial contribution to the mass and power budgets of the S/C. In contrast, monocular cameras are characterized by a lower complexity and their use for autonomous relative navigation would translate into significant savings in terms of cost, mass and power requirements. All these benefits come at the expense of very high complexity of the image processing algorithms. In addition, monocular sensors are characterized by a weaker robustness to lighting conditions and variable backgrounds, compared to a LiDAR. This aspect is particularly challenging, given the low signal-to-noise ratio that characterizes spaceborne optical images.

The estimation of the relative pose by means of a cheap, low-mass and low-power monocular camera is an appealing possibility, especially within the framework of on-orbit servicing missions. Among the missions of this kind, that are slated for launch during the next few years, an example might be NASA's *Restore-L* mission [Ree16], whose launch date is currently set for December 2023, along with the commercial servicing programs proposed by companies like Infinite Orbits and Astroscale. In addition, the first ever

Active Debris Removal (ADR) mission, *ClearSpace-1* [ESA], is expected to launch in 2025.

It is then clear that the ability to accurately estimate the pose of an uncooperative S/C, by relying on hardware with minimal complexity, represents a key enabling technology in all the aforementioned scenarios. For instance, a life-extension mission in which the servicer spacecraft has the only purpose of re-fueling an old satellite, would be economically viable only by making the mission extremely cheap, thus allowing a profit margin to the service provider while still being advantageous for the customer.

The feasibility of a life-extension mission has also been recently demonstrated by the Mission Extension Vehicle-1 (MEV-1), which, in February 2020, rendezvoused with the Intelsat 901 satellite. The latter is a telecommunications satellite, launched in 2001, that was nearing the end of its mission due to fuel consumption. After the docking between the two spacecrafts, MEV-1 started performing station-keeping with its own thrusters for the old satellite, thus extending the operational lifetime of Intelsat 901 by five more years [Red20].

We will now continue our discussion by presenting the state-of-the-art techniques for S/C pose estimation, in Section 1.2. In Section 1.3, we will successively introduce the Spacecraft PosE Estimation Dataset (SPEED), which has been used for validating the pose estimation algorithms proposed in this dissertation. And ultimately, at the end of this introductory chapter, we will provide the outline of the remainder of this work, in Section 1.4.

1.2 State-of-the-art

In this section we will present a brief survey of the state-of-the-art techniques used for estimating the pose of a spacecraft from a monocular image.

Typically, all these techniques make use of an image-processing subsystem that identifies the position in the image frame of certain semantic features of the S/C. This is followed by a pose solver consisting in a geometric optimization subsystem, that fits a known 3D model of the target S/C to the features matched in the image.

The aforementioned routine shall then be embedded in a navigation filter, in order to be used in an actual rendezvous scenario, during which the inter-spacecraft distance ranges from tens of meters to a few centimeters.

Depending on the approach adopted for image processing, two main classes of monocular pose estimation methods may be identified. Feature-based methods seek for correspondences between edges detected in the image and line segments of the known wireframe model of the spacecraft. Deep learning-based approaches, instead, make use of deep neural networks to either regress the location of semantic keypoints or directly estimate the pose.

1.2.1 Feature-based pose estimation

The traditional computer vision techniques for estimating the pose of an object (from a single perspective view) resort to feature-based methods.

M. Dhome is among the pioneers in this field and in 1989 he proposed one of the first feature-based solutions to the pose estimation problem [Dho89]. The underlying idea is

that of trying to match all possible sets of linear-ridge triplets of the object model with the triplets of edges detected in the image, in order to find the attitude that is consistent with the perspective projection of a triplet of edges. The geometrical transformation corresponding to a given choice of triplets in the image is computed by solving an 8th degree equation.

In 2014, S. D'Amico proposed a monocular vision-based navigation system [DAm14] that, for the first time, enabled proximity navigation with respect to a completely uncooperative space object. Indeed, unlike previous work, neither supportive means (e.g. light-emitting markers) nor a-priori knowledge of the target's pose are required. The pipeline consists of four main subsystems, whose key steps are reported here below.

i) The image processing stage is aimed at extracting straight-line segments and consists of three main steps: a.) low pass filtering; b.) Canny-edge detection [Can86]; c.) Hough transform [Dud72]. This stage also involves the tuning of several hyperparameters.

ii) Perceptual grouping [Low12] is a technique inspired by human vision and is applied for organizing the detected segments into higher-level perceptually relevant structures, that are known to be part of the model.

iii) Pose initialization is required in order to obtain a rough estimate of the pose to be used as the initial guess. The adopted approach is the one described in [Low87].

iv) Pose refinement, consisting in: a.) Newton-Raphson optimization,[(1)] starting from the previously computed initial guess; b.) model matching; c.) least-squares fitting, which yields the final pose.

The method has been successfully tested on actual flight imagery captured during the PRISMA mission [Bod12]. However, two fundamental limitations are highlighted in [DAm14]: the excessive computational cost, that prevents real-time usage on spaceborne hardware, and the lack of robustness to changes in lighting conditions and in the background.

In 2018, S. Sharma, J. Ventura and S. D'Amico proposed their Sharma-Ventura-D'Amico (SVD) feature-based method [Sha18b]. The method has been tested on actual flight imagery from the PRISMA mission and, compared to previous work, it proved enhanced computational efficiency and superior robustness to the background. The latter is achieved thanks to the fusion of the Weak Gradient Elimination (WGE) technique[(2)] with state-of-the-art edge detectors.

The SVD method is composed of two subsystems, which are briefly described here below.

i) Image processing. As it can be seen from Figure 1.1, after applying a Gaussian filter to attenuate noise, the image undergoes processing along two parallel streams, whose outputs are then merged. In the first stream, WGE is employed, followed by the Hough transform. In the second stream, the Sobel edge detector is applied [Sob68],

[(1)] the objective is that of iteratively minimizing the fit error between the projection of the 3D model onto the image plane (according to the estimated pose) and the detected features in the image frame

[(2)] it basically eliminates gradients where they are weak and highlights regions where gradients are strong: this is of particular importance whenever Earth is present in the background

again followed by a Hough transform. After merging the results of the two streams, duplicate detections will have to be discarded. Eventually, the detected segments are organized into higher-level features, which allows to drastically reduce the search space of possible correspondences, thus translating into reduced computational burden.

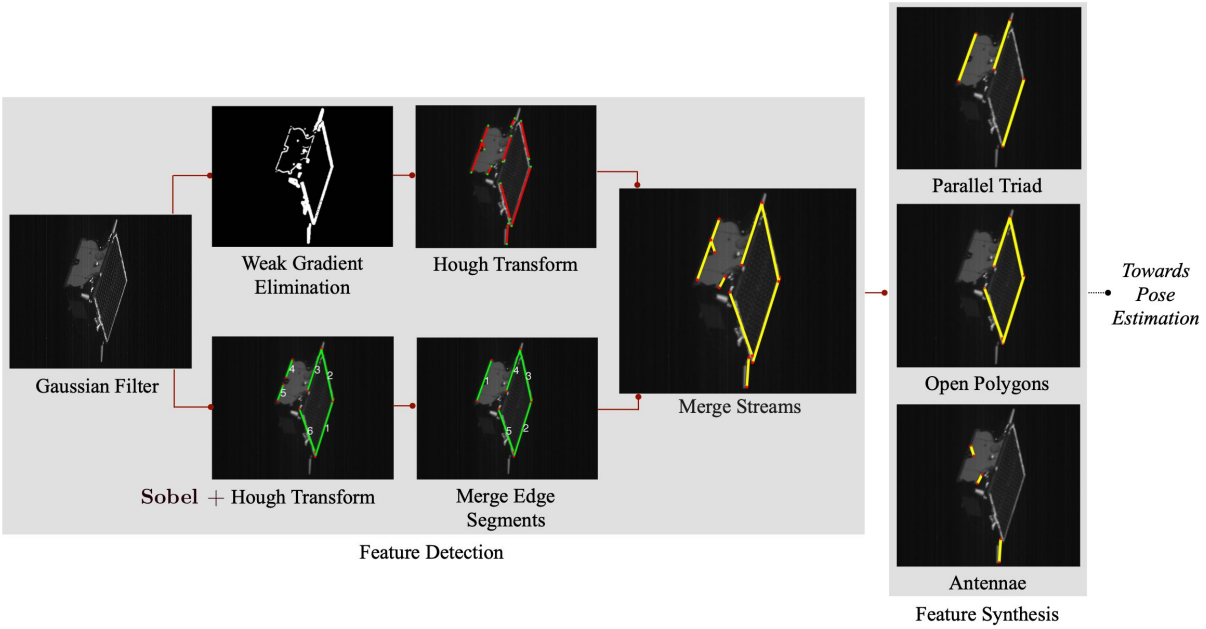


Figure 1.1: *Image processing subsystem of the SVD method (credits to: [Sha18b])*

ii) Pose estimation. For each combination of possible feature correspondences, the 3D-2D perspective projection equations are initially solved for the pose using the EPnP method [Lep09]. The pose candidates with the lowest reprojection error are then iteratively refined using the Gauss-Newton Method (GNM).

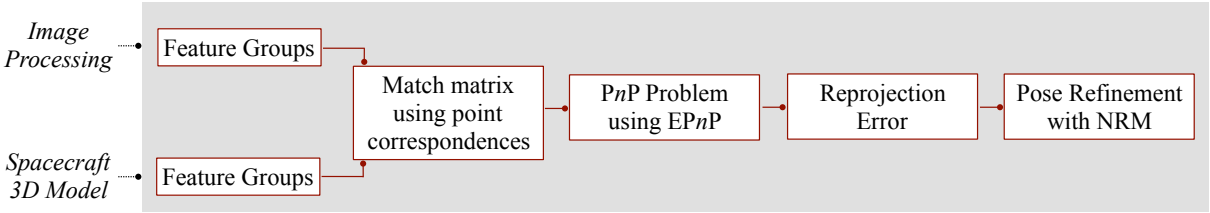


Figure 1.2: *Pose estimation subsystem of the SVD method (credits to: [Sha18b])*

1.2.2 Deep learning-based pose estimation

Deep learning-based methods make use of a Convolutional Neural Network (CNN) pipeline whose job, depending on the approach, may either consist in:

- regressing the position in the image frame of predefined keypoints, that later become the input of a pose solver [Par19; Che19]
- directly estimating the pose, according to one of the following formulations of the pose estimation problem:

- regression problem [Pro20]
- classification problem, which requires a sufficiently dense discretization of the pose space [Sha18a; Sha17]
- hybrid classification-regression problem [Sha20]

We will now present the architecture of two of the most promising deep learning-based pose estimation pipelines to date, that have been developed in 2019, within the context of the SLAB/ESA Pose Estimation Challenge [Kel]. They are of particular importance, since the architecture proposed in this work has been mainly inspired by these two pipelines, and shares several features with both of them.

SLAB baseline (Pose Estimation Challenge) - Park et al.

The Space rendezvous LABoratory (SLAB), besides organizing the competition and releasing the related dataset, also proposed their own baseline solution [Par19], which eventually scored 4th place.

The proposed architecture is depicted in Figure 1.3 and consists of three main subsystems.

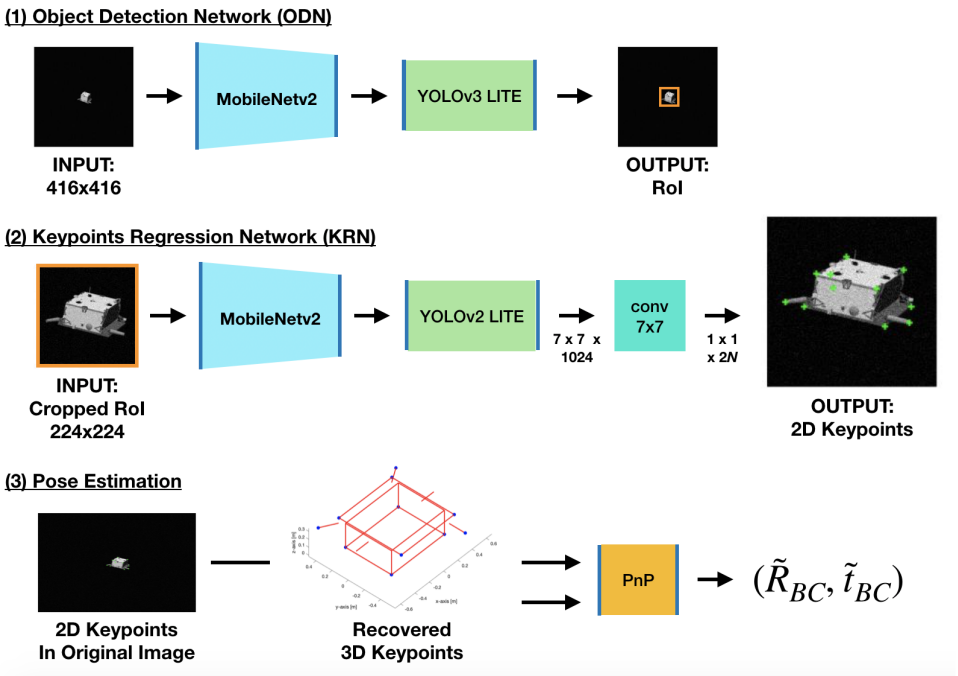


Figure 1.3: *Architecture of SLAB's pose estimation pipeline (credits to: [Par19])*

i) Object Detection Network. A CNN based on the `YOLOv3` [Red18] architecture is trained to detect the bounding box enclosing the S/C, from a 416×416 input image. In order to reduce the number of parameters in the network and thus allow real-time inference, two fundamental changes are applied to the original `YOLO` architecture. First of all, the `Darknet-53` backbone has been replaced by `MobileNetv2` [San18]. Secondly, conventional convolution operations have been replaced by depth-wise separable convolutions, i.e. depth-wise convolutions followed by point-wise convolutions.

ii) Keypoint Regression Network. The network is trained to regress the location in the image of semantic keypoints of the satellite. It exploits an architecture quite similar

to the one previously described, except for the fact that `YOLOv2` is used instead of `v3`. In order to maintain a reasonable number of parameters `MobileNetv2` is still used, together with depth-wise separable convolutions.
The RoI detected by the Object Detection Network is resized to 214×214 and fed into this network.

iii) Pose Estimation. It receives as input the 2D locations of the keypoints detected by the previous CNN and, using the EPnP algorithm [Lep09], this subsystem outputs the final pose estimate of the spacecraft.

Both neural networks of the pipeline are trained for 200 epochs. The initial learning rate is set to 5×10^{-4} and it is scheduled to halve every 50 epochs. ADAM optimization is used, with momentum equal to 0.9 and a weight decay of 5×10^{-5}.

A key aspect of the training procedure is that the original dataset of synthetic spacecraft imagery is augmented by randomizing the texture of the satellite's surface, by making use of the Neural Style Transfer technique. The underlying idea behind this choice is that of forcing the network to learn the global shape of the satellite, rather than focusing on local texture. As a result, the pose estimation pipeline trained on a synthetic dataset improves its capability to generalize to spaceborne images, despite having never been exposed to actual flight imagery during training.

UniAdelaide solution (Pose Estimation Challenge) - Chan et al.

The pose estimation pipeline proposed by the team from University of Adelaide [Che19] scored 1st place in the competition. The key concept behind their superior performance is the use of the `HRNet` architecture [Sun19a] which, in contrast to conventional CNNs, connects high-to-low resolution subnetworks in parallel rather than in series. This means that the initial high-resolution representation is maintained throughout the whole inference process, thus resulting into extremely high accuracy of the regressed landmark positions.

The global architecture of the pipeline is structured into three main subsystems and is similar to that of SLAB's baseline, as it can be seen from Figure 1.4.

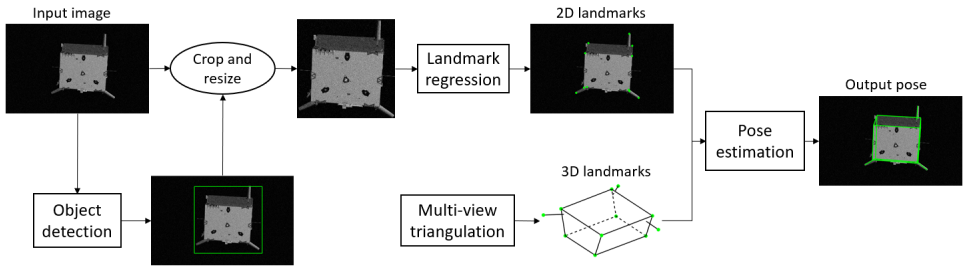


Figure 1.4: *Architecture of University of Adelaide's pose estimation pipeline (credits to: [Che19])*

i) Object Detection Network. It is based on a region-proposal approach, using `HRNet18`[(3)] as backbone in the `Faster R-CNN` framework [Ren15]. The network outputs the bounding box corresponding to the spacecraft's location.

[(3)] the number 18 indicates the version of the network having 18 channels in the highest-resolution subnetworks in the last three stages

ii) Landmark Regression Network. The cropped RoI is resized to 768×768 and fed into the `HRNet32` network, which regresses a separate heatmap for each of the selected landmarks to detect.

iii) Pose Estimation. The best pose fit is iteratively computed by solving a non-linear least-squares problem, using the Levenberg-Marquardt Method (LMM). In addition, the LMM iterations are coupled with a Simulated Annealing scheme to progressively remove outliers. The initial pose guess is computed using a RANSAC-fashion PnP solver implemented in `Matlab`.

1.3 Spacecraft Pose Estimation Dataset

The Spacecraft PosE Estimation Dataset (SPEED) consists of 15300 grayscale images of the Tango spacecraft, along with the corresponding pose labels. 15000 of these images have been generated synthetically, while the remaining 300 are actual images of a 1:1 mock-up, captured under high-fidelity illumination conditions at the TRON facility. SPEED is the first and only publicly available ML dataset for spacecraft pose estimation and has been released in February 2019, with the start of the Pose Estimation Challenge organized by SLAB in collaboration with ESA (Feb-Jul 2019). The camera model used for rendering the synthetic images is that of the actual camera employed for capturing the 300 images of the mock-up. The related parameters are reported in Table 1.1.

Table 1.1: *SPEED camera model*

Parameter	Value
Resolution ($N_u \times N_v$)	1920×1200 px
Focal length f	17.6 mm
Pixel pitch ($\rho_u \equiv \rho_v$)	5.86 μm/px
Horizontal FoV	35.452°
Vertical FoV	22.595°

1.3.1 Synthetic images

All the photo-realistic renderings of Tango are generated using an OpenGL-based pipeline. In half of these 15k images, random Earth images are inserted in the background of the satellite. The Earth backgrounds are obtained by cropping actual images captured by the Himawari-8 geostationary weather satellite: a distribution of 72 images taken over 12 hours (10 minutes apart from each other). In all images with Earth background, the illumination conditions used for rendering Tango are consistent with those in the image of the Earth disk.
In addition, Gaussian blurring ($\sigma = 1$) and Gaussian white noise ($\sigma^2 = 0.0022$) are eventually superimposed to all images.

The relative position vector for each generated image is obtained by separately sampling the total distance and the bearing angles, resulting in the distribution in Figure 1.5:

- total distance $\sim \mathcal{N}(\mu = 3, \sigma = 10)$ m (any value either < 3 m or > 50 m is rejected)
- bearing angles $\sim \mathcal{N}(\mu = [u_0, v_0], \sigma = [5u_0, 5v_0])$ px (where $u_0 = \frac{N_u}{2}$, $v_0 = \frac{N_v}{2}$ denote the camera principal point)

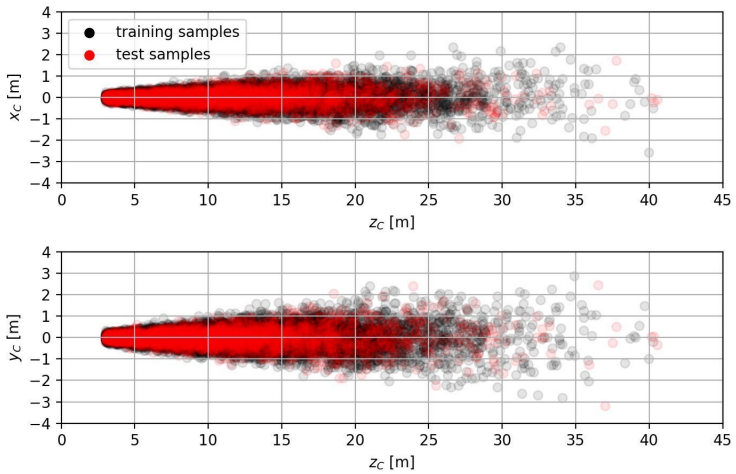


Figure 1.5: *Relative distance distribution of SPEED (credits to: [Kis20])*

The attitude distribution corresponds to uniformly distributed random rotations, as it can be seen from Figure 1.6.

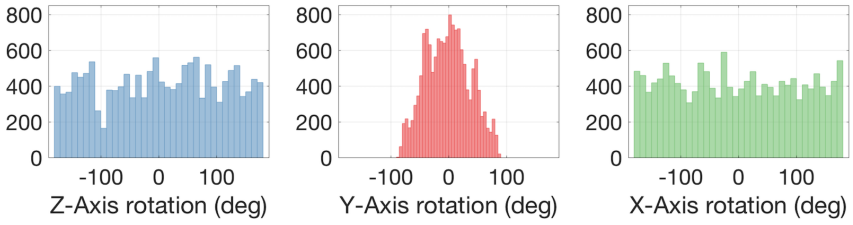


Figure 1.6: *Relative attitude distribution of SPEED, parametrized using Euler angles (credits to: [Kis20])*

1.3.2 Actual mock-up images

Given the physical constraints of the TRON facility, the distance distribution of real images is very limited compared to synthetic ones and ranges between 2.8 and 4.7 m. In addition, unlike the synthetic image source (for which pose labels are automatically annotated), the accurate determination of "ground truth" relative poses of the mock-up requires a complex calibrated motion capture system. The facility includes 10 Vicon Vero v1.3x cameras that track several infrared reflective markers placed onto Tango's body and in the robotic arm that holds the camera (the one that collects the 300 images in the dataset). High accuracy light sources are present in order to mimic sunlight and Earth's albedo.

1.3.3 SLAB/ESA challenge

The SLAB/ESA Pose Estimation Challenge ended in July 2019 and its leaderboard is based on the evaluation of a single scalar error metric. For convenience, we will refer to it as the "SLAB score". Although this metric is separately computed for both the real and synthetic datasets, participants are exclusively ranked based on the performance on synthetic images.

The SLAB score of each individual image is determined as the sum of a translation error and a rotation error, as defined in Equation (1.1). The translation error is computed as the norm of the difference between the Ground Truth (GT) relative distance vector $\mathbf{r}$ and the estimated one $\hat{\mathbf{r}}$, normalized with respect to the GT distance. The rotation error is defined as the quaternion error between the GT relative attitude and the corresponding estimate.

$$e_{\text{SLAB}}^{(\text{i})} = \underbrace{\frac{\left\|\mathbf{r}^{(\text{i})} - \hat{\mathbf{r}}^{(\text{i})}\right\|}{\left\|\mathbf{r}^{(\text{i})}\right\|}}_{e_t^{(\text{i})}} + \underbrace{2 \cdot \arccos\left|\mathbf{q}^{(\text{i})} \cdot \hat{\mathbf{q}}^{(\text{i})}\right|}_{E_q^{(\text{i})}} \tag{1.1}$$

The overall score is then just the average over all the N test images.

$$e_{\text{SLAB}} = \frac{1}{N} \sum_{i=1}^{N} e_{\text{SLAB}}^{(\text{i})} \tag{1.2}$$

A total of 48 teams participated in the competition and three of them succeeded at outperforming SLAB's baseline. `UniAdelaide` obtained the best score on the synthetic test set, hence winning the competition. `EPFL_cvlab` scored a second place, yet achieving the highest score on the real test set.

The overall outcome of the competition is described in [Kis20] and summarized in Table 1.2.

Table 1.2: *Leaderboard of the top 10 teams*

Team name	**Synthetic images score**	**Real images score**	**Translation error** [m] ($\mu \pm \sigma$)	**Quaternion error** [deg] ($\mu \pm \sigma$)
1. `UniAdelaide`	**0.0094**	0.3752	0.032 ± 0.095	0.41 ± 1.50
2. `EPFL_cvlab`	0.0215	**0.1140**	0.073 ± 0.587	0.91 ± 1.29
3. `pedro_fairspace`	0.0571	0.1555	0.145 ± 0.239	2.49 ± 3.02
4. `stanford_slab`	0.0626	0.3951	0.209 ± 1.133	2.62 ± 2.90
5. `Team_Platypus`	0.0703	1.7201	0.221 ± 0.530	3.11 ± 4.31
6. `motokimura1`	0.0758	0.6011	0.259 ± 0.598	3.28 ± 3.56
7. `Magpies`	0.1393	1.2659	0.314 ± 0.568	6.25 ± 13.21
8. `GabrielA`	0.2423	2.6209	0.318 ± 0.323	12.03 ± 12.87
9. `stainsby`	0.3711	5.0004	0.714 ± 1.012	17.75 ± 22.01
10. `VSI_Feeney`	0.4658	1.5993	0.734 ± 1.273	23.42 ± 33.57

1.3.4 Dataset re-partitioning

As of November 2020, the Pose Estimation Challenge is still virtually running in post-mortem mode, with a separate leaderboard for all the results submitted after July 2019. For this reason, in order to maintain the integrity of the post-mortem competition, the ground truth labels of the test set have not been publicly disclosed.

Given the purposes of this dissertation, which include a detailed evaluation of both performance and uncertainty of our pose estimation pipeline, it was clearly of paramount importance to be provided with test labels. It was therefore decided to perform a re-partitioning of the original training set (for which pose labels are publicly available) into three new training, validation and test sets.

In particular, the original 12k training examples were first of all randomly shuffled and then divided into:

- 7680 training images (64%)
- 1920 validation images (16%)
- 4800 test images (24%)

1.4 Outline

This dissertation is divided into 5 chapters.

In Chapter 2, which follows here below, we will introduce the underlying mathematical preliminaries that are necessary to understand the architecture of the Relative Pose Estimation Pipeline (RPEP) proposed in this work. In particular, since we will resort to a deep learning-based image processing, the fundamentals of Convolutional Neural Networks (CNNs) are first of all introduced, with a particular focus on the state-of-the-art architectures developed for object detection and landmark regression. In addition, the Perspective-n-Point (PnP) problem is introduced right after, along with the available approaches for its solution.

In Chapter 3, the architecture of our RPEP is described in detail, with a separate section for each of the three subsystems composing the pipeline. In particular: Section 3.1 presents our Spacecraft Localization Network (SLN), which is the first subsystem that analyzes the input image; Section 3.2 introduces the Landmark Regression Network (LRN); Section 3.3 describes the third and last subsystem of the pipeline, which is the pose solver.

In Chapter 4 we analyzed the performance achieved by the architecture presented in Chapter 3 on our synthetic test set of SPEED images. Besides evaluating errors and uncertainty, we identified the most important characteristics of the input image that correlate with estimation accuracy. In addition, an intuitive visualization of the pose estimate, along with intermediate processing steps, is provided at the end of the chapter.

In Chapter 5, we summarized the results of the present work and issued some recommendations for future research in this field.

2 Chapter

Mathematical background

2.1 Convolutional Neural Networks

Machine Learning (ML) is a field of Artificial Intelligence, whose techniques give a computer program the ability to learn from data, thus building a mathematical model based on such "training data", that is capable of making predictions without being explicitly programmed to do so.

Any ML algorithm requires building a model to train on data. Nowadays, the most common family of models for ML applications is represented by Artificial Neural Networks (ANNs), whose working principle vaguely mimics the human brain.[1]

Among ANNs, Convolutional Neural Networks (CNNs) constitute a class of models that is particularly suited for computer vision applications, i.e. for processing image data. The advantage of CNNs is that they allow to drastically reduce the computational resources required for processing an image, compared to traditional ANNs. In particular, this is achieved by leveraging on two fundamental operations: convolution and pooling.

2.1.1 Architecture of a CNN

The overall architecture of a basic CNN can be described in terms of three fundamental building blocks:

- convolutional layers
- pooling layers, which typically follow a convolutional layer
- fully connected[2] layers, which are basically the layers of a traditional ANN, and are also encountered in the last few layers of a CNN

Convolution operation

For deep learning applications, convolutions will typically involve multi-dimensional inputs/outputs. The input is convolved with one or more multi-dimensional filters (or "kernels"), which are basically weighting functions.

[1] in particular, ANNs resemble to some extent the way brain neurons process the information coming from "dendrites" and output it into "axons"

[2] in the sense that all neurons in adjacent layers are mutually connected

The convolution operation can be thought of as having a filter sliding through the input volume with a given stride. Each new entry of the output is computed as the sum of all entries resulting from the element-wise multiplication between the corresponding sub-volume of the input and the filter, as depicted in Figure 2.1. For image processing applications, the filter shall have the same depth as the input, while the depth of the output corresponds to the number of filters employed.[3] In other words, we are interested in 2D convolutions[4] of an input volume with arbitrary depth.

In signal processing literature, the operation described here above is actually referred to as "cross correlation". Indeed, the rigorous mathematical definition of 2D convolution would actually require to flip the filter both vertically and horizontally, prior to performing the element-wise multiplication. It is nevertheless much more common in ML literature to refer to the operation here presented simply as "convolution".

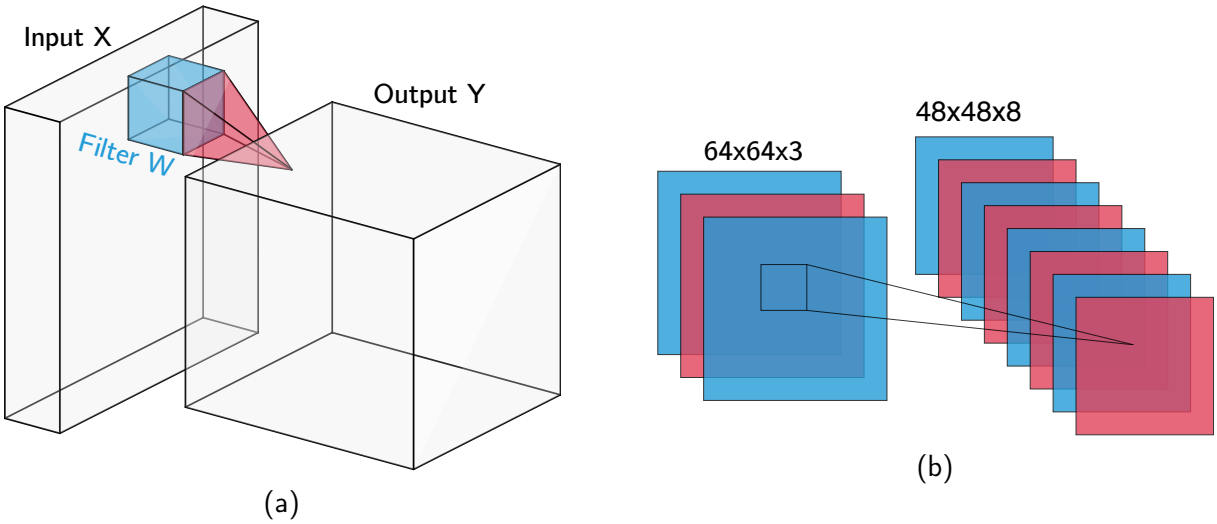


Figure 2.1: *Graphical representation of the 2D convolution with a volume*

The entries of the output tensor $\mathbf{Y}$, resulting from the discrete 2D convolution between an input volume $\mathbf{X}$ and a kernel $\mathbf{W}$ may then be computed as

$$\begin{aligned} \mathbf{Y}_{ij} &= (\mathbf{X} * \mathbf{W})_{ij} \\ &= \sum_m \sum_n \mathbf{X}_{mn} \mathbf{W}_{(i-m),(j-n)} \end{aligned} \tag{2.1}$$

The purpose of the filter is to act as a feature detector. For instance, considering a simple image processing application of a grayscale image (i.e. 2D input) convolved with a filter matrix, one may perform horizontal/vertical edge detection using a Sobel filter:

$$\mathbf{W}_{\text{vert}} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad \mathbf{W}_{\text{horiz}} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

A CNN will instead learn the parameters of such filters, in order to detect edges, corners, blobs, textures or higher-level features in the input image. Any convolutional layer

[3] i.e. the individual outputs of each filter are stacked along the depth direction

[4] in the sense that the filter is only slid horizontally and vertically

will include a collection of learnable filters, with the addition of a bias $\mathbf{b}$ to be learned as well and a given non-linear activation function $g(\cdot)$:

$$\text{output} = g\big((\text{input} * \text{filter}) + \mathbf{b}\big)$$

Typical activation functions are:

- hyperbolic tangent, $g(\mathbf{z}) = \tanh(\mathbf{z})$
- sigmoid, $g(\mathbf{z}) = \frac{1}{1+e^{-\mathbf{z}}}$ (its use is typically limited to the final output layer only)
- ReLU, $g(\mathbf{z}) = \max(0, \mathbf{z})$
- leaky ReLU, $g(\mathbf{z}) = \max(\underset{\downarrow}{\varepsilon}\mathbf{z}, \mathbf{z})$
  e.g. $= 0.01$

Pooling operation

Convolutional layers are typically followed by a pooling layer, which in principle might be of two kinds: max-pooling or average-pooling (much less common).

Similarly to convolutions, pooling can be visualized by considering a sliding cube through the input volume, but the operation is carried out independently over each channel. Pooling consists in either taking the maximum or the average of the elements enclosed in each square slice of the sliding cube. The size of the pooling window is usually 2-3 pixels.

It is then clear that the purpose of the aforementioned operation is to condense input information, hence speeding up both training and inference. In other words, the number of parameters of successive layers becomes progressively lower, as the size of the intermediate representations of the input image is reduced.

As it could be intuitively expected, pooling also has the effect of increasing the robustness of a CNN to translations in the image frame of a given object.

Overall architecture

The general rule of thumb for a typical CNN architecture is to follow two simple principles: to progressively decrease the image size and to progressively increase the number of channels.

This translates into an architecture in which the shallowest layers are responsible for detecting very basic low-level features (e.g. edges, corners, etc.), while, as we go deeper in the network, the layers tend to focus on increasingly more sophisticated high-level features.

An example of early architecture that reflects these principles and has been widely employed for various purposes is `AlexNet` [Kri12], which is depicted in Figure 2.2.

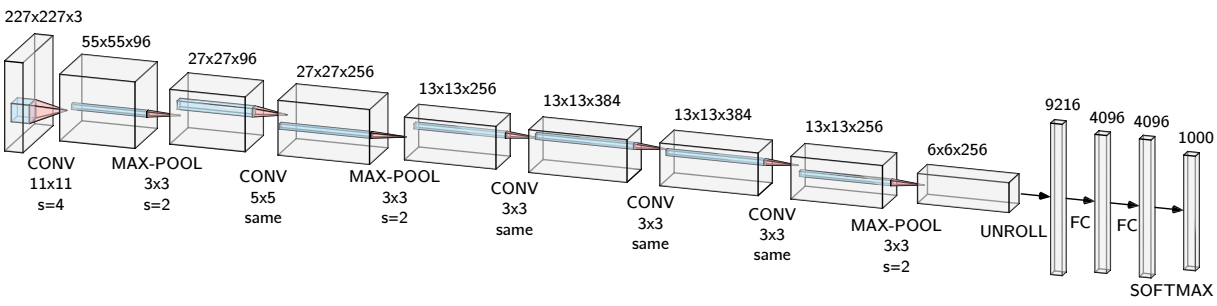


Figure 2.2: *AlexNet architecture*

In a typical CNN architecture, just as in `AlexNet`, the last output volume is unrolled into a vector, which is followed by one or more fully connected (FC) layers and eventually a softmax classifier. This allows classifying an input image according to an arbitrary number of classes that may be present in the dataset and to output the corresponding confidence level of that prediction.

It can happen that an architecture built for a very specific application may perform extremely well also on completely different computer vision tasks, if properly trained.

Transfer learning A concept of paramount importance in deep learning-based computer vision is that of "transfer learning". Training an accurate computer vision network from scratch would typically take a very long time and huge computational resources. Nowadays, thanks to the huge ML open source community, it is actually quite easy to retrieve the values of weights for a given architecture that has been pre-trained on extremely large computer vision datasets[(5)] for weeks, using many high-end GPUs. These huge datasets contain images of terrestrial objects, animals, people, etc. and although it might appear a completely different domain compared to spaceborne applications, it actually turns out that very basic low-level features detected by the shallowest layers of a CNN are the same across disparate domains, including spaceborne imagery.

We may hence conclude that, whenever pre-trained weights are available, it always makes sense to use them for initializing the parameters before training on our own dataset. Indeed, this saves a large number of training epochs, given that the neural network has "already learned" to detect low-level features once we start training it.

In addition, since a very large dataset may not always be available, it would be convenient in these situations to exclusively train the deepest layers of the CNN, while the first layers are kept frozen (i.e. same weights of the pre-trained architecture). In the event of a very small dataset, it is suggested to train the softmax classifier only.

2.1.2 Gradient-based learning

The learning process during the training of an ML algorithm occurs via minimization of a cost function, computed as the sum between a loss function cost and a regularization cost. Let m be the number of training examples and L be the number of layers in a generic network.

$$J = \underbrace{\frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\mathbf{Y}_i, \hat{\mathbf{Y}}_i)}_{\text{loss function cost}} + \underbrace{\frac{\lambda}{2m}\sum_{l=1}^{L}\left\|\mathbf{W}^{[l]}\right\|_F}_{\text{regularization cost}} \tag{2.2}$$

The loss function in Equation (2.2) indicates the average deviation of the predictions $\hat{\mathbf{Y}}_i$ from the corresponding ground truths $\mathbf{Y}_i$. L_2-regularization is instead the standard technique to prevent the model from overfitting the training set[(6)] and $\|\mathbf{W}^{[l]}\|_F$ indicates the Frobenius norm of the weights at layer l.

The iterative minimization of the cost function requires to compute the gradients of J with respect to the learned parameters, i.e. $\mathrm{d}\mathbf{W}^{[l]} := \frac{\partial J}{\partial \mathbf{W}^{[l]}}$ and $\mathrm{d}\mathbf{b}^{[l]} := \frac{\partial J}{\partial \mathbf{b}^{[l]}}$. Such

(5) such as `ImageNet`, `MS COCO`, `PASCAL VOC`, etc.

(6) this is intuitively achieved by penalizing all trained parameters, which are multiplied with a scalar value λ, called the regularization hyper-parameter, that requires tuning

gradients are computed by means of the backpropagation algorithm [Cha95].

The most basic fashion for iteratively converging towards a minimum of the cost function is by means of the Gradient Descent Method (GDM), which consists in updating the learned parameters by taking small steps along the direction opposite to the computed gradient values:

$$\begin{aligned} \mathbf{W}^{[l]} &= \mathbf{W}^{[l]} - \alpha \, \mathrm{d}\mathbf{W}^{[l]} \\ \mathbf{b}^{[l]} &= \mathbf{b}^{[l]} - \alpha \, \mathrm{d}\mathbf{b}^{[l]} \end{aligned} \tag{2.3}$$

where the hyper-parameter α is called the learning rate, which is a scalar value that rules the speed of convergence. α requires proper tuning: an excessively high value may cause the algorithm to diverge or to continuously overshoot the minimum of the cost function; on the contrary, a very small learning rate would result into a very slow training.

Gradient-based methods are actually never guaranteed to converge to the global optimum of the objective function, which is in general non-convex. Indeed, these method have been in use since late 50's, but they were initially limited to linear systems until it was realized that the presence of local minima did not represent a major issue in practice [LeC98]. It is conjectured that, if a generic ANN is oversized for the task,[7] which is usually the case, then the presence of extra-dimensions in a very high-dimensional space translates into a very low risk of unattainable regions.

Stochastic Gradient Descent

In the deep-learning era it is not untypical to deal with extremely large datasets, in which the number of training examples may be as high as $m = 10^{7 \div 8}$. In order to speed up training, instead of processing the entire dataset before updating the learned parameters, a very common strategy is that of partitioning the dataset into several "mini-batches". In this way, the gradient-based optimization only has to be executed on a small subset of training examples prior to performing the parameter update. This results into noisier yet much faster convergence.

This new framework is called Stochastic Gradient Descent (SGD), as opposed to Batch Gradient Descent (BGD) for which the processed batch corresponds to the entire dataset.

ADAM optimization

Other gradient-based algorithms that are more sophisticated than the basic GDM are nowadays in use.

For instance, GDM with momentum term [Qia99] is particularly helpful for accelerating convergence whenever mini-batches are used. Instead of updating the parameters proportionally to the gradients theirselves, it uses their exponentially-weighted[8] moving average. This intuitively translates into smoother convergence, which in turn also allows setting larger learning rates.

An additional method to speed up mini-batch learning is represented by RMSProp [Tie12]. The method damps the oscillating behavior in directions of the vector-space with high curvature, while speeding up convergence along directions with soft and consistent gradients.

[7] in the sense that it has many extra-dimensions more than the minimal parametrization of the model would require

[8] exponential weighting is an approximation of the actual moving average, but turns out to be much computationally cheaper

The underlying idea of ADAptive Momentum (ADAM) optimization [Kin14] is that of combining the momentum update with RMSProp, along with the implementation of bias correction. Specifically, ADAM computes an adaptive learning rate based on the normalized approximate first and second moments[(9)] of the gradients. The approximations are computed using the exponentially-weighted average of past gradients. The importance of normalization stems from the fact that, since the moments will undergo zero-initialization, they tend to be biased towards zero, especially during the initial iterations. The resulting algorithm is reported in Equation (2.4).

$$
\begin{aligned}
&\texttt{zero-initialization: } \mathbf{v}_{\mathrm{dW}},\ \mathbf{S}_{\mathrm{dW}},\ \mathbf{v}_{\mathrm{db}},\ \mathbf{S}_{\mathrm{db}} \\
&\texttt{for } t \texttt{ in mini-batches:} \\
&\qquad \texttt{compute: } \mathrm{d}\mathbf{W},\ \mathrm{d}\mathbf{b} \quad (\texttt{backpropagation}) \\
&\qquad \mathbf{v}_{\mathrm{dW}} = \frac{1}{1-\beta_1^t}\left[\beta_1 \mathbf{v}_{\mathrm{dW}} + (1-\beta_1)\,\mathrm{d}\mathbf{W}\right] \\
&\qquad \mathbf{v}_{\mathrm{db}} = \frac{1}{1-\beta_1^t}\left[\beta_1 \mathbf{v}_{\mathrm{db}} + (1-\beta_1)\,\mathrm{d}\mathbf{b}\right] \\
&\qquad \mathbf{S}_{\mathrm{dW}} = \frac{1}{1-\beta_2^t}\left[\beta_2 \mathbf{S}_{\mathrm{dW}} + (1-\beta_2)\,\mathrm{d}\mathbf{W}^2\right] \\
&\qquad \mathbf{S}_{\mathrm{db}} = \frac{1}{1-\beta_2^t}\left[\beta_2 \mathbf{S}_{\mathrm{db}} + (1-\beta_2)\,\mathrm{d}\mathbf{b}^2\right] \\
&\qquad \texttt{update:} \\
&\qquad \mathbf{W} = \mathbf{W} - \alpha \frac{\mathbf{v}_{\mathrm{dW}}}{\sqrt{\mathbf{S}_{\mathrm{dW}}} + \varepsilon} \\
&\qquad \mathbf{b} = \mathbf{b} - \alpha \frac{\mathbf{v}_{\mathrm{db}}}{\sqrt{\mathbf{S}_{\mathrm{db}}} + \varepsilon}
\end{aligned}
\tag{2.4}
$$

ε denotes a small number that is summed with the denominator to prevent from dividing by zero, usually set to $\varepsilon = 10^{-8}$. The learning rate is now adaptive, but the hyper-parameter α still requires tuning. The remaining hyper-parameters β_1, β_2 might in principle be tuned as well, although they are typically set to the default values proposed by the authors of [Kin14], namely $\beta_1 = 0.9$, $\beta_2 = 0.999$.

2.1.3 Object detection

The object localization problem consists in an extension of the image classification problem: besides identifying the presence of one of the dataset classes, the CNN is also required to predict the portion of the image in which the object lies in, by drawing a Bounding Box (BB) that encloses it.

The object detection task is a further generalization of the problem and implies the capability of detecting multiple objects within the same image. In our discussion we will always refer to a relative navigation scenario between two S/Cs only, which means that the task of interest is indeed object localization. On the other hand, all current state of the art algorithms in this field were actually developed in the more general object detection framework: for this reason, object detection algorithms were considered in our

(9) i.e. mean and uncentered variance, respectively

pose estimation pipeline, taking into account all possible simplifications that may be introduced in the code under the assumption of a single spacecraft in the image frame.

Early approaches to object detection made use of a sliding window implementation. The sliding window successively crops overlapping regions of the input image and then a classifier is run individually on each of them. In the pre-ANN era, when much simpler linear classifiers were still in use, the computational cost of such an approach was acceptable. With the advent of deep neural networks characterized by millions of parameters, the need for a different approach that avoids redundant computation became immediately clear. Indeed, if we were to run a CNN on image crops with substantial overlap, many redundant calculations would take place, entailing drastic consequences for real-time execution.

Current state-of-the-art methods for object detection can be classified into two main categories: region proposal methods (such as `R-CNN` [Gir14], `Fast R-CNN` [Gir15] and `Faster R-CNN` [Ren15]) and one-stage methods (such as `YOLO` [Red16], `SSD` [Dai16] and `EfficientDet` [Tan20]).

Region proposal detectors

The underlying idea that pushed the development of these methods is to avoid wasting computations on regions that are not of interest, i.e. those for which it can be easily excluded that any of the dataset classes are present. This translates into a common feature of all these methods: they always pre-process the image in order to propose candidate regions where to look for objects.

R-CNN (2013) The algorithm [Gir14] is composed of three subsystems:

i) selective search [Uij13] is run on the input image to propose candidate regions

ii) all candidate regions are individually processed by a CNN to extract a feature map for each of them

iii) based on the computed feature maps, class-specific Support Vector Machines (SVMs) classify each region and output the final BB

The algorithm is nevertheless quite computationally inefficient, making it unsuitable for real-time execution. Indeed, the selective search algorithm ends up performing many redundant computations. The same goes for feature extraction and object classification, which are run multiple times on several overlapping portions of the proposed regions.

Fast R-CNN (April 2015) This new architecture [Gir15] tackles the main issue of `R-CNN`, i.e. its inefficient redundant computations. While the original `R-CNN` individually computed feature maps on as many as 2000 proposed regions, here the image is initially pre-processed by running a CNN once on the whole image. Like its predecessor, `Fast R-CNN` also uses selective search for region proposal.

Faster R-CNN (June 2015) Although the `Fast R-CNN` method significantly improved the computational efficiency of the original `R-CNN`, the former is still characterized by an evident computational bottleneck: the selective search algorithm, which represents most of the total runtime. In the `Faster R-CNN` architecture [Ren15], region proposal is integrated into the CNN itself, no longer needing to run selective search.

One-stage detectors

One-stage detectors exhibit superior computational efficiency compared to state-of-the-art region proposal architectures, while maintaining high detection accuracy. We will now focus on the You Only Look Once (YOLO) architecture. Which is currently the most popular object detection algorithm and has been implemented in the pose estimation pipeline proposed in this dissertation.

YOLO architecture The `YOLO` architecture went through various official and unofficial revisions since the original version, `YOLOv1`, was proposed in 2016 by Redmon et al. [Red16]. As the name may suggest, the basic idea is to make use of a convolutional implementation that directly processes the entire image. In other words, unlike all previous work, object detection is reframed as a single regression problem.

In the first iteration of the architecture, `YOLOv1`, the image is resized to a fixed 448×448 size and divided into grid squares. At this point the image is fed to a CNN that detects whether any of the cells contains the center of an object, while also computing the confidence probability of such prediction.

One of the main limitations of this approach is that, since the method corresponds to running object localization over each cell (although convolutionally, i.e. all at once), only a single object per cell can be detected. In fact, a 19×19 grid is often used, which usually guarantees that no more than one object's center falls within the same cell.

Eventually, the raw predictions are refined by means of thresholding and non-max suppression, which respectively reject low-confidence predictions[(10)] and eliminate duplicate detections of the very same object. Non-max suppression is an iterative procedure based on the following workflow:

i) `initialize`:
P = list of proposed BBs (after thresholding)
F = [empty]

ii) `while` P is not empty:
Pick the BB with the largest p_c, remove from P all BBs having IoU ≥ 0.5 with the former, add the selected BB to F

iii) `return`: $F \leftarrow$ final predictions

`YOLOv2` (2017) [Red17] substantially improved detection accuracy compared to its predecessor.

A key difference from the previous iteration is the introduction of multi-scale training, which consists in changing the size of the input image every 10 training epochs: this makes the network robust to processing images of different sizes.

This version also introduced the use of Anchor Boxes (ABs), which are pre-defined shapes[(11)] representative of the BB aspect ratios that we are most likely to find in a test image. The most sophisticated fashion for choosing anchor boxes is to run K-means clustering [Mac67] on the dataset. If an $S \times S$ grid is used, then the dimensionality of the

[(10)] typical threshold: $p_c \leq 0.6$
[(11)] $5 \div 10$ different ones are typically assigned

output will be $S \times S \times \big(\#\text{ABs} \cdot (\#\text{classes} + 5)\big)$, which means that for each grid cell the neural network will output a prediction encoded in a vector of the following kind:

$$\mathbf{Y}_{ij} = \big[\underbrace{p_c \quad b_x \quad b_y \quad b_h \quad b_w \quad C_1 \dots C_N}_{\text{AB}_1} \mid \underbrace{p_c \quad b_x \quad b_y \quad b_h \quad b_w \quad C_1 \dots C_N}_{\text{AB}_2} \mid \dots \big]^\intercal$$

in which the predictions for each of the ABs are vertically stacked. p_c indicates the confidence score; b_x, b_y denote the center of the BB and b_h, b_w its height and width; $\big[C_1 \quad C_2 \dots \quad C_N\big]$ is a boolean vector that encodes the predicted class.

Combining ABs with non-max suppression yields an extremely powerful and accurate algorithm, that allows identifying multiple objects within the same grid cell.

`YOLOv3` (2018) [Red18] is characterized by a more complex structure, hence trading speed for higher accuracy, compared to its predecessor. This new architecture features multi-scale prediction: during inference the image is scaled to three different sizes, which proved particularly helpful at increasing the detection accuracy for small objects. The new architecture, called *Darknet-53*, as the name suggests, is made up of 53 convolutional layers, compared to only 19 convolutional layers that were present in `YOLOv2`.

Right after J. Redmon announced in February 2020 that he was stepping away from computer vision research, two unofficial revisions of `YOLO` (from two different authors) followed in the upcoming months.

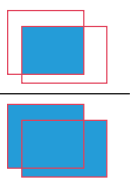
`YOLOv4` [Boc20] was released in April 2020, which introduced significant accuracy improvements compared to the latest official release.

A few weeks later, in June 2020, a second unofficial release referred to as `YOLOv5` [Ult] further outperformed all its predecessors. In particular, 5 different versions of the model with different sizes were implemented, and the smallest one (`YOLOv5s`) still achieves outstanding accuracy while being extremely light to run (only 7.5M parameters). Compared to the latest official release, this new architecture replaced the *Darknet* backbone with Cross Stage Partial Networks (CSPNet)[Wan20], which recently proved substantial runtime improvements in deep network architectures. The second fundamental difference of `YOLOv5` is that it generates feature pyramids using the Path Aggregation Network (PANet) [Liu18]. Feature pyramids make the model robust to object scaling, i.e. they help the network generalize to previously unseen scales of the same object.

Performance metrics: IoU and AP

The simplest way to evaluate the degree of overlap between the predicted bounding box and the corresponding ground truth is by measuring the area ratio between their intersection and their union, namely

$$\text{IoU} := \frac{\text{BB}_1 \cap \text{BB}_2}{\text{BB}_1 \cup \text{BB}_2} = \frac{\quad}{\quad} \tag{2.5}$$



Before introducing some global evaluation metrics that measure the performance of an object detection algorithm on the entire dataset, we should first define some quantities. A given prediction, depending on its correctness, may either be classified as a True Positive

(TP), False Positive (FP) or False Negative (FN). We may at this point define Precision and Recall, which are respectively computed as

$$\begin{aligned} P &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \\ R &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \end{aligned} \tag{2.6}$$

The AP is individually computed for each class in the dataset as the area under the precision-recall curve

$$\mathrm{AP} := \int_0^1 P(R)\,\mathrm{d}R \tag{2.7}$$

The various precision-recall pairs that define the curve are obtained by gradually increasing the prediction confidence cutoff used for thresholding raw predictions. Missed detections resulting from a low confidence score are clearly treated as FNs. The precision-recall curve should be in principle monotonically decreasing. It nevertheless may happen that $P(R)$ locally deviates from a purely monotonically decreasing behavior, which is usually corrected by considering the so called "interpolated precision", in the integral in Equation (2.7). This is simply achieved by defining:

$$P_{\mathrm{interp}}(\overline{R}) = \begin{cases} P(\overline{R}) & \text{if} \quad P(R) \leq P(\overline{R}) \quad \forall\, R > \overline{R} \\ \max\big(P(R > \overline{R})\big) & \text{else} \end{cases} \tag{2.8}$$

In an object detection scenario, since a perfect match of the BB is almost never achieved, identifying a prediction as a TP requires setting a threshold that defines the minimum required degree of overlap. Three typical metrics are defined accordingly:

- AP_{50}, for which the threshold is set to $\mathrm{IoU}_{\mathrm{min}} = 0.5$
- AP_{75}, for which $\mathrm{IoU}_{\mathrm{min}} = 0.75$
- $\mathrm{AP}_{50:5:95}$, that is simply the average of the 10 values AP_{50}, AP_{55}, $\mathrm{AP}_{60}, \ldots, \mathrm{AP}_{95}$

Although it is not of interest under the assumptions in this work, in a multi-class framework, a global performance metric for the whole dataset would be computed as the mean of the AP values over all categories and is called the mean Average Precision (mAP). We may hence define accordingly the mAP_{50}, mAP_{75} and $\mathrm{mAP}_{50:5:95}$ metrics.

2.1.4 Landmark regression

CNNs that regress the location of semantic keypoints have been extensively developed in the fields of human/object pose estimation and face expression recognition. There are basically two deep-learning based approaches to the problem: either directly regressing the keypoints' position or regressing a heatmap[(12)] that indicates the probability, at a given position in the image, of locating the landmark of interest.

We will now describe the details of a state-of-the-art architecture that follows the latter approach. This model has been developed for human pose estimation and has been proved

[(12)] the keypoint location is clearly estimated to be in correspondence of the peak in each individual heatmap

to outperform all previous work in this field, both in terms of accuracy and computational efficiency. The same architecture is also employed in the pose estimation pipeline proposed in this dissertation.

HRNet architecture

The strength of the High-Resolution Network (HRNet) architecture [Sun19a] lies in two main distinctive aspects.

- Most previous architectures recover high resolution representations by performing an upsampling process downstream of a high-to-low resolution network. In contrast, `HRNet` maintains the initial high-resolution representation throughout the entire network. This clearly eliminates the loss of information associated with traditional approaches, resulting in more accurate heatmaps, which is of paramount importance in a spaceborne relative navigation scenario.
  In particular, the network starts with a high-resolution subnetwork whose resolution is kept unaltered up to the last layer. As it is depicted in Figure 2.3, lower-resolution subnetworks are gradually stacked in parallel as we go deeper in the network.

- Instead of aggregating high- and low-resolution representations, `HRNet` performs repeated multi-scale fusions to boost the low-level representations with the aid of high-level representations, and vice-versa.

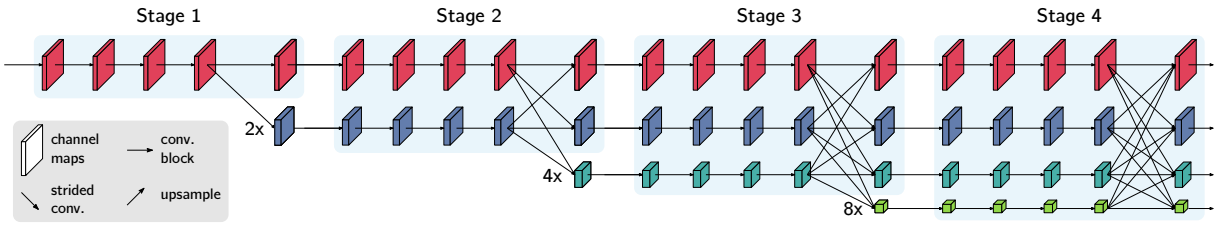


Figure 2.3: *Main body of the `HRNet` architecture*

The overall pipeline consists of a 4-stage main body that outputs feature maps having the same resolution as its input. This followed by a regressor which estimates the heatmaps. There will be as many heatmaps as the number of landmarks, and each of them has the same size of the input image.

At each new stage, one additional lower-resolution subnetwork is stacked in parallel. Downsampling factors are successive powers of 2, while the number of channels is accordingly doubled every time.

Multi-scale fusions Each stage is divided into exchange blocks: specifically, the 2nd, 3rd and 4th stages contain 1, 4, 3 exchange blocks, respectively.
Let s = stage #, r = rth resolution, b = bth exchange block. Each block will hence contain s parallel convolution units C^b_{sr}, with an exchange unit ε^b_s across the parallel units. Without loss of generality, let us consider the 3rd stage, just for visualizing the multi-scale fusion process:

$$\begin{matrix} C^1_{31} & \searrow & & \nearrow & C^2_{31} & \searrow & & \nearrow & C^3_{31} & \searrow & & \nearrow & C^4_{31} & \searrow & \\ C^1_{32} & \rightarrow & \varepsilon^1_3 & \rightarrow & C^2_{32} & \rightarrow & \varepsilon^2_3 & \rightarrow & C^3_{32} & \rightarrow & \varepsilon^3_3 & \rightarrow & C^4_{32} & \rightarrow & \varepsilon^4_3 \\ C^1_{33} & \nearrow & & \searrow & C^2_{33} & \nearrow & & \searrow & C^3_{33} & \nearrow & & \searrow & C^4_{33} & \nearrow & \end{matrix}$$

Each successive output is computed as an aggregation of input maps, i.e. $\mathbf{Y}_k = \sum_{i=1}^{s} \texttt{resampling}(\mathbf{X}_i, k)$ where the `resampling`() function may either indicate downsampling or upsampling, from the ith resolution to the kth. One or more strided 3×3 convolutions are used for downsampling; upsampling, instead, is achieved by means of nearest-neighbor interpolation followed by a 1×1 convolution.

Heatmap regression Heatmaps are regressed from the high-resolution output of the final exchange unit. We recall that such a representation is actually enriched by the high-level features detected in lower-resolution subnetworks. For training this type of network, the GT heatmaps are labeled as 2D Gaussian distributions with a standard deviation of 1×1 pixel, centered in correspondence of the GT position of each landmark. The loss function for a given image is defined as the mean squared error (in terms of pixel distance) over all keypoints.

Performance metrics: OKS and AP

The typical evaluation metrics in a keypoint detection framework are based on the Object Keypoint Similarity (OKS).

$$\text{OKS} := \sum_{i=1}^{n} v_i \cdot \exp\left(\frac{-d_i^2}{2s^2k_i^2}\right) \tag{2.9}$$

In Equation (2.9), d_i denotes the Euclidean distance in pixels between the estimated keypoint and the corresponding GT; v_i is a boolean visibility flag that indicates whether or not the ith GT landmark lies inside the image frame; s indicates the object scale;[13] k_i is a keypoint-specific constant that controls falloff.[14]

Similarly to the IoU in an object detection framework, the OKS indicates the average degree of overlap between detected landmarks and their actual location. We may therefore set an OKS threshold to distinguish between correct and incorrect detections and likewise define AP_{50}, AP_{75} and $\text{AP}_{50:5:95}$.

2.2 Perspective-n-Point problem

The Perspective-n-Point (PnP) problem consists in estimating the pose of an object, given a set of n points of the object itself with known (or estimated) 3D model coordinates, and given the corresponding 2D projections detected in the image. Pose estimation methods also leverage the knowledge of the camera intrinsics, so as to seek for the pose that yields the best fit between the resulting projection of the object's points and the corresponding detected keypoints in the image frame.

Let C be the camera-fixed frame, while P indicates the image frame and B is the body-fixed frame, which in our case is representative of the S/C to rendezvous with. In the camera frame, C_z is the axis aligned with the boresight direction, while C_x, C_y are parallel to the P_x, P_y image axes, respectively. These reference frames are depicted in Figure 2.4.

[13] defined as the fraction of the total image occupied by the BB

[14] e.g. in a human pose estimation scenario, one would set larger k_i coefficients for hips and shoulders than for nose and ears

The coordinates in the camera frame of a generic point $\mathbf{r}^B$ of the target spacecraft can be expressed as

$$\mathbf{r}^C \equiv \begin{Bmatrix} x^C \\ y^C \\ z^C \end{Bmatrix} = \mathbf{R}_{\mathrm{B/C}}\mathbf{r}^B + \mathbf{t}_{\mathrm{C/B}} \tag{2.10}$$

where $\mathbf{t}_{\mathrm{C/B}}$ is the translation vector from the camera to the origin of the body frame and $\mathbf{R}_{\mathrm{B/C}}$ represents the direction cosine matrix expressing the rotation that aligns the body axes with the camera axes.

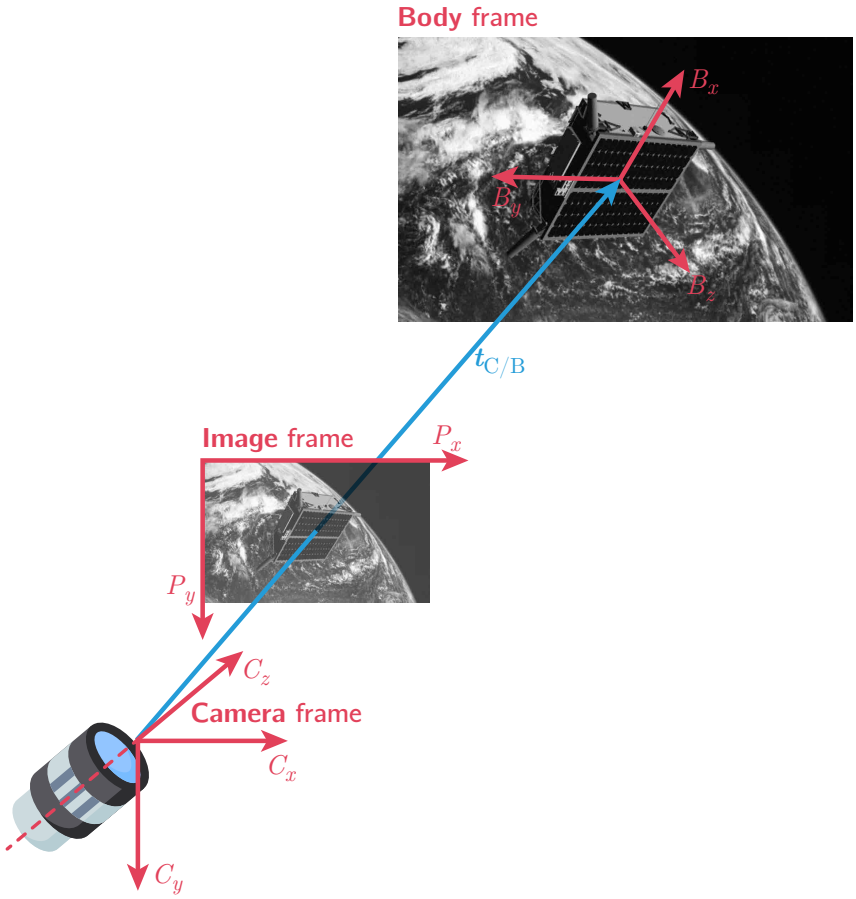


Figure 2.4: *Reference frames*

Using the so called "pinhole" camera model, one may easily compute the projection of $\mathbf{r}^C$ onto the image plane, based on the knowledge of camera intrinsics:

$$\mathbf{p} \equiv \begin{Bmatrix} u \\ v \end{Bmatrix} = \begin{Bmatrix} \dfrac{x^C}{z^C}\dfrac{f}{\rho_u} + u_0 \\ \dfrac{y^C}{z^C}\dfrac{f}{\rho_v} + v_0 \end{Bmatrix} \tag{2.11}$$

Specifically, f is the focal length, ρ_u and ρ_v are the horizontal and vertical pixel densities (which may in general differ), while (u_0, v_0) are the coordinates of the camera principal point.

Combining Equations (2.10) and (2.11), and rewriting using homogeneous coordinates, eventually yields the perspective projection equations:

$$\begin{Bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{Bmatrix} = \underbrace{\begin{bmatrix} f/\rho_u & 0 & u_0 \\ 0 & f/\rho_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{:= \mathbf{K}} \underbrace{\begin{bmatrix} \mathbf{R}_{\mathrm{B/C}} & | & \mathbf{t}_{\mathrm{C/B}} \end{bmatrix}}_{\substack{:= \mathbf{P} \\ (3 \times 4)}} \begin{Bmatrix} x^B \\ y^B \\ z^B \\ 1 \end{Bmatrix} \tag{2.12}$$

The actual coordinates in the image frame are then retrieved as

$$\mathbf{p} \equiv \begin{Bmatrix} u \\ v \end{Bmatrix} = \begin{Bmatrix} \tilde{u}/\tilde{w} \\ \tilde{v}/\tilde{w} \end{Bmatrix}$$

Note that the matrix $\mathbf{P}$, which is the unknown of our problem, has 6 degrees of freedom. Three parameters are required to describe the relative attitude (e.g. 3 Euler angles) and three more to describe the relative translation.

State-of-the-art PnP solvers are basically divided into two categories. Iterative solvers minimize a measure of the fit error between the projected model points and the image points. Multi-stage analytical approaches (e.g. EPnP) will instead leverage a linearized form of the projection equations.

2.2.1 Iterative solvers

Iterative solvers require a sufficiently close initial guess and $n \geq 3$ point correspondences. If on the one hand they are characterized by longer runtimes, compared to multi-stage analytical approaches, on the other hand they exhibit superior accuracy after a few iterations and higher robustness to the presence of outliers in the model-image correspondences.

The basic idea of these methods is to iteratively minimize the reprojection error, which is defined as the mean over all keypoints of the squared Euclidean distances (in pixels) between the projected model points and the detected image points:

$$E_{\mathrm{repr}} = \frac{1}{n} \sum_{i=1}^{n} \| \mathbf{p}_i^{\mathrm{model}} - \mathbf{p}_i^{\mathrm{image}} \| \tag{2.13}$$

Gauss-Newton optimization

The Gauss-Newton Method (GNM) is a modification of the Newton-Raphson Method (NRM). While the latter is an iterative method for computing the roots of a differentiable function $\mathbf{f}(\mathbf{x})$ written in the form $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, the GNM has been developed to minimize the sum of squared function values.

We may express the estimated pose as a 6-variable state vector, composed of three translation components and three Euler angles: $\mathbf{x} = \begin{bmatrix} \mathbf{t}^\intercal & \boldsymbol{\theta}^\intercal \end{bmatrix}^\intercal$.

Just as NRM, GNM is based on a first order Taylor expansion: in our case, at each iteration, we will linearize the fit error about the current pose estimate. Considering the ith point correspondence:

$$\mathbf{E}_i = \frac{\partial \mathbf{p}_i}{\partial \mathbf{r}_i} \frac{\partial \mathbf{r}_i}{\partial \mathbf{t}} \Delta \mathbf{t} + \frac{\partial \mathbf{p}_i}{\partial \mathbf{r}_i} \frac{\partial \mathbf{r}_i}{\partial \boldsymbol{\theta}} \Delta \boldsymbol{\theta} \tag{2.14}$$

The partial derivatives can be easily computed from Equation (2.12) (see [DAm14]) and are eventually rearranged in the form of a $2n \times 6$ Jacobian matrix

$$\mathbf{J}_i = \left[\frac{\partial \mathbf{p}_i}{\partial \mathbf{r}_i}\frac{\partial \mathbf{r}_i}{\partial \mathbf{t}}\Delta\mathbf{t} \quad \frac{\partial \mathbf{p}_i}{\partial \mathbf{r}_i}\frac{\partial \mathbf{r}_i}{\partial \boldsymbol{\theta}}\Delta\boldsymbol{\theta}\right] \Longrightarrow \mathbf{J} = \begin{bmatrix} \mathbf{J}_1 \\ \hline \mathbf{J}_2 \\ \hline \vdots \\ \hline \mathbf{J}_n \end{bmatrix} \tag{2.15}$$

At this point we may update the state vector at each kth iteration as $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta\mathbf{x}^{(k)}$, where the update vector is computed by solving the linear system

$$(\mathbf{J}^\intercal\mathbf{J})\Delta\mathbf{x} = \underset{\substack{\downarrow \\ 2n \times 1}}{\mathbf{J}^\intercal\mathbf{E}} \tag{2.16}$$

The typical stopping criterion is either a maximum number of iterations or whenever the norm of the computed update vector falls below a given threshold (e.g. 10^{-10}). As it is observed in [DAm14], pixel-level accuracy is achieved after $3 \div 4$ iterations, provided that the initial pose guess is within $\sim \pm 40°$ of error.

Levenberg–Marquardt optimization

The Levenberg-Marquardt Method (LMM) is another non-linear least squares fitting method that can be used to find a local minimum of a function.

The algorithm can be seen as an interpolated version between GNM and the Gradient Descent Method (GDM) described in Section 2.1.2. In particular, GDM performs parameter update by moving along the direction of steepest descent (i.e. the one parallel and opposite to the gradient). GNM will instead update parameters by assuming that the error function is locally quadratic in the parameters, hence finding the minimum of this quadratic. LMM combines these two algorithms and it tends to GDM when parameters are still far from the optimal value, while it is more similar to GNM once we approach the minimum.

Even though the method is substantially more robust to initial guesses that are far off the minimum, compared to GNM, this comes at the expense of the speed of convergence, which is slightly slower.

LMM solves a regularized least-squares problem, from which the state vector update may be eventually derived as the solution of the linear system

$$(\mathbf{J}^\intercal\mathbf{J} + \lambda\mathbb{I})\Delta\mathbf{x} = \mathbf{J}^\intercal\mathbf{E} \tag{2.17}$$

where λ is the regularization parameter (sometimes also indicated as the "damping parameter"). Without going into the details of how λ is adaptively tuned iteration by iteration, let us just provide some intuition about how this is done. The parameter is initialized to be sufficiently large, so as to perform updates along a direction closely aligned with the gradient, during the first few iterations. If any iteration results in a better approximation compared to the previous one, then λ is decreased (i.e. the algorithm approaches GNM); if on the contrary the approximation gets worse, λ is increased.

2.2.2 Efficient PnP solver

The Efficient Perspective-n-Point (EPnP) method [Lep09] is a closed-form solution to the PnP problem, having complexity of order $\mathcal{O}(n)$ and little loss of accuracy compared to iterative solvers. EPnP requires $n \geq 4$ point correspondences and its underlying idea is

that of expressing the n points as the weighted sum of 4 virtual control points $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4$ that become the unknowns of this formulation.

$$\mathbf{r}_i = \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j \qquad (i = 1, 2, \ldots, n)$$

We may then rewrite Equation (2.12) for a generic landmark i in terms of the 4 control points, each of which is described by 3 coordinates, which means that we end up with 12 control point coordinates $\mathbf{c}_j = \begin{bmatrix} c_j^x & c_j^y & c_j^z \end{bmatrix}^\mathsf{T}$ (where $j = 1, 2, 3, 4$) in our projection equations:

$$\begin{Bmatrix} \tilde{u}_i \\ \tilde{v}_i \\ \tilde{w}_i \end{Bmatrix} = \mathbf{K} \sum_{j=1}^{4} \alpha_{ij} \begin{Bmatrix} c_j^x \\ c_j^y \\ c_j^z \end{Bmatrix} \tag{2.18}$$

Plugging the third row of Equation (2.18) into the first two rows yields two linear equations for each model-point/image-point pair:

$$\begin{aligned} \sum_{j=1}^{4} \alpha_{ij} \frac{f}{\rho_u} c_j^x + \alpha_{ij}(u_0 - \tilde{u}_i) c_j^z = 0 \\ \sum_{j=1}^{4} \alpha_{ij} \frac{f}{\rho_v} c_j^y + \alpha_{ij}(v_0 - \tilde{v}_i) c_j^z = 0 \end{aligned} \tag{2.19}$$

$2n$ equations are obtained from Equation (2.19), which can be rearranged in matrix form as

$$\underset{2n \times 12}{\mathbf{A}} \underset{12 \times 1}{\mathbf{x}} = \mathbf{0}$$

where the unknown vector $\mathbf{x}$ contains the 12 control point coordinates. Since the image points are estimated by another subsystem, i.e. they are not the perfect projections of the true model points, they matrix $\mathbf{A}$ may have up to 4 linearly dependent columns. This means that there could be as much as four possible solutions, among which, the one having the lowest reprojection error is selected.

3 Chapter

Relative Pose Estimation Pipeline

In this chapter we will present the architecture of the Relative Pose Estimation Pipeline proposed in this dissertation. Our algorithms require the knowledge of camera intrinsics and of the 3D model of the target spacecraft to rendezvous with. Based on this, the architecture that has been developed is capable of estimating the pose of the target spacecraft, from a single monocular grayscale image given as input.

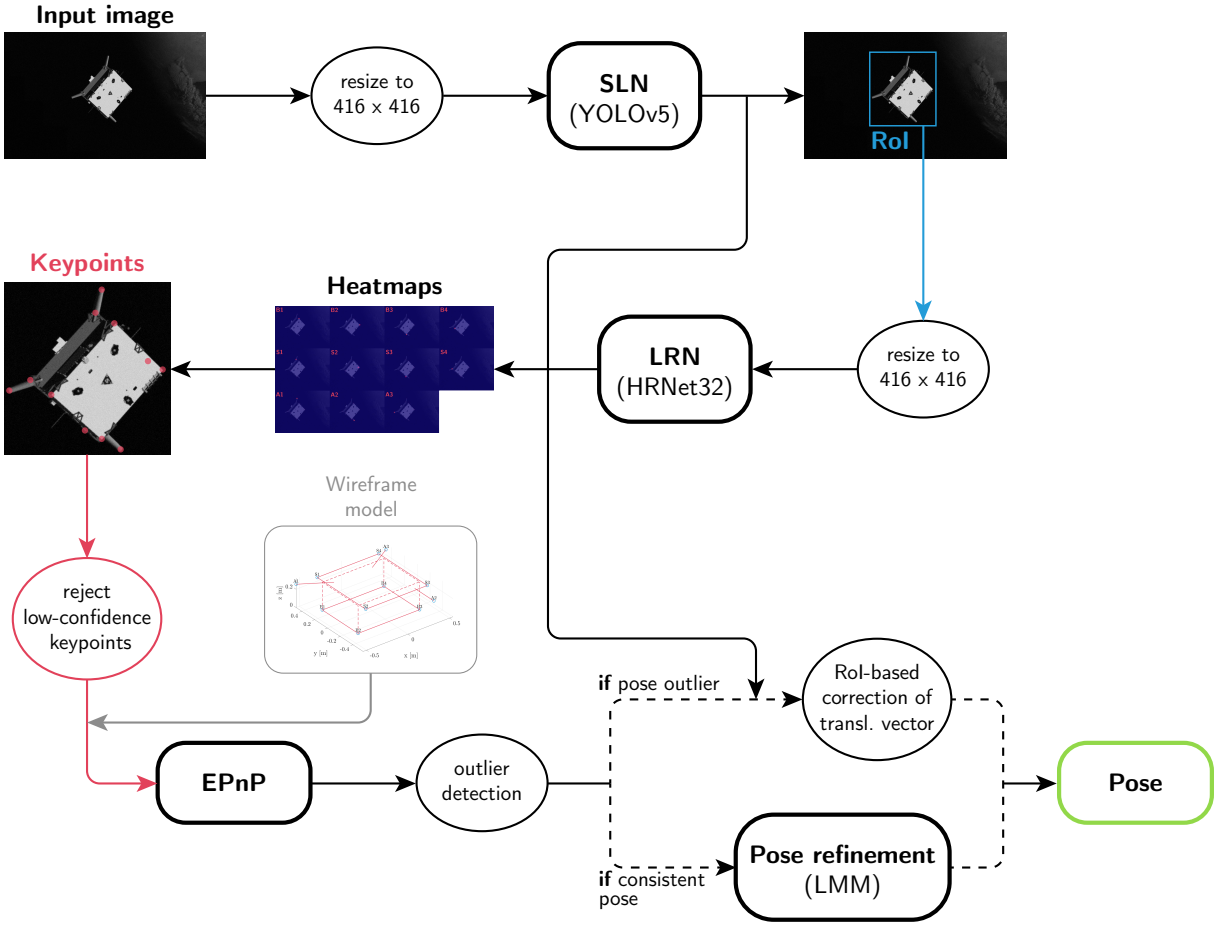


Figure 3.1: *Architecture of the pose estimation pipeline at inference time*

The outline of our architecture is represented in Figure 3.1 and it consists of three main subsystems.

The first subsystem, called the Spacecraft Localization Network (SLN) and described in Section 3.1, is responsible for identifying the Region of Interest (RoI) in the image.

This is followed in the pipeline by the Landmark Regression Network (LRN), that we detailed in Section 3.2, whose role is to detect semantic keypoints of the target S/C in the RoI.

The third and last subsystem is the pose solver, which, given the landmarks identified by LRN, seeks for the corresponding best pose fit. It will first run the EPnP algorithm to obtain an initial estimate of the pose and, in a nominal situation (i.e. if no pose outlier is detected), it will successively refine the initial solution using the Levenberg-Marquardt Method.

3.1 Spacecraft Localization Network

The Spacecraft Localization Network (SLN) is the first image processing subsystem of the Relative Pose Estimation Pipeline (RPEP) proposed in this dissertation. The SLN receives as input a grayscale image, that is properly resized to match the input size of 416×416 of our `YOLOv5` architecture. This subsystem outputs the so called Region of Interest (RoI), namely the Bounding Box (BB) coordinates associated with the portion of the image containing the S/C. Based on this, further processing of the image will exclusively focus on the identified RoI.

First of all, a one-stage detection approach was chosen over region proposal networks, given the clear superiority in terms of computational efficiency of the former class of methods. This is of paramount importance in a spaceborne navigation scenario, where the computing power constraints always make it necessary to opt for efficient yet robust algorithms. In this sense, the smallest model-size version of `YOLOv5` [Ult], named `YOLOv5s`, proved particularly interesting for our purposes and has been eventually selected.

In Table 3.1 the four `YOLOv5` available versions, characterized by four different model sizes, are compared [Ult] with the performance metrics achieved by `YOLOv3`,[(1)] on the `MS COCO` dataset, using one single NVIDIA Tesla V100 GPU. `YOLOv5`, with 7.5M parameters and 191 layers, appears as an excellent trade-off between speed and accuracy and is almost twice as fast as `v3`.

Table 3.1: *Performance comparison of the four `YOLOv5` architectures with `YOLOv3`*

Model	AP_{val}	AP_{test}	AP_{50}	$t^{GPU}_{inference}$	FPS_{GPU}	# parameters	FLOPs
`yolov5s`	37.0	37.0	56.2	2.4 ms	416	**7.5M**	**13.2B**
`yolov5m`	44.3	44.3	63.2	3.4 ms	294	21.8M	39.4B
`yolov5l`	47.7	47.7	66.5	4.4 ms	227	47.8M	88.1B
`yolov5x`	49.2	49.2	67.7	6.9 ms	145	89.0M	166.4B
`yolov3`	45.6	45.5	65.2	4.5 ms	222	63.0M	118.0B

[(1)] which is the latest "official" `YOLO` release

3.1.1 Training

The SPEED training labels released by SLAB only include the pose of the Tango spacecraft, provided in terms of translation vector and attitude quaternion for each image. This means that any further label that might be used for intermediate processing steps will have to be annotated.

The minimum rectangle enclosing the S/C in the image frame can be obtained by projecting a simple wireframe model of Tango using Equation (2.12) and the known pose. We may at this point annotate the BB of each image by taking the minimum and maximum values of the (P_x, P_y) coordinates of the small amount of points in this simplified 3D model.

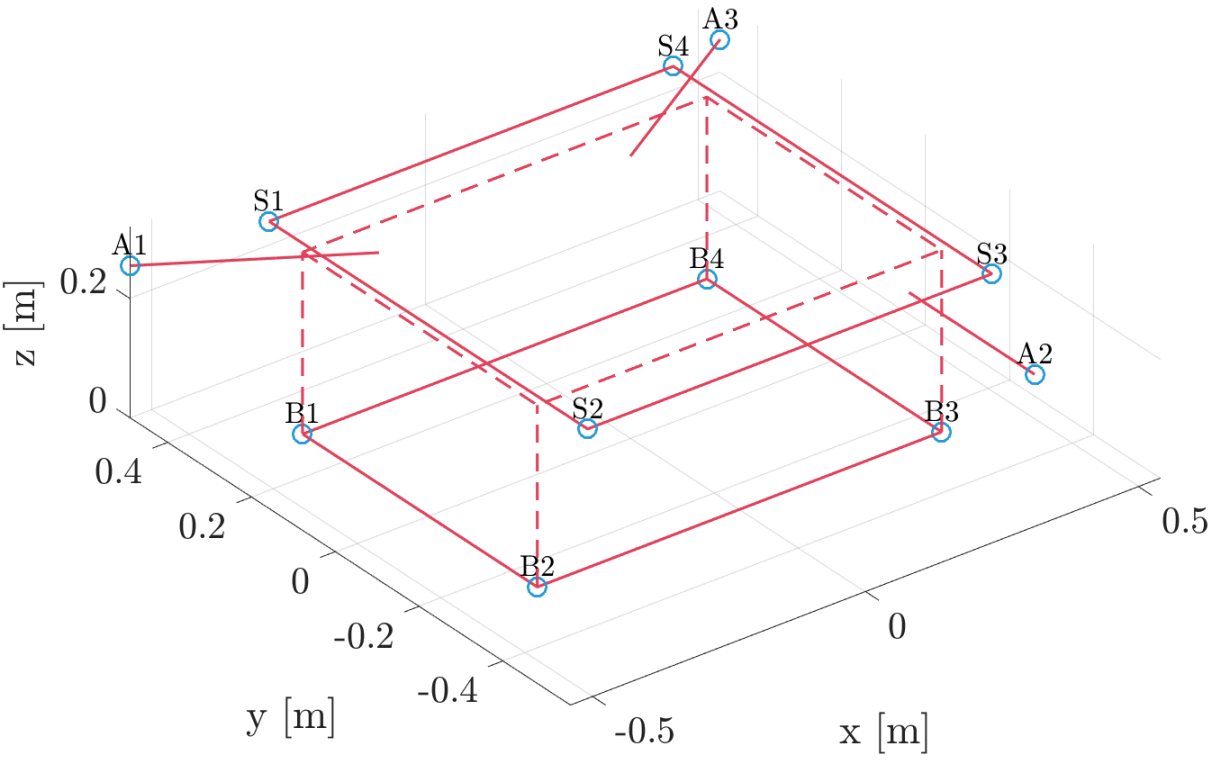


Figure 3.2: *Wireframe model of the Tango spacecraft*

The wireframe model used in our work is depicted in Figure 3.2. In particular, the model is composed of 11 semantic keypoints:

- points B1 to B4 are the edges of the bottom surface
- points S1 to S4 are the edges of the solar panel
- points A1 to A3 indicate the tips of the Formation Flying Radio Frequency (FFRF) antennas

The very same keypoints are actually those that are detected by the successive subsystem, the Landmark Regression Network (LRN), which we will soon introduce. The reason behind this choice is that these landmarks represent strong visual features of the spacecraft, and, independently of the pose, most of them will not be occluded by other surfaces.

In order to avoid unintentionally cropping out portions of the S/C from the detected RoI during inference, we will slightly relax the minimum rectangle enclosing the projected

wireframe model. Specifically, the BB labels are enlarged by the 10% of the average side between width and height of the minimum rectangle. In so doing, the CNN is indeed trained to predict a relaxed bounding box. In Figure 3.3, the dashed yellow line indicates the minimum rectangle, while the continuous line is the actual BB label.

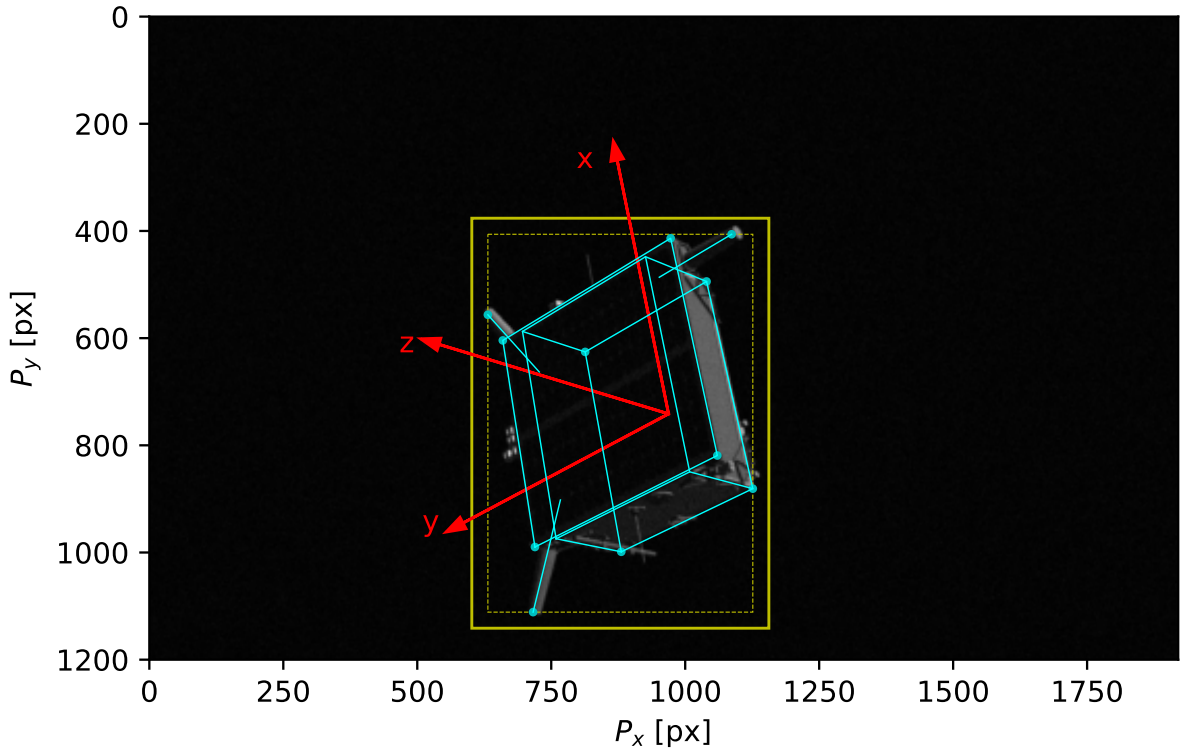


Figure 3.3: *Bounding box label of the* `img001971.jpg` *training image*

The network was trained for 125 epochs using SGD, with a mini-batch size of 48 images, learning rate $\alpha = 10^{-3}$, momentum equal to 0.9 and a weight decay of 5×10^{-5}. The binary cross-entropy loss is used during training, which, for a mini-batch of size m, is computed as

$$L_{\text{cross-entr}} = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i \ln(\hat{y}_i) + (1-y_i)\ln(1-\hat{y}_i)\right] \tag{3.1}$$

where y_i and the $\hat{y}_i$ are the GT and estimated values, respectively, of a given scalar entry of the output tensor.

In addition, given the assumption of single-class/single-object in the image, a few simplifications in the algorithm were introduced. In so doing, we are able to get rid of some unnecessary computation, also making sure that the algorithm outputs one single RoI, provided that the prediction confidence is at least 60%. In other words, we can directly output the prediction with the highest objectness score, with no need to process the raw results using the non-max suppression algorithm.

3.1.2 Performance evaluation

The performance of SLN has been evaluated using the standard metrics defined in Section 2.1.3. The 10 precision-recall curves in correspondence of the IoU thresholds 0.5, 0.55, 0.6, 0.65, . . . , 0.95 are reported in Figure 3.4. The curves are specifically computed considering the interpolated precision, as defined in Equation (2.8).

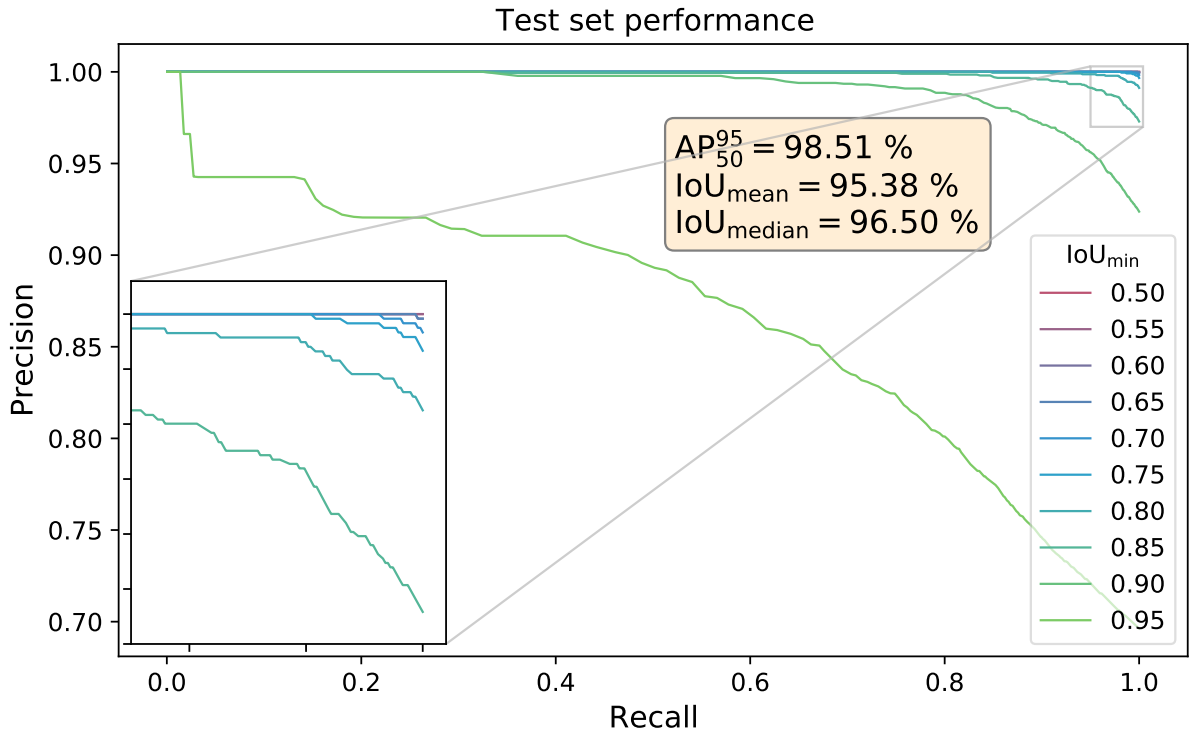


Figure 3.4: *Precision-recall curves, in correspondence of different IoU thresholds*

The `YOLOv5` architecture achieves an excellent accuracy, with $AP_{50}^{95} = 98.51\%$, after only 125 training epochs.

Indeed, by comparing the mean and median IoU metrics in Table 3.2, our spacecraft localization subsystem outperformed at this task both the SLAB baseline and the architecture proposed by the `UniAdelaide` team, which respectively ranked 4th and 1st in the Pose Estimation Challenge.

Table 3.2: *Performance comparison of SLN with other state of the art RoI detection subsystems*

	`stanford_slab` [Par19]	`UniAdelaide` [Che19]	Our SLN
Mean IoU	91.9%	95.34%	**95.38%**
Median IoU	93.6%	96.34%	**96.50%**

Let us now visualize the prediction results of our SLN subsystem, on a few randomly chosen test images.

In Figure 3.5 the predicted bounding box is drawn in each of the 6 corresponding synthetic test images and the related confidence is also reported. It can be seen that our

CNN performs extremely well at identifying a very tight RoI, independently of distance and illumination conditions.

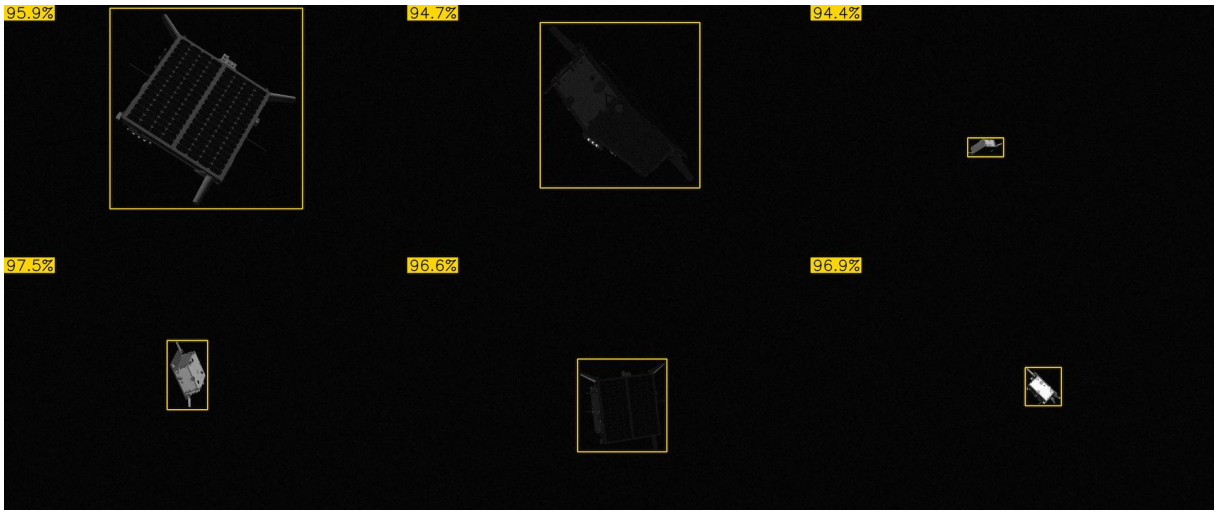


Figure 3.5: *SLN prediction on 6 test images with black background*

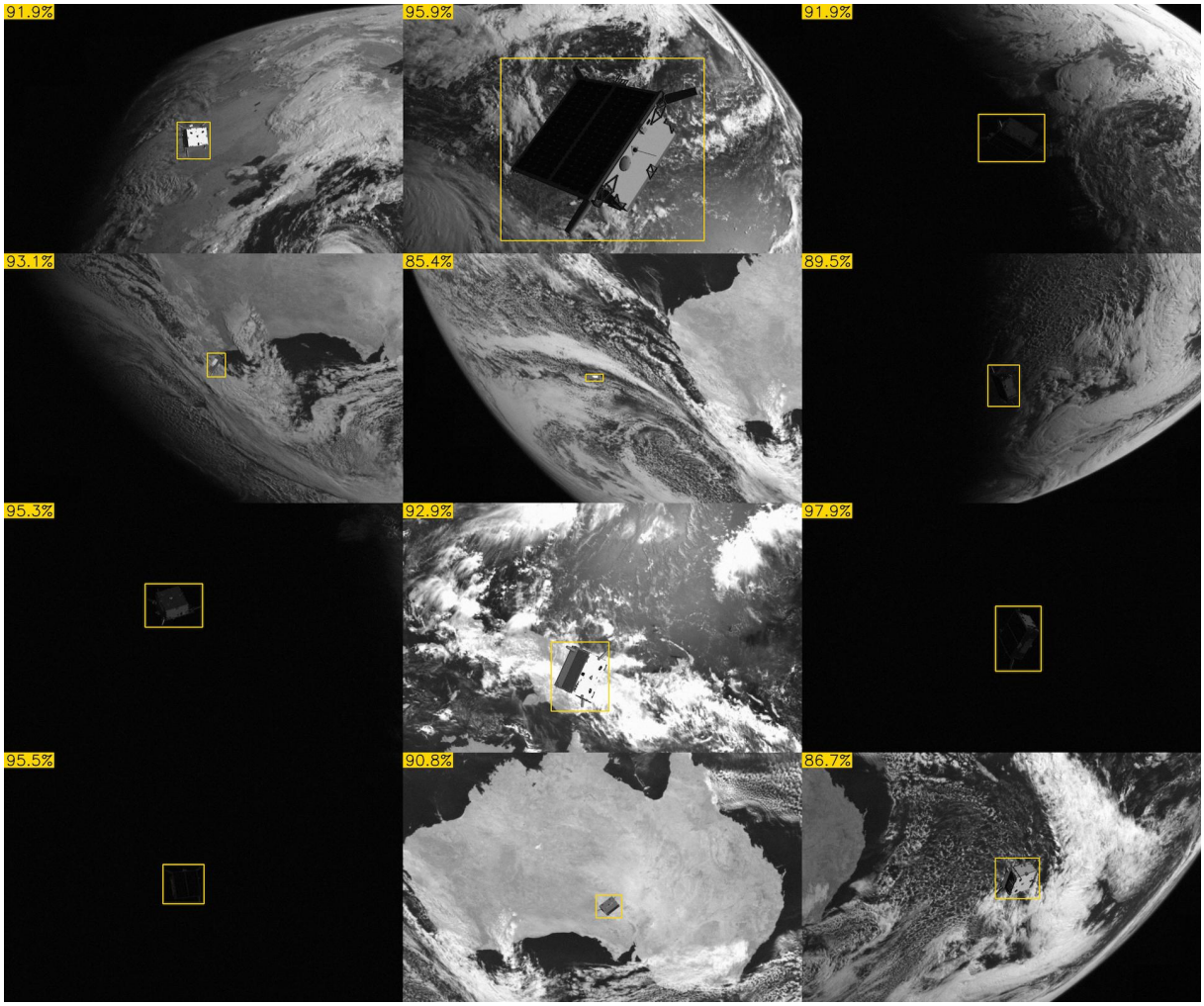


Figure 3.6: *SLN prediction on 9 test images with Earth background*

In Figure 3.6 inference was run on images characterized by the presence of Earth in

the background (a few of these images were rendered in eclipse condition). The excellent robustness to the Earth's presence in the FoV appears evident. This is true in a wide variety of lighting conditions, even in cases in which the S/C is very distant (~ 30 m) and poorly illuminated. In some cases Tango appears hard to distinguish from the patterns in our planet, also to the human eye.

Although the test labels of real images are not available, it was decided to run `YOLOv5` on this distribution as well, to figure out from simple visual inspection how well the model generalizes to actual imagery. The experiment was successful and the CNN appeared to identify correctly the RoI in the entire set of images, with very high confidence scores. In Figure 3.7 the results obtained for 6 random images are reported. As it can be seen, the detection performed nominally also in cases in which part of the S/C is outside of the image frame. Looking at the top-right image in Figure 3.7, one may point out that a portion of the inter-satellite link antenna pointing rightward lies slightly outside of the detected BB. Indeed, this very slender antenna, unlike the three larger ones, is not part of the wireframe model. It was decided not to include the tip of this antenna in the set of keypoints, because it is not in general a strong visual feature and it would be hard to detect at large distances. This means that our neural network was trained to ignore the fact this tiny antenna may in some cases contain an extremal point, that should in principle further extend the BB.

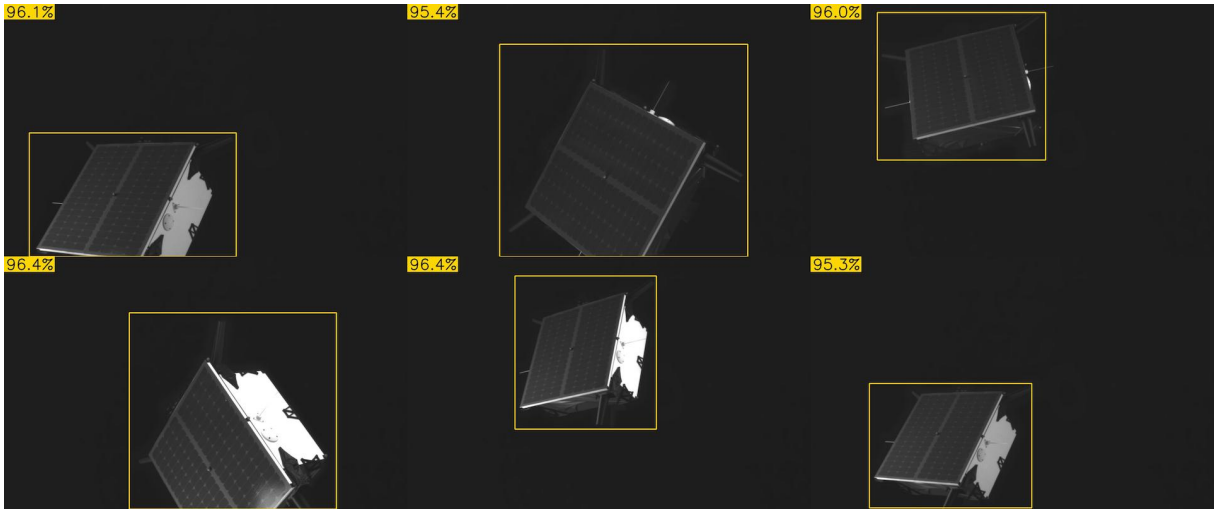


Figure 3.7: *SLN prediction on 6 test images of the mock-up spacecraft*

3.2 Landmark Regression Network

The Landmark Regression Network (LRN), which is the second image processing subsystem in our pipeline, receives as input the grayscale RoI detected by SLN. The input size of LRN is again 416×416, which means that RoIs whose largest side is greater than 416 pixels will undergo downscaling.

The unprecedented accuracy demonstrated by `HRNet` in the field of human pose estimation led to the implementation of this model in our architecture. The CNN has been trained to regress 11 heatmaps with a size of 416×416, corresponding to the 11 semantic keypoints specified in Figure 3.2. The final predicted landmark locations are then obtained

as the individual peaks in each heatmap, which will appear as 2D pseudo-Gaussians.

In [Sun19a] two different versions of the HRNet model are presented, which were named HRNet32 and HRNet48. The numbers 32 and 48 indicate versions of the network having respectively 32 and 48 channels in the highest-resolution subnetworks in the last three stages. The performance metrics achieved by these two versions on the MS COCO test set are compared in Table 3.3.

Table 3.3: *Performance of HRNet32 and HRNet48 on the MS COCO test set [Sun19a]*

Model	# parameters	FLOPs	AP^{95}_{50}	AP_{50}	AP_{75}
HRNet32	28.5M	16.0B	74.9	92.5	82.8
HRNet48	63.6M	32.9B	75.5	92.5	83.3

It was decided to implement the HRNet32 version, given a performance level quite close to the larger version of the network. The latter appears slightly superior, but this comes at the expense of more than twice the number of FLoating-Point Operations compared to the smaller model.

A major difference with respect to the implementation of HRNet in the pose estimation pipeline proposed by the UniAdelaide team is that the latter makes use of a very large 768×768 size of the input window.[(2)] Such a choice was clearly aimed at achieving the best possible score in the competition, but currently appears unrealistic for real-time spaceborne execution. In addition, it can be observed that a size of 416 pixels of the non-resized RoI, given our camera intrinsics and the size of Tango, corresponds to a distance of about $8 \div 10$ m. On the contrary, a 768 pixel size is achieved at $4 \div 5$ m, which means that only a small fraction of the SPEED images would benefit from this very high resolution processing, i.e. those at distances ≤ 5 m.

Due to the above mentioned motivations, a more reasonable 416×416 input window was selected for LRN. Indeed, as we will later discuss in Chapter 4, this already guarantees centimeter-level and sub-degree accuracy in the pose estimated on close-to-mid range images of SPEED.

3.2.1 Training

The Ground Truth labels have been annotated by projecting onto the image frame the 11 keypoints defined in the 3D wireframe model of Tango, based on the known training poses and using Equation (2.12). The corresponding GT training heatmaps are then set to 2D Gaussians with 1-pixel standard deviation and mean value in correspondence of the projected landmark coordinates.

Despite the use of high-end GPUs on the Google Colab platform, the training of this architecture turned out to be very expensive and has only been carried out for 80 epochs. ADAM optimization has been used, with a batch-size of 16 images, $\beta_1 = 0.9$, $\beta_2 = 0.99$, learning rate equal to 10^{-3} and a weight decay of 10^{-4}.

The loss function for the ith image is defined as the mean squared error between the regressed heatmap $\hat{\mathbf{H}}$ and the corresponding Ground Truth $\mathbf{H}$, averaged over all the n landmarks lying inside the image frame:

[(2)] i.e. 3.4 times the number of pixels compared to our pipeline

$$\mathbf{L}_{\text{MSE}}^{(\text{i})} = \frac{1}{n}\sum_{j=1}^{n} v_j^{(\text{i})} \cdot \left[\hat{\mathbf{H}}_j^{(\text{i})} - \mathbf{H}(\mathbf{p}_j^{(\text{i})})\right]^2 \tag{3.2}$$

The loss computed for an entire mini-batch is simply the average over all images in the batch, namely $\mathbf{L}_{\text{MSE}} = \frac{1}{m}\sum_{i=1}^{m} \mathbf{L}_{\text{MSE}}^{(i)}$.

3.2.2 Performance evaluation

The performance of our LRN has been evaluated in terms of the standard metrics introduced in Section 2.1.4. The 10 precision-recall curves in Figure 3.8 are computed in correspondence of the 10 equally spaced Object Keypoint Similarity (OKS) thresholds, from 0.5 to 0.95. The Average Precision is then calculated for each of these curves, according to Equation (2.7), from which we eventually obtain the global metric $\text{AP}_{50}^{95} = 98.97\%$. This indeed indicates an excellent regression accuracy, obtained after only 80 training epochs.

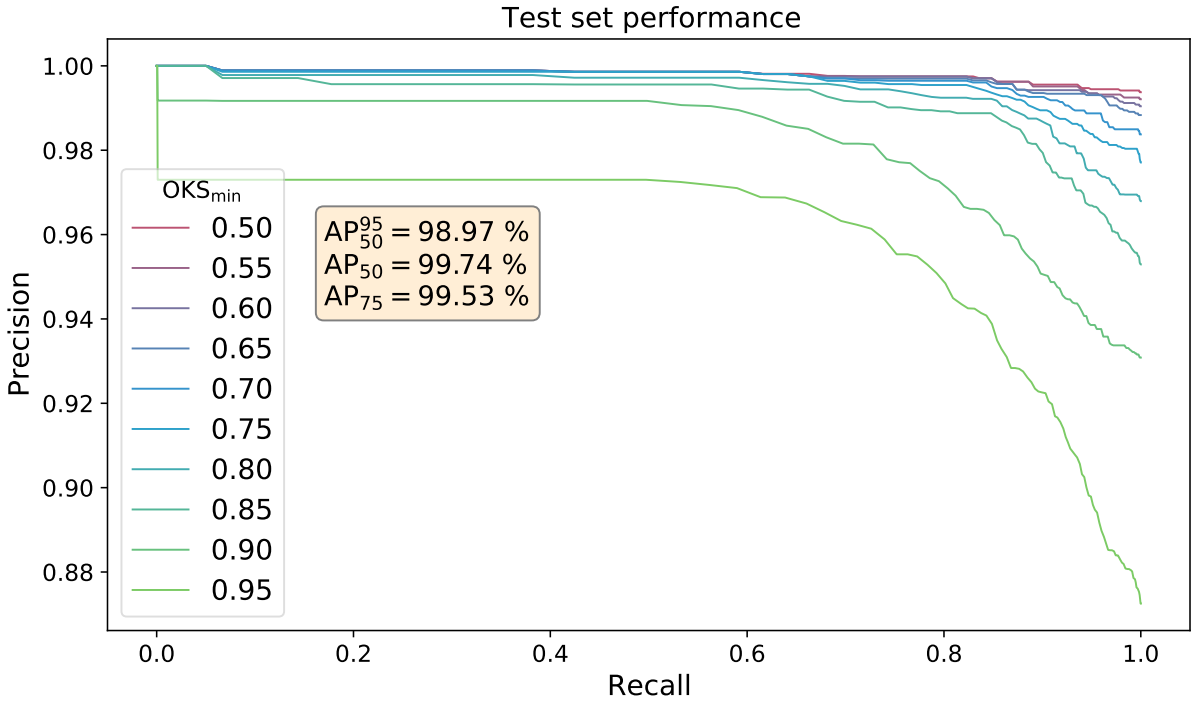


Figure 3.8: *Precision-recall curves, in correspondence of different OKS thresholds*

In Figure 3.9 we reported the 11 regressed heatmaps in correspondence each of the two randomly picked input images. All the heatmaps in this figure are characterized by a very sharp peak in correspondence of the landmark location.

It is worth highlighting that the capabilities of our CNN are not limited to locating a landmark by identifying in the image the feature associated to it (e.g. a specific edge). Indeed, the network is also able to accurately estimate the position of a keypoint, whenever the related semantic feature is occluded by some other portion of the spacecraft itself: this is the case, for instance, of point B3 in the bottom image or point S2 in the top image of Figure 3.9.

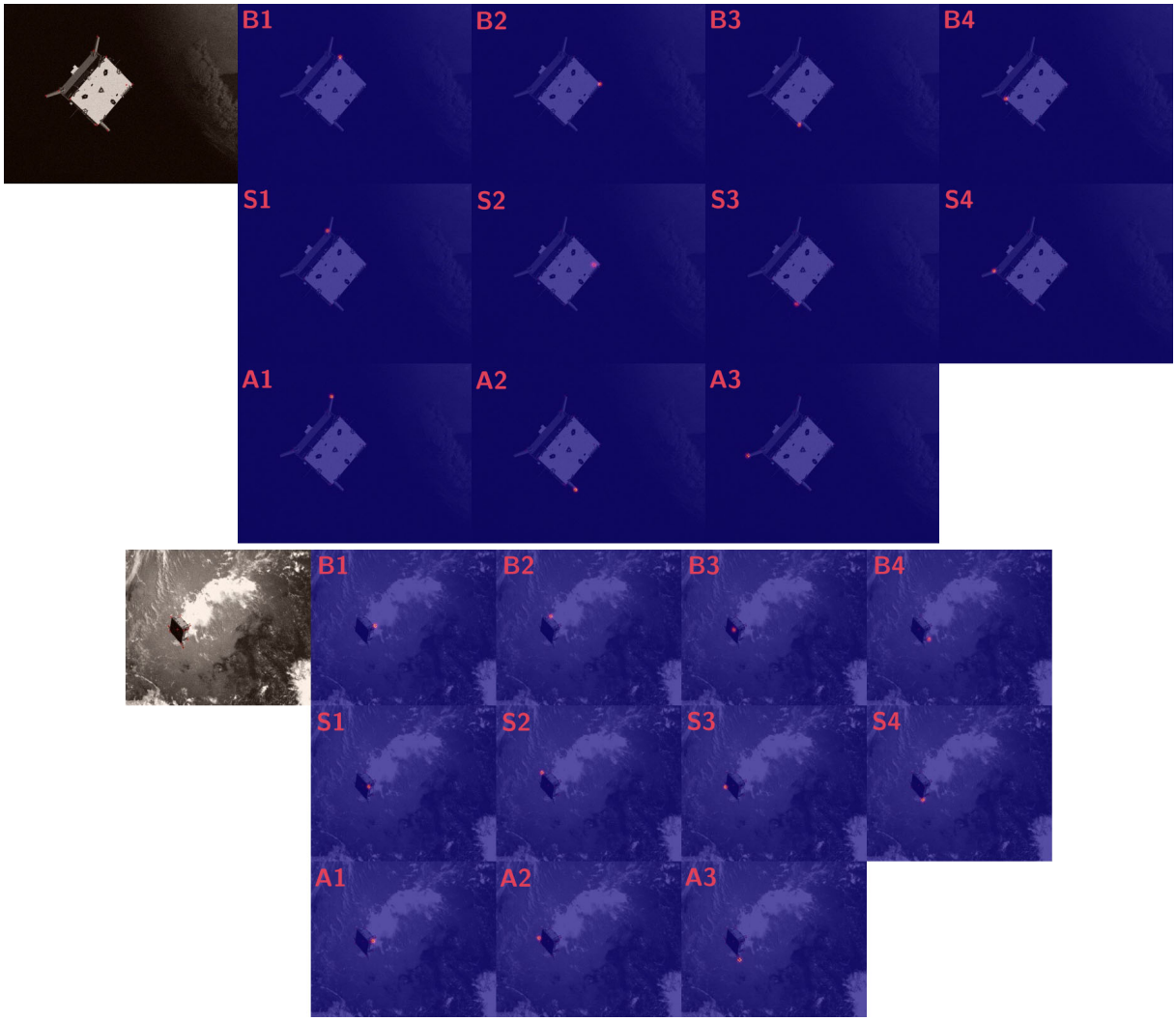


Figure 3.9: *Examples of regressed heatmaps*

3.3 Pose solver

The pose solver is the third and last subsystem of our Relative Pose Estimation Pipeline, that identifies the best pose fit, based on the keypoints detected by LRN. The pose solver also leverages Bounding Box information (i.e. the output of SLN) to identify the presence of outliers among the considered keypoints, and partially correct the resulting wrong pose estimate.

3.3.1 Keypoint selection

The availability of a heatmap that provides a confidence score for a given detected landmark can be leveraged to filter out potential outliers, which may cause a pose solver to diverge or to output a completely wrong pose. In particular, two hyper-parameters have been tuned, in order to find a good compromise between rejecting potential outliers and retaining a sufficient number of points. Regarding this last goal, it is clearly beneficial

in terms of accuracy to over-constrain the 3D model, as long as we keep adding precise keypoint detections. The hyper-parameters that have been consequently selected are:

- # landmarks$_{\text{min}}$: size of the minimal set of landmarks, i.e. the minimum number of the highest-confidence detected landmarks to be always retained, independently of their associated scores
- confidence$_{\text{min}}$: minimum confidence required to retain any landmark in addition to the minimal set

This means that, in general, only a subset of the 11 keypoints will be effectively fed to the pose solver.

The optimal tuning of the two above mentioned hyper-parameters will be discussed in Chapter 4.

3.3.2 Initial pose estimation and refinement

After discarding low-confidence landmarks, the remaining ones are fed to the EPnP algorithm (Section 2.2.2), which computes a first pose estimate and does not require any initial guess. EPnP is characterized by a weak robustness to the presence of outliers among the input keypoints. However, if no outliers are present, the resulting pose estimate turns out to be quite accurate.

At this point, our algorithm checks whether or not the estimated pose is consistent with the BB detected by SLN. Indeed, it was concluded that, after proper training, we can "trust" SLN more than LRN, just because the former actually performs a simpler task. Thus, whenever an inconsistency is found between the two subsystems, it is reasonable to believe that LRN is to blame. In other words, whenever the projection of Tango's 3D model (based on the initial pose estimate) is inconsistent with the detected BB, this is very likely due to the presence of one or more outliers among the retained landmarks, which translates into a completely wrong pose computed by EPnP.

If no inconsistency is found in the output of EPnP and the reprojection error is acceptable, this initial pose is refined using the Levenberg-Marquardt Method (Section 2.2.1), that iteratively minimizes the reprojection error.

3.3.3 Outlier identification & translation correction

If, on the contrary to what previously described, a pose outlier is flagged by our algorithm, we will then partially correct the pose by replacing the translation vector output by EPnP with an approximate yet robust estimation.

In order to identify a possible pose outlier, an approximate translation vector $\tilde{\mathbf{t}}_{\text{C/B}}$ is first of all computed, by exploiting a RoI-based estimation. This method leverages the knowledge of the characteristic length L_C of the spacecraft, along with the BB's center $(P_x^{\text{BB}}, P_y^{\text{BB}})$ and diagonal length d_{BB} that are detected by SLN. The aforementioned dimensions and coordinates are indicated in Figure 3.10.

Given our camera intrinsics, we are able to relate the size of our real-world S/C to the corresponding size in the image frame, hence obtaining the following expression for the distance between the camera-fixed and the body-fixed frames

$$\tilde{t}_{\text{C/B}} = \frac{f/\rho_u + f/\rho_v}{2} \cdot \frac{L_C}{d_{\text{BB}}} \tag{3.3}$$

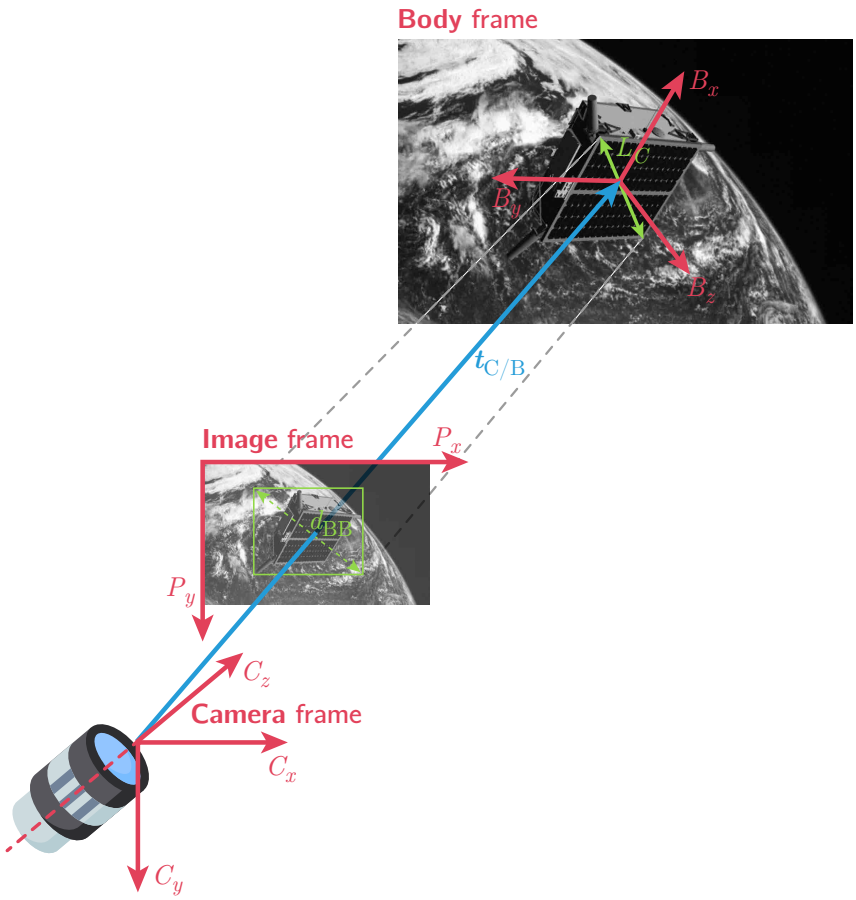


Figure 3.10: *Reference frames and RoI*

We may similarly compute also the azimuth and elevation angles, α and β, as

$$\begin{aligned} \alpha &= \arctan\left(\frac{P_x^{\mathrm{BB}} - u_0}{f/\rho_u}\right) \\ \beta &= \arctan\left(\frac{P_y^{\mathrm{BB}} - v_0}{f/\rho_v}\right) \end{aligned} \tag{3.4}$$

At this point, a coarse estimate of the camera-to-body translation vector may be derived as

$$\tilde{\mathbf{t}}_{\mathrm{C/B}} = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & \sin\beta \\ 0 & -\sin\beta & \cos\beta \end{bmatrix} \begin{Bmatrix} 0 \\ 0 \\ \tilde{t}_{\mathrm{C/B}} \end{Bmatrix} \tag{3.5}$$

An outlier will be flagged whenever any of the following conditions is encountered.

- The projected geometric center of the S/C, according to the pose estimated by EPnP, has a $> 50\%$ offset[(3)] from the BB center

$$\frac{|\hat{p}_x^{\mathrm{c}} - P_x^{\mathrm{BB}}|}{w_{\mathrm{BB}}} > 0.5 \quad \text{or} \quad \frac{|\hat{p}_y^{\mathrm{c}} - P_y^{\mathrm{BB}}|}{h_{\mathrm{BB}}} > 0.5$$

[(3)] the pixel offset is normalized with respect to the BB width and height

- Large mismatch between the distance estimated by EPnP, $\hat{t}$, and the one obtained from the RoI-based approximation, $\tilde{t}$

$$\left|\frac{\hat{t}-\tilde{t}}{\tilde{t}}\right| > 75\%$$

- Medium distance mismatch and low average confidence of the retained landmarks

$$\left|\frac{\hat{t}-\tilde{t}}{\tilde{t}}\right| > 15\% \quad \text{and} \quad \text{confidence}_{\text{avg}} < 50\%$$

- Medium distance mismatch and high relative reprojection error

$$\left|\frac{\hat{t}-\tilde{t}}{\tilde{t}}\right| > 15\% \quad \text{and} \quad \frac{E_{\text{repr}}}{d_{\text{BB}}} > 10\%$$

Whenever a pose outlier is flagged by our algorithm, the initial estimate of the translation vector will be replaced by the corresponding RoI-based approximation $\tilde{\mathbf{t}}_{\text{C/B}}$.

Chapter 4

Results

4.1 Error metrics

Prior to presenting the results achieved by our Relative Pose Estimation Pipeline, we will dwell on the definition of the error metrics that allow us to evaluate the performance of our architecture on the Spacecraft PosE Estimation Dataset (SPEED).

In particular, both translation and rotation errors can be classified based on two main features.

- Absolute vs. normalized.

  Absolute error: we will denote it with the capital E and it simply indicates the difference between the estimated translation/rotation and the corresponding Ground Truth (GT).

  Normalized (or relative) error: we will denote it with the lowercase e and it is normalized based on the GT distance and angular size of the S/C in a given image.

- Mean vs. Median.

  Mean error: in order to evaluate the performance on the entire dataset, we simply compute the average of a given metric over all images; this value will be strongly influenced by the presence of outliers.

  Median error: global performance on the whole dataset is evaluated by computing the median over all images of the error. In our case, median error is actually more representative of the accuracy as compared to mean: the reason is that we experienced the presence of very few outliers, which are nevertheless characterized by an error that is orders of magnitude larger than nominal detections.

4.1.1 Translation error

The absolute translation error for a given image is obtained as

$$E_t = \left\| \hat{\mathbf{t}}_{\mathrm{C/B}} - \mathbf{t}_{\mathrm{C/B}} \right\| \tag{4.1}$$

which can be easily normalized if we divide it by the GT distance:

$$e_t = \frac{E_t}{\left\|\mathbf{t}_{\mathrm{C/B}}\right\|} \tag{4.2}$$

4.1.2 Rotation error

Absolute error

The absolute rotation error might be measured in two different fashions.

In terms of quaternion error, which represents the overall attitude error with a single scalar metric, it will be computed as

$$E_{\mathrm{q}} = 2 \cdot \arccos\left|\mathbf{q} \cdot \hat{\mathbf{q}}\right| \tag{4.3}$$

In terms of Euler angles, the error will be obtained as the difference between a given estimated Euler angle and the corresponding GT

$$E_{\theta_x} = |\hat{\theta}_x - \theta_x|, \qquad E_{\theta_y} = |\hat{\theta}_y - \theta_y|, \qquad E_{\theta_z} = |\hat{\theta}_z - \theta_z| \tag{4.4}$$

Normalized error

The main weakness of the SLAB score defined in Equation (1.1) is that, although it accounts for distance-normalization in its translation component, it does not account for normalization of the rotation error component. This means that the same absolute angular error has the same exact effect upon measured performance, independently of whether that occurs in correspondence of a close-range image or at a distance in which the RoI is just a very small fraction of the entire image area.

This led us to introduce a normalized version of the quaternion error defined in Equation (4.3), which accounts for the angular size of the object relative to the FoV of the camera. In particular, an object's angular size is defined as the angle measured between the two lines of sight corresponding to opposite sides of the object. In our case, we will consider the angle associated with the diagonal size of each GT Bounding Box.

If we resort to the pinhole camera model, which is represented in Figure 4.1, the diagonal angular size associated with the spacecraft can be computed as

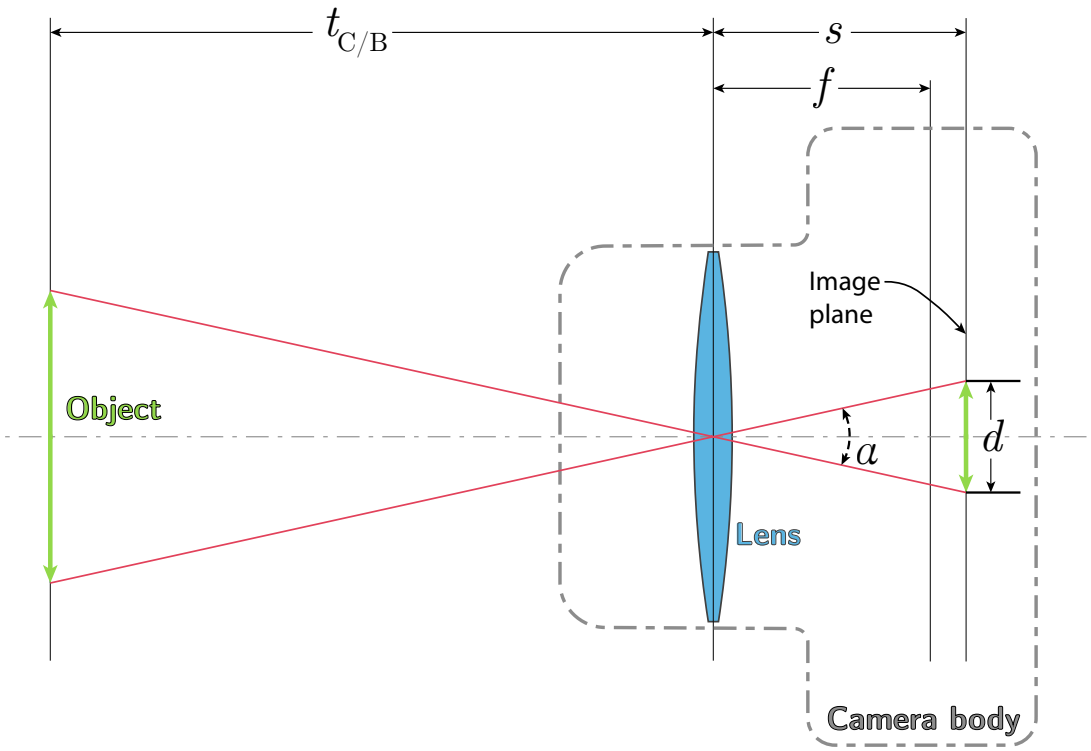


Figure 4.1: *Pinhole camera model*

$$\alpha = 2 \cdot \arctan \frac{\rho \cdot d_{BB}}{2s} \qquad \text{where } s = \frac{f \cdot t_{C/B}}{t_{C/B} - f} \tag{4.5}$$

However, we may approximate the distance between the lens and the image plane as $s \approx f$. In particular, this relation becomes exact whenever the lens is set for infinity focus. In Equation (4.5), $\rho \equiv \rho_u \equiv \rho_v$ is the pixel pitch [μm/px], d_{BB} is the diagonal length of the BB [px], while f is the focal length [mm].

Note that, in order to normalize the rotation error, we need to divide it by a quantity that increases as the attitude gets harder to estimate. We will therefore divide the quaternion error defined in Equation (4.3) by the portion of the diagonal FoV of the camera that is not occupied by the spacecraft, which reads

$$e_q = \frac{E_q}{\text{FoV}_{diag} - \alpha} \tag{4.6}$$

where, considering an $N_u \times N_v$ image, the diagonal FoV can be obtained as

$$\text{FoV}_{diag} = 2 \cdot \arctan \frac{\rho \cdot \sqrt{N_u^2 + N_v^2}}{2f}$$

4.1.3 Pose error

The overall pose error is simply measured as the sum of the translation and rotation errors.

The SLAB score, which has already been defined in Equation (1.1), measures the total error as the mean of $\left(e_t^{(i)} + E_q^{(i)}\right)$ computed over all the N test images.

After having highlighted the weaknesses the aforementioned metric, we are hereby proposing an alternative performance index that we deem to be more relevant. It has been called the Median Normalized Pose Error (MNPE):

$$e_{\text{MNP}} = \underset{i=1}{\overset{N}{\texttt{median}}}\left(e_t^{(\text{i})} + e_q^{(\text{i})}\right) \tag{4.7}$$

where e_t and e_q are defined in Equations (4.2) and (4.6), respectively.

4.2 Optimal keypoint rejection

Grid-search optimization was run to seek for the combination of the two hyper-parameters defined in Section 3.3.1 that yields the lowest possible MNPE. Recall that these two thresholds determine which of the keypoints detected by the Landmark Regression Network (LRN) will be discarded and which ones will instead become the input of the pose solver. The obtained results are given in Figure 4.2.

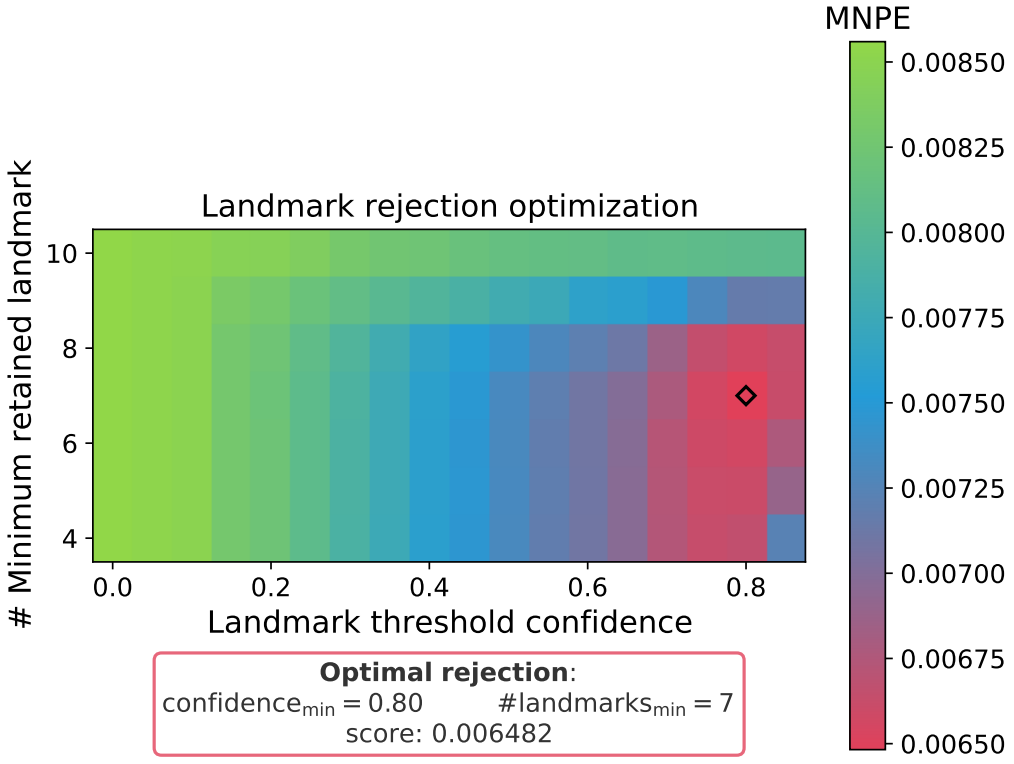


Figure 4.2: *Optimization of the keypoint rejection process*

As it can be noted from the plot, the minimal set of landmarks shall contain at least 4 points. This due to the fact that the EPnP algorithm requires $n \geq 4$ points to compute a solution. In addition, although the total number of keypoints in the wireframe model amounts to 11, the corresponding hyper-parameter that defines the size of the minimal set of landmarks is only varied between 4 and 10: this is because the condition in which we retain all the 11 landmarks is already included in the case of zero threshold confidence.

The beneficial effect of filtering out low-confidence landmarks is evident from Figure 4.2. The threshold confidence alone allows a 24% reduction of the MNPE, compared to retaining

all predicted keypoints. In particular, we experienced that accuracy increases monotonically as we increase the threshold confidence up to 0.8, but larger values of such threshold become too restrictive, which causes many precise keypoint detections to be discarded. Introducing a second hyper-parameter that defines the minimal size of the set of retained keypoints allows to further reduce the error, although the effect is less evident.

All the results presented in the remainder of this discussion have been obtained in correspondence of the optimal values: $\text{confidence}_{\text{min}} = 0.8$ and $\#\text{landmarks}_{\text{min}} = 7$.

4.3 Performance evaluation

Our Relative Pose Estimation Pipeline, achieved a SLAB score of 0.04627 on our test set. This means that, based on the official leaderboard of the SLAB/ESA Pose Estimation Challenge reported in Table 1.2, our architecture would virtually score 3rd place, hence outperforming the SLAB baseline.

Indeed, this performance level has been confirmed by participating in the post-mortem competition, which is still running on the ESA website. Figure 4.3 has been printed from the website of the post-mortem competition[(1)] and reports the score achieved by the 5 top teams, as of November 24th 2020.

Timeline

The competition is in progress.

Leaderboard

Name	Submissions	Last Submission	Best Submission	Real Image Score	Best Score
	competition winner UniAdelaide			0.36340645622528017	0.00864899489025079
arunkumar04	5	June 11, 2020, 2:09 p.m.	June 11, 2020, 3:22 a.m.	0.2897316198709755	0.00965354346853769
UT-TSL	1	July 29, 2020, 8:46 p.m.	July 29, 2020, 8:46 p.m.	0.29182320619186036	0.040888808313561543
massimo.piazza	1	Nov. 20, 2020, 4:55 p.m.	Nov. 20, 2020, 4:55 p.m.	0.12725700558317155	0.046734132793689716
haoranhuang	45	Nov. 23, 2020, 9 a.m.	Sept. 28, 2020, 8:29 a.m.	0.2340254300626748	0.05931408717012119
julien_poly	5	Nov. 10, 2020, 8:46 p.m.	Oct. 10, 2020, 12:28 a.m.	0.14404709940008242	0.06703709707262259

Figure 4.3: *Top 5 participants of the post-mortem competition*

It can be noted that our architecture attained a SLAB score, on the synthetic original test set of SPEED, equal to 0.04673. This corresponds to a performance level that is practically identical to the one estimated on our test set. The score on the synthetic distribution is here labeled as "best score", while the "real image score" indicates the accuracy achieved on the 300 real images of a mockup of the Tango spacecraft.

[(1)] https://kelvins.esa.int/pose-estimation-challenge-post-mortem/leaderboard/

Feb. 1, 2019, 5 a.m. UTC Timeline July 1, 2019, 4 a.m. UTC

The competition is over.

Virtual placement of our architecture

Results

Rank	Name	Real Image Score	Best Score
1	UniAdelaide	0.3752442418711471	0.009449622064660844
2	EPFL_cvlab	0.11397767001637173	0.02153775817984222
3	pedro_fairspace	0.1554876108763784	0.057050185272129426
4	stanford_slab	0.3950914435276558	0.06262229611857424
5	Team_Platypus	1.7201238117705309	0.07028457489821285

Figure 4.4: *Top 5 participants of the original competition (Feb - Jul 2019)*

Figure 4.4 reports instead the top 5 participants (out of 48 individuals/teams) of the original competition.[(2)] By comparing the results it can be seen that, in terms of synthetic score, only two of these 48 participants outperformed our architecture. In addition, if we were to merge the leaderboards of the two competitions (59 overall participants), only the `EPFL_cvlab` team would achieve a better score on the real dataset (0.11398 vs. 0.12726).

We should recall that our synthetic test set is obtained by re-partitioning the original training set of SPEED. If we were to train our architecture on the full original training set and evaluate performance on the original test set, we should in principle expect a slight increase in performance. According to the so called "large data rationale", which is a common paradigm in the ML framework, we are supposed to obtain an improvement in accuracy as we increase the number of training examples.[(3)] In our case, if the labels of the original test set were available, we would no longer need to re-partition the original training set: this would translate into a performance improvement, despite making training more expensive.

In Table 4.1 we reported the most important performance metrics attained by our architecture on our test set.

It can be immediately noticed that there is a substantial difference between mean and median error. In particular, the latter is typically ~ 3 times smaller, both in terms of translation and rotation errors. This immediately highlights the presence of pose outliers, which are small in number yet with an error that is orders of magnitude larger compared to the extremely accurate detections that nominally take place.

This difference between mean and median error is less pronounced for the t_x, t_y translation components. As we will analyze more in detail in the following sections, this behavior has been traced back to the successful RoI-based correction of the translation vector, at least in terms of (x, y) components. The correction of the relative distance component along the boresight direction proved extremely beneficial when completely wrong poses were detected, although there still exist a significant degree of uncertainty due to the fact that this approximate estimation of t_z is based on measuring the size of the detected RoI. At a given distance, the latter may nevertheless vary in a relatively wide

(2) https://kelvins.esa.int/satellite-pose-estimation-challenge/results/

(3) this is intuitively due to the fact that the larger the training set, the less likely to overfit it

range, depending on the S/C's attitude.

Table 4.1: *Global end-to-end performance of the RPEP*

	Absolute error	
	Mean	**Median**
E_t	10.36 cm	3.58 cm
$\mathbf{E}_t$	[0.52 0.56 10.25] cm	[0.24 0.27 3.50] cm
E_q	2.24°	0.81°
$\mathbf{E}_\theta$	[1.57° 0.84° 1.72°]	[0.52° 0.33° 0.34°]
	SLAB score = 0.04627	MNPE = 0.00648
	Standard deviation of the error	
$\sigma_{\mathbf{E}_t}$	[1.62 1.71 30.44] cm	
$\sigma_{\mathbf{E}_\theta}$	[8.92° 5.11° 10.82°]	
$\sigma_{\mathbf{e}_t}$	[0.001157 0.001093 0.014890]	
$\sigma_{\mathbf{e}_\theta}$	[0.022179 0.012689 0.026854]	

For what concerns rotation errors, the estimation of the Euler angle about the y-axis appears to be more accurate compared to the two other components. On the contrary, the θ_z rotation is the one affected by the largest degree of uncertainty and is also the one for which the gap between mean and median performance is most evident. It has been conjectured that this effect is closely related to the distribution of the chosen keypoints, relative to the geometric center of the S/C. Indeed, one may compute a quantity that is analogous to the moment of inertia of a set of points:[(4)] to each keypoint we assign a weight equal to its mean detection confidence across the entire dataset, while the square distance is computed with respect to an (x, y, z) frame having its origin at the geometric center of the S/C. This leads to the definition of the corresponding three quantities, whose values are $I_{xx} = 1.518$ m^2, $I_{yy} = 1.589$ m^2, $I_{zz} = 2.736$ m^2. It can be seen that I_{zz} is almost twice as larger as the two other "inertias". In addition note that, in correspondence of the same angular error, the reprojection error (measured in pixels) is higher for keypoints that are farther from the center of the S/C. Since the pose fit is chosen based on the minimization of the reprojection error, this translates into a pose that is more prone to satisfy a wrong constraint imposed by a wrong keypoint detection that is far from the center. In other words, whenever an outlier is present among the detected keypoints,[(5)] the estimated Euler angle about the axis associated with the highest keypoint-inertia will be particularly biased towards that outlier.

Table 4.1 also reports the 1-σ uncertainty of both absolute ($\mathbf{E}_t$, $\mathbf{E}_\theta$) and relative ($\mathbf{e}_t$, $\mathbf{e}_\theta$) error components.

4.3.1 Estimation uncertainty

A fundamental part of our analysis is the quantification of the uncertainty affecting our estimation, given the ultimate goal of this work of proposing a Relative Pose Estimation

[(4)] $I_{ii} = \sum_{l=1}^{N_{\text{points}}} m^{(l)} \left[(d_j^{(l)})^2 + (d_k^{(l)})^2 \right]$ where $i, j, k = (1, 2, 3), (2, 3, 1), (3, 1, 2)$

[(5)] e.g. a certain feature is mistaken for a different one, which is visually similar

Pipeline that can be embedded in a navigation filter. In Figures 4.5 and 4.6 we plotted the distribution of absolute and relative errors, respectively, over a $[-3\sigma, +3\sigma]$ range.

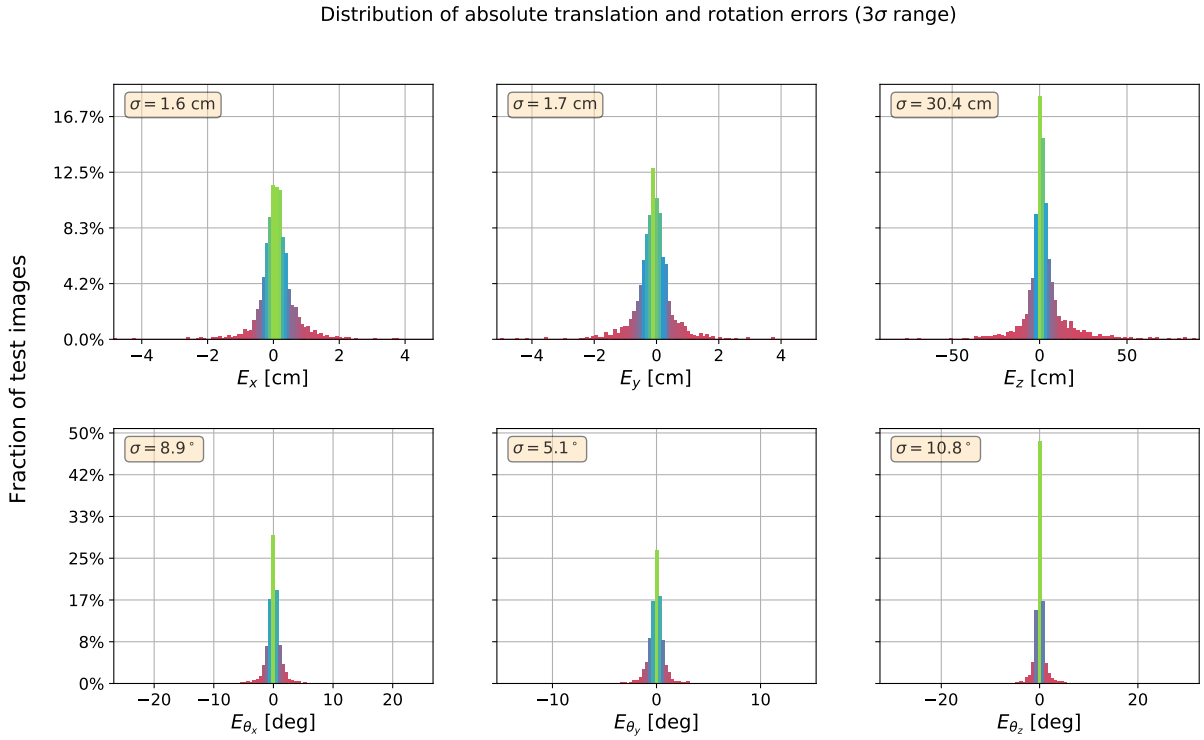


Figure 4.5: *Absolute error distribution*

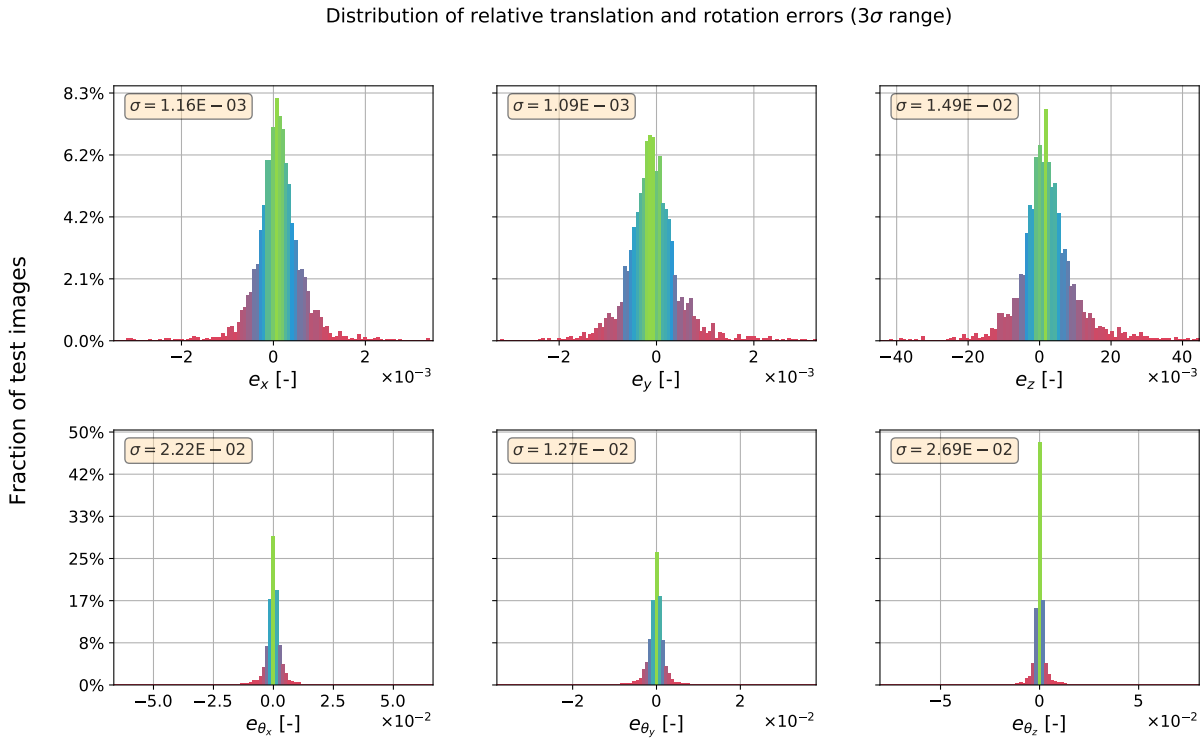


Figure 4.6: *Relative error distribution*

As one may expect, all 6 pose components are characterized by a normally distributed

error with zero mean. In each subplot, the y-axis indicates the fraction of test images associated with a given error bin. All these distributions were plotted using 101 bins.

Note that the two distributions of the lateral position errors are practically identical, both in terms absolute and relative errors. The translation error along the boresight direction is clearly much higher, with an uncertainty that is one order of magnitude larger compared to the two other translation components.

The fact that the distributions of the three rotation error components are similar and characterized by the same order of magnitude indicates a suitable choice of the semantic keypoints. Indeed, their selection should always be aimed at breaking as much as possible the symmetry of the structure, thus avoiding attitude ambiguities, while at the same time being associated with strong visually relevant features. Note that the error associated with θ_z is affected by a larger uncertainty, which may be explained by our previous conjecture.

4.4 Error distribution

In Figures 4.7 and 4.8 we reported the thresholds, in terms of the 6 error components, as a function of the test set fraction that does not exceed them. In both figures, the top plot provides the error distribution for the entire test set, while in the bottom plot we truncated the dataset at the best 95% fraction, just to zoom-in on the dataset portion that is unaffected by outliers.

In Figure 4.7 the error components are plotted in semi-logarithmic scale, due to the huge difference between lateral and boresight errors. At a given distance, the latter is one order of magnitude larger than lateral errors.

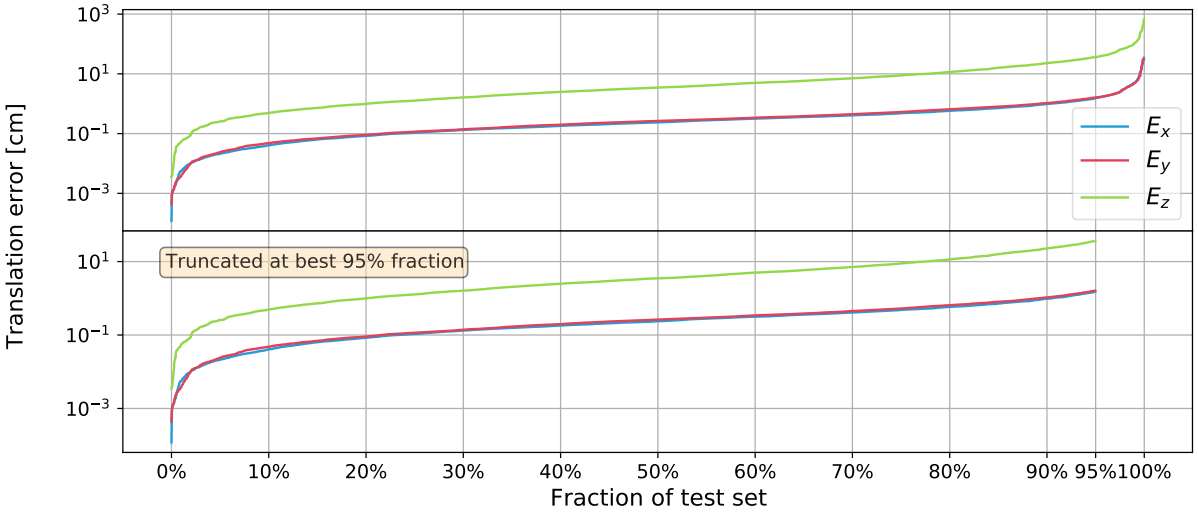


Figure 4.7: *Translation error distribution across the test set*

In Figure 4.8 the three Euler angle error components are similarly plotted, but in linear scale. It may be interesting to observe the trend of E_{θ_z}, which up to the best 70% fraction is practically coincident with E_{θ_y}. At that point, E_{θ_z} starts increasing sharply, compared to the two other components, and becomes the largest rotation error component of the 2% worst fraction of the test set. This behavior is clearly linked to the effect of outliers, whose presence, as we already explained in Section 4.3, has a more detrimental repercussion upon

the estimation of θ_z.

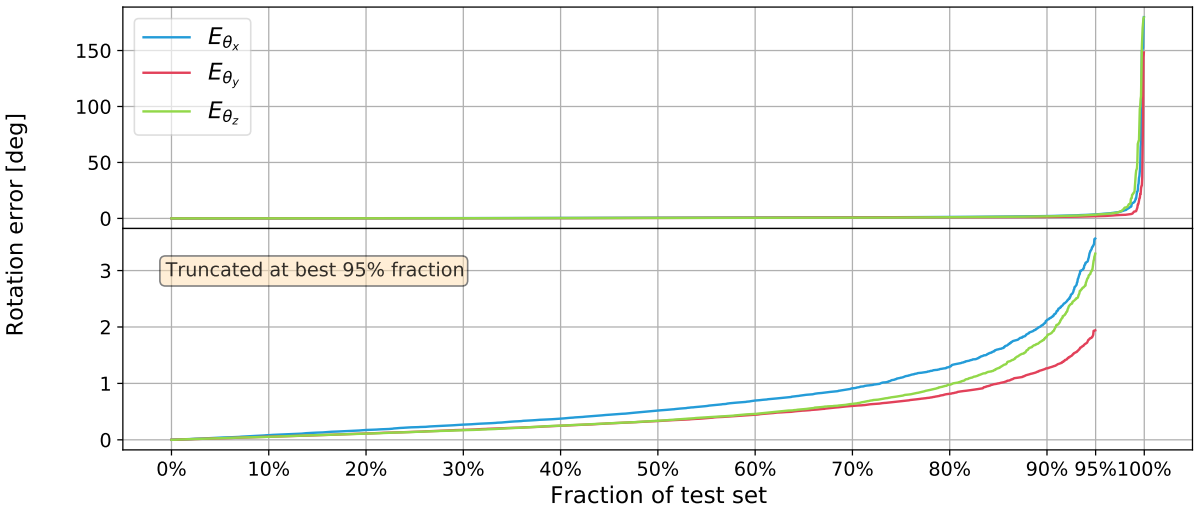


Figure 4.8: *Euler angle error distribution across the test set*

4.4.1 Effect of relative distance

At this point, we investigated the effect of the inter-spacecraft distance upon the accuracy of our RPEP. In Figures 4.9 to 4.11 such effect can be visualized in terms of various error metrics. In particular, all test set images were first of all sorted, based on relative distance, and then grouped into 30 batches of 80 images each. For each batch, the corresponding mean performance is plotted against the mean distance. The shaded region indicates the 1σ range uncertainty, i.e. between the 15.87% and 84.13% percentiles.

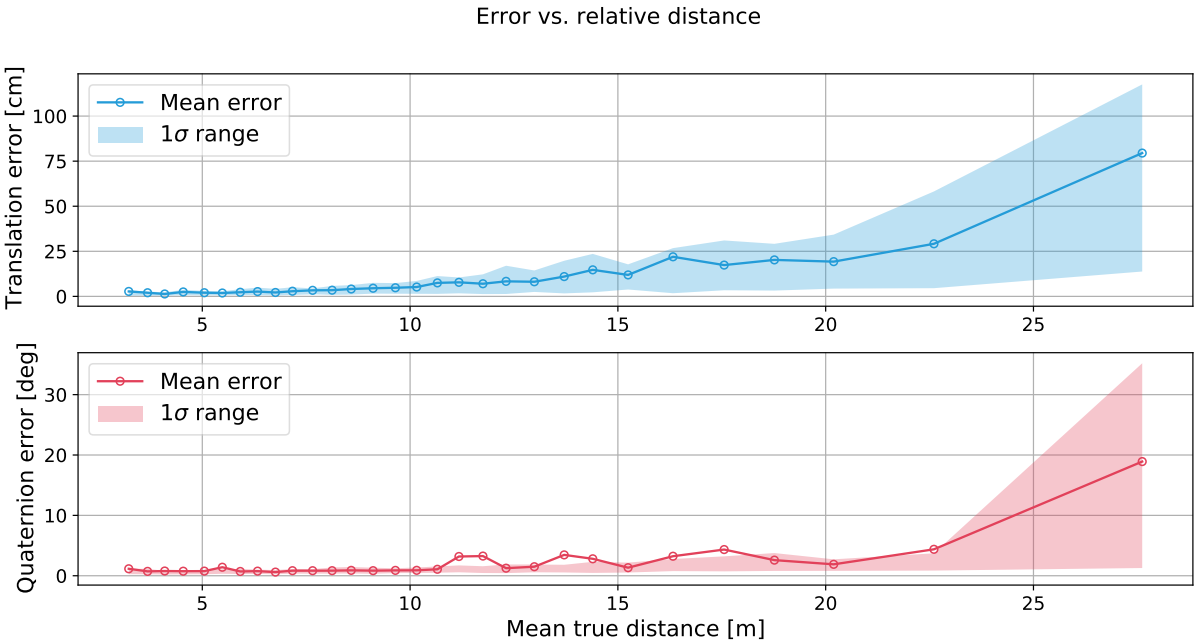


Figure 4.9: *Effect of inter-spacecraft distance upon absolute errors E_t and E_q*

In Figure 4.9 we analyzed this distance dependency for what concerns the absolute

translation and quaternion errors. From these two plots it is also evident the behavior that we anticipated in Section 3.2, according to which the performance of our pipeline remains practically constant ($E_t \sim 3$ cm and $E_q \sim 0.8°$) for all close-range images up to $8 \div 10$ m. To this threshold, it corresponds a size of the non-resized RoI of about 416 pixels, which means that for all images taken at lower distances there is a loss of information implicit in the downscaling to 416×416. In other words, for all images in which Tango is located at a distance $\leq 8 \div 10$ m, the degree of detail in the features that can be detected from the resized digital picture is exactly the same. For what concerns the attitude error, a sudden performance drop-off takes place at separations larger than 25 m. If we only consider image batches with mean distance ≤ 20 m, the highest batch-errors are $E_t = 22$ cm and $E_q = 4.3°$

In Figure 4.10 we performed a similar analysis, here visualizing the error breakdown in terms of translation components and Euler angles. For ease of visualization, we used a semi-logarithmic scale when plotting E_x, E_y, E_z. From the bottom plot it can be noticed that the sharp loss of attitude accuracy, when $t_{\mathrm{C/B}} > 25$ m, is mainly due to the E_{θ_z} component.

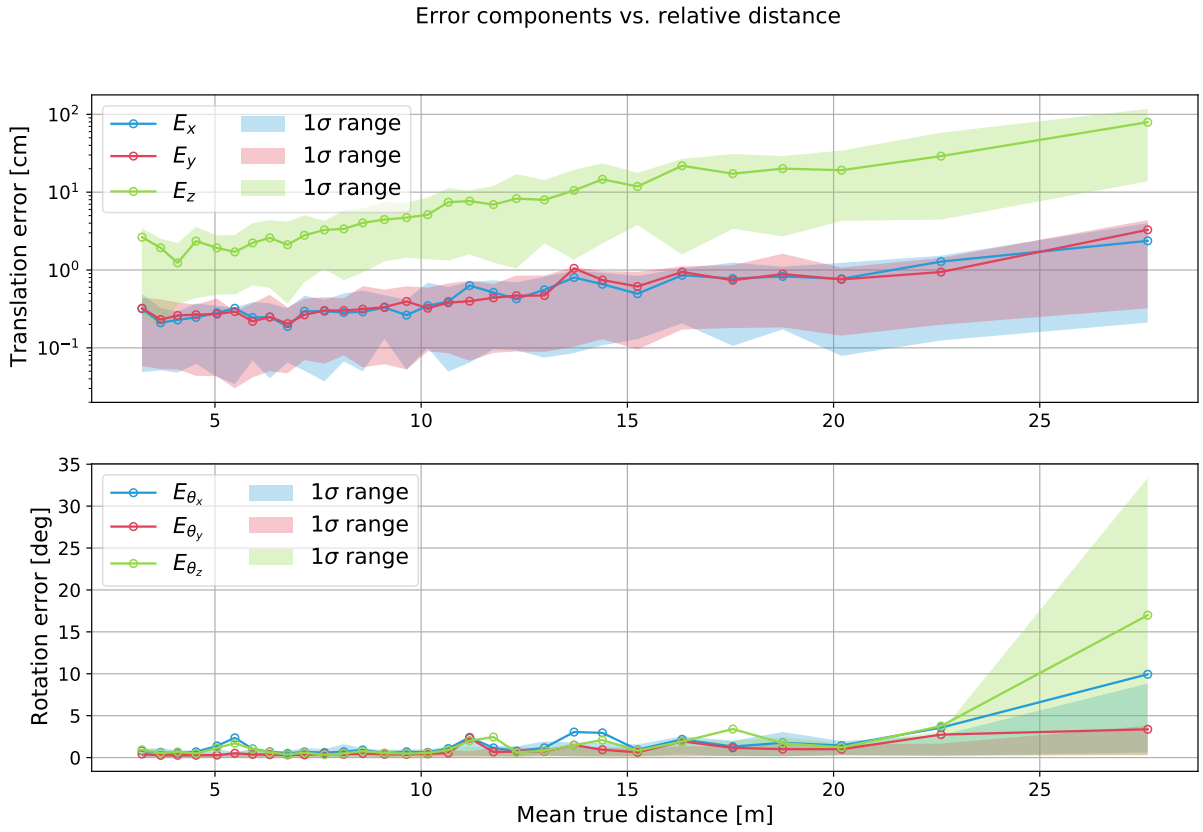


Figure 4.10: *Effect of inter-spacecraft distance upon absolute error components*

The distance dependency has been analyzed also in terms of global score and relative errors. The difference between the SLAB score and our MNPE is particularly evident from Figure 4.11. For a more direct comparison between the two metrics, in the bottom plot we used a slightly different definition of the Normalized Posed Error, compared to the one in Equation (4.7): we computed the mean over a batch of images, instead of the median. Indeed, it is immediately visible that in the SLAB score, the non-normalized quaternion error is up to one order of magnitude larger than the relative translation error,

which results into a score that is strongly biased towards attitude error. On the contrary, using a fully normalized score reduces this gap. This is therefore supposed to produce a more meaningful and consistent global evaluation metric.

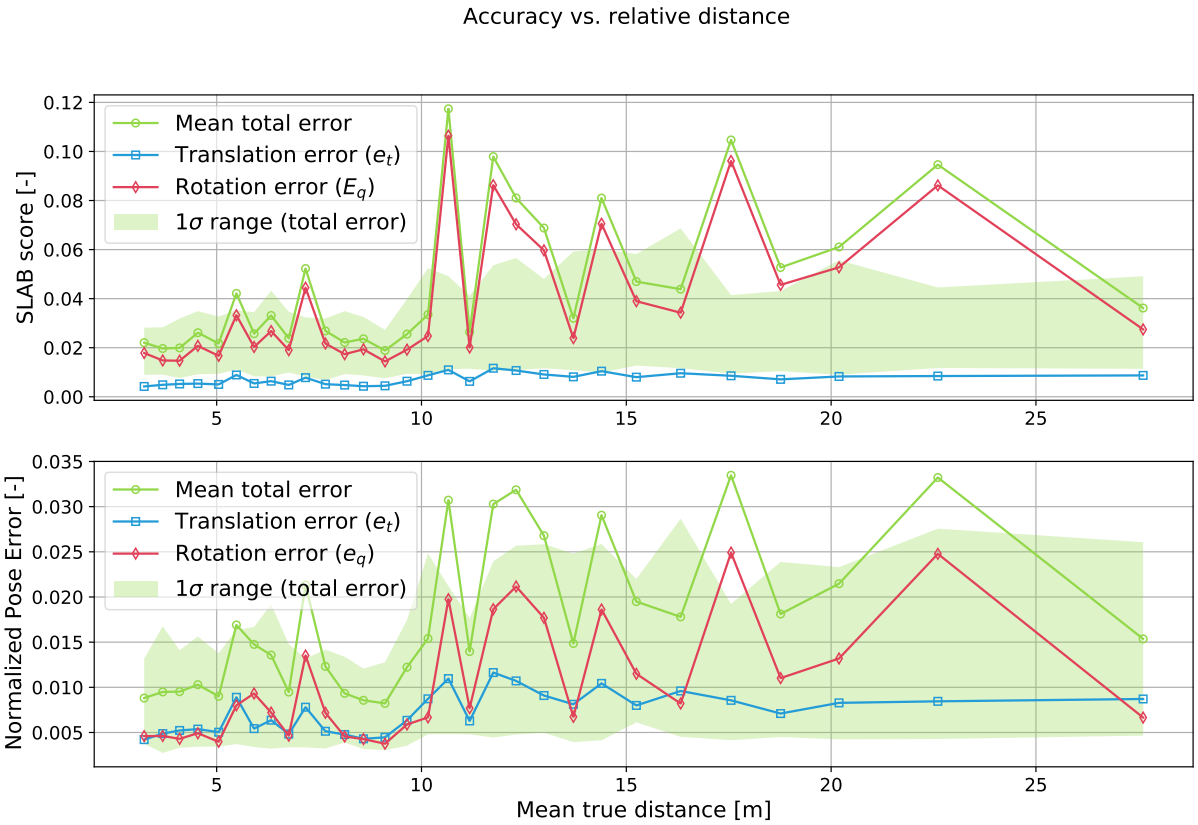


Figure 4.11: *Effect of inter-spacecraft distance upon SLAB score and Normalized Pose Error*

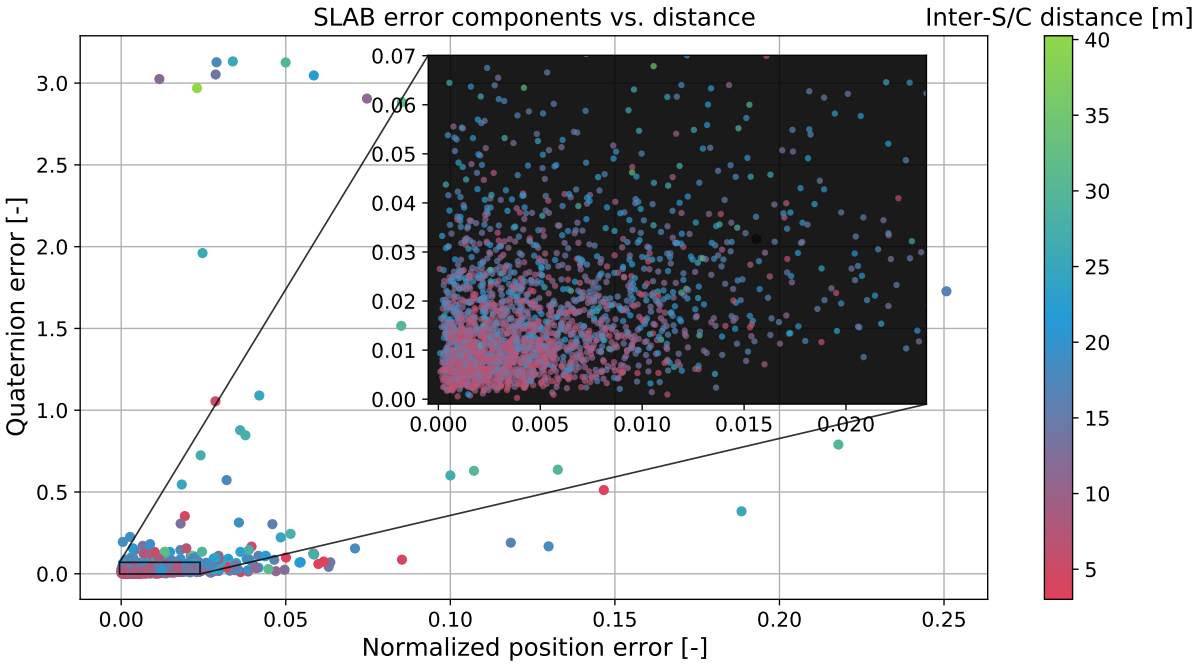


Figure 4.12: *SLAB error components of all test set images*

The difference between the orders of magnitude of the two error components that define the SLAB score is further stressed in Figure 4.12, in which we provided a scatter plot, representing the results obtained for all the 2400 test images. The color of each dot indicates the Ground Truth distance associated with each individual image. The close-up region corresponds to the 2σ rectangle, i.e. whose sides span over the 95.44% of the related error components.

4.4.2 Effect of the image background

Half of the images in our test set were rendered using a black background behind the S/C, while the other half was rendered with the presence of Earth in the image background, either in Eclipse condition or not. It is then clear that, despite all the actions taken during training to improve the robustness of our CNNs to a variable background, the presence of Earth in the image may still cause a performance degradation in our pipeline. This is especially true whenever the target is very far from the chaser. Indeed, it can be experienced that in long-range images with the presence of Earth in the background, SLN still works exceptionally well, but LRN may sometimes struggle at properly detecting all semantic keypoints.

The aforementioned performance drop is clearly visible in Figure 4.13. Here, all test set images were sorted based on their individual SLAB score. Each image is represented as a blue dot if it has a black background, otherwise, if the Earth lies inside the image frame we will plot a red dot. It can be observed that most of the images with a very low error have a black background. On the contrary, the right-hand side of this distribution is characterized by the increasing prevalence of images with Earth in the background.

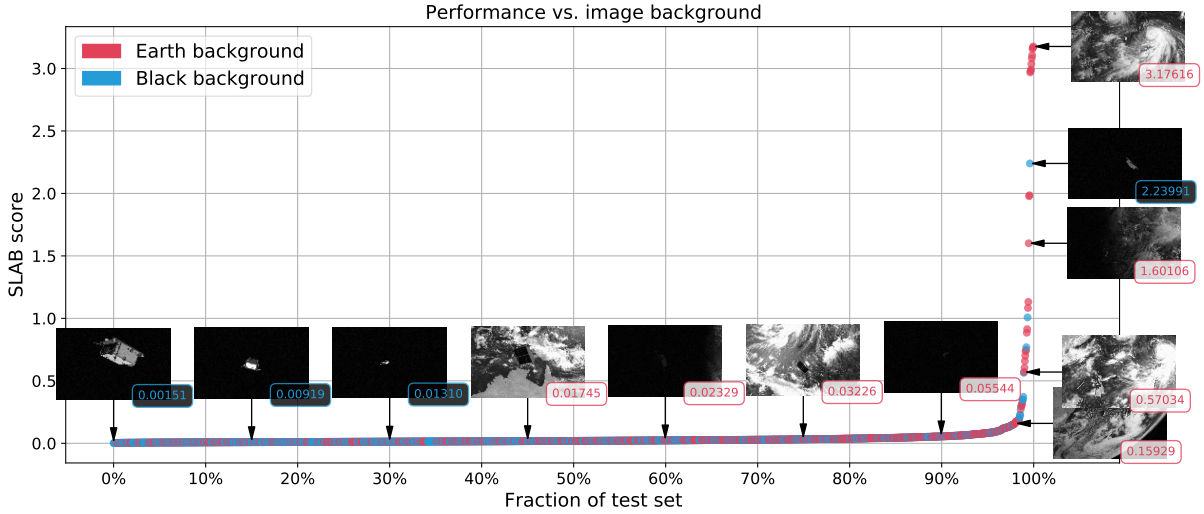


Figure 4.13: *Effect of the image background upon SLAB score*

4.5 Benefit from iterative pose refinement

Unless a pose outlier is detected, the initial pose estimate computed using the EPnP algorithm will always be iteratively refined by employing the Levenberg-Marquardt Method (LMM). It is intuitive that, despite resulting into an improvement of the final estimate, such a strategy will also entail an increased computational cost. It was eventually concluded that the negligible impact in terms of computational burden is certainly justified by the tangible increase in accuracy, compared to exclusively relying on EPnP.

For instance, on an Intel Core i7-4870HQ (2.5 GHz) CPU, the runtime associated with the EPnP algorithm only is about 10^{-6} s, while the LMM pose refinement is in the order of 10^{-4} s. In any case, the impact of the pose solver subsystem upon total runtime will remain negligible with respect to the two other subsystems of the pipeline (SLN and LRN).

In Figure 4.14 we compared the results obtained with and without LMM refinement, by plotting the 2σ distribution of the two normalized error components, across the whole test set. It is immediately visible that the distribution of errors obtained without pose refinement tends to be slightly shifted towards higher errors.

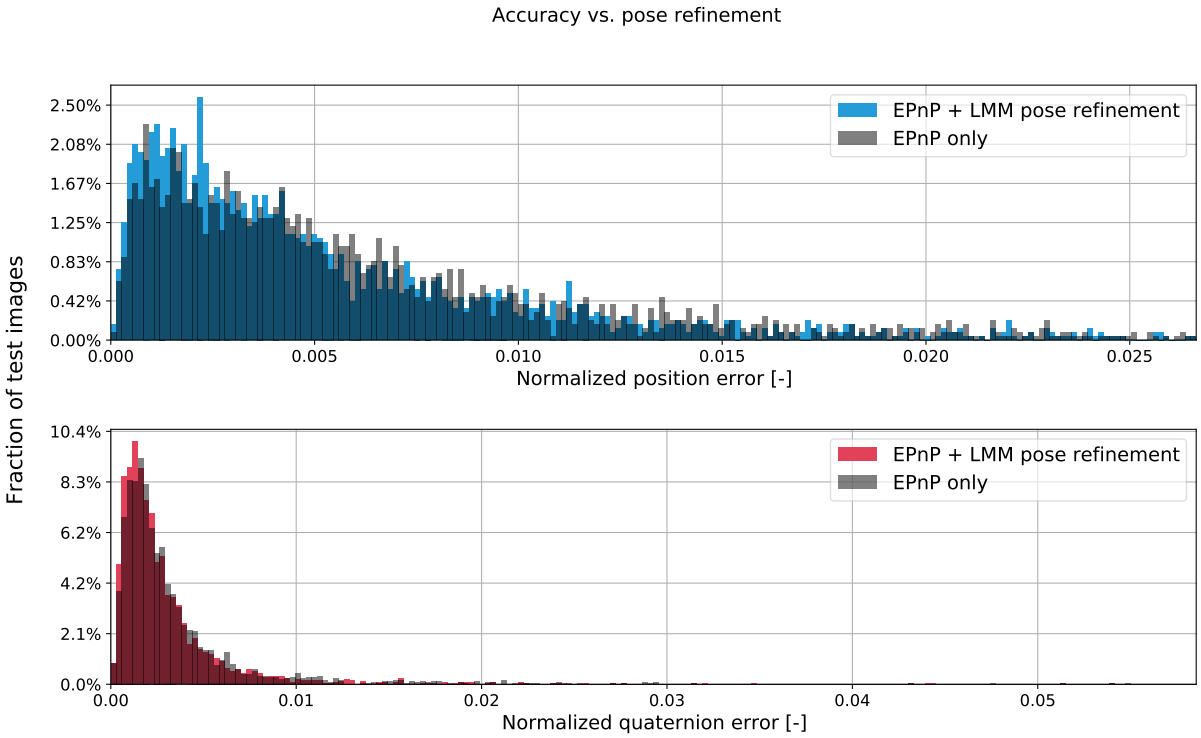


Figure 4.14: *Effect of pose refinement upon the normalized pose error*

We may at this point compare the resulting global performance metrics, which are reported in Table 4.2. Note that that pose refinement allows a 12.3% reduction of the Median Normalized Pose Error (MNPE).

Table 4.2: *Main performance metrics of the pipeline, with and without pose refinement*

	EPnP + LMM	EPnP only
Mean E_t	10.36 cm	11.14 cm
Median E_t	3.58 cm	4.31 cm
Mean E_q	2.24°	2.39°
Median E_q	0.81°	0.89°
MNPE	0.00648	0.00739
SLAB score	0.04627	0.04966

4.6 Runtime

The entire pipeline has been tested on an NVIDIA Tesla P4 GPU, in order to evaluate runtime across the whole test set. The resulting execution times for processing each individual image are reported in Figure 4.15, from which it is also possible to appreciate the order of magnitude of the computational cost associated with each of the three subsystems. The average total runtime is 0.089 s, which means that our pipeline runs at 11 FPS.

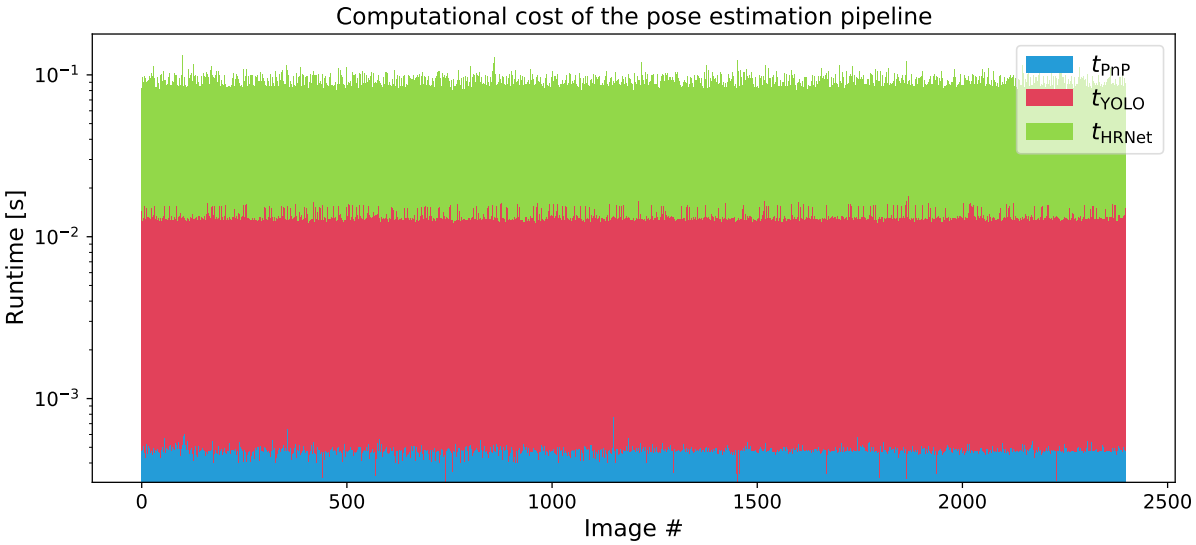


Figure 4.15: *Runtime breakdown, across the entire dataset*

It has to be highlighted that the performance here reported is not meant to be representative of actual spaceborne hardware/software integration. This is basically due to two main reasons:

- SLN and LRN were implemented using the `PyTorch` framework, which allows high-level programming for quick prototyping of an ANN architecture, but cannot be clearly compared, in terms of computational efficiency, with a C/C++ implementation
- a high-end off-the-shelf GPU has been used to evaluate runtime, which clearly outperforms any space-grade hardware

The two aforementioned aspects may somehow compensate in an actual spaceborne scenario, although further investigation is clearly required.

Nonetheless, with the obtained results, we may still compare in relative terms the computational cost of each subsystem. The most computationally expensive subsystem is LRN, which represents about 84% of the execution runtime. SLN and the pose solver will require instead an average time of 0.014 s and 0.0005 s, respectively.

4.7 Prediction visualization

For an immediate and straightforward visualization of the pose estimation results, an apposite graphical representation has been implemented. In Figures 4.16 and 4.17 the final estimated pose, along with the intermediate results from SLN and LRN, are represented. The two corresponding input images were randomly chosen from mid-range test images and are somehow representative of median performance, with Earth background (Figure 4.17) and without (Figure 4.16).

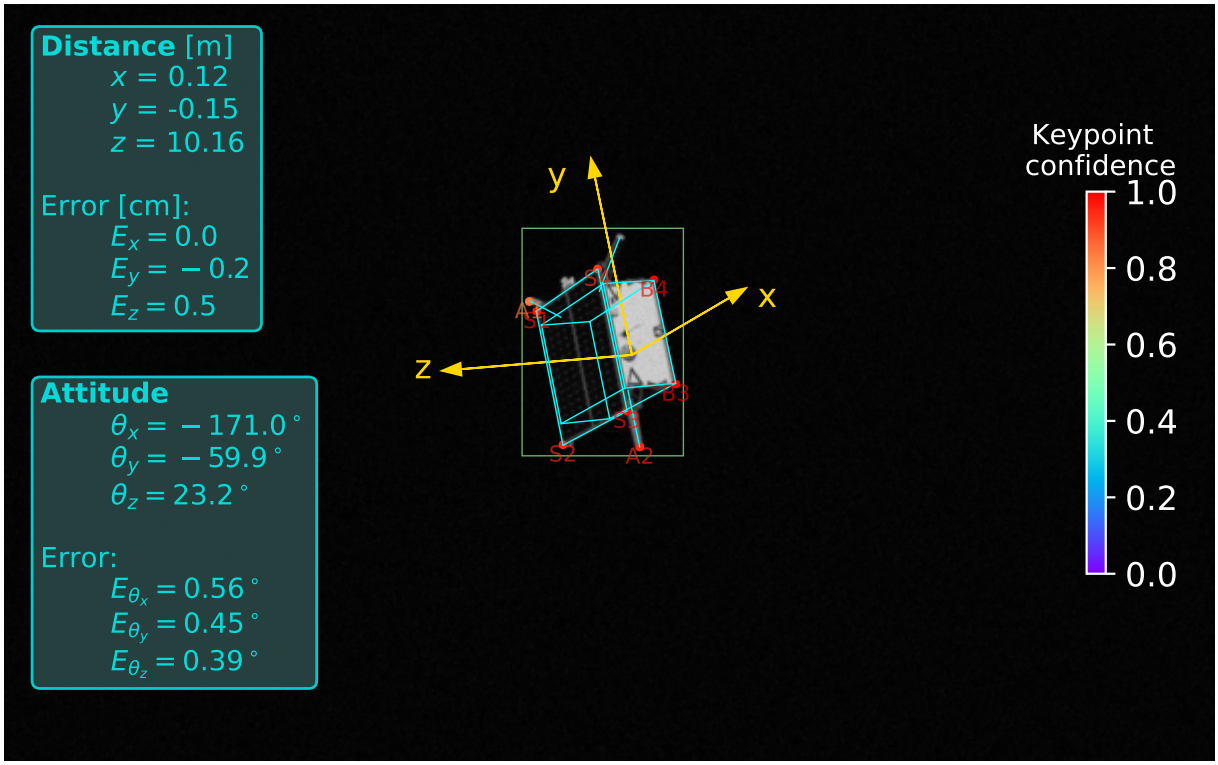


Figure 4.16: *Mid-range test image with black background*

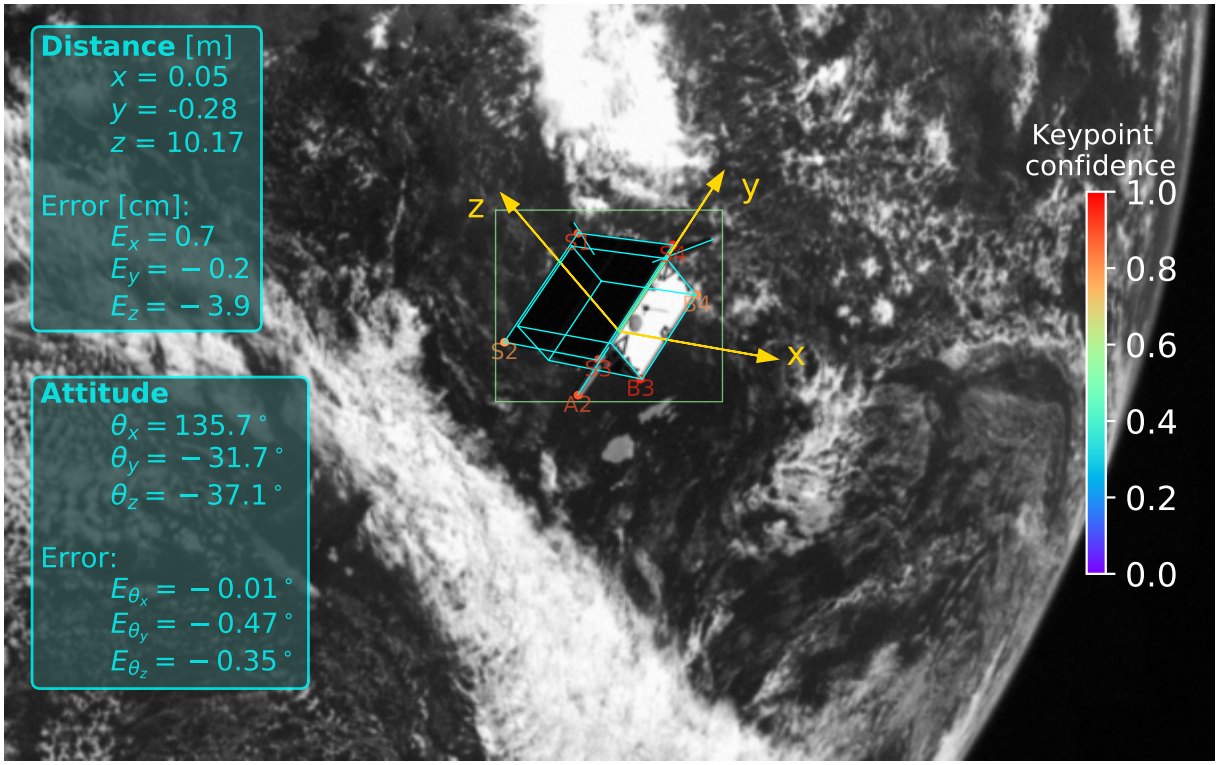


Figure 4.17: *Mid-range test image with Earth background*

In particular, in each image:

- a green rectangle delimits the RoI detected by the Spacecraft Localization Network
- the subset of estimated keypoint positions[(6)] that are fed to the pose solver is properly plotted and labeled; the color of each landmark indicates the confidence score predicted by the Landmark Regression Network, according to the color-bar provided on the right side of the image
- a cyan wireframe model is projected onto the image, based on the final pose estimate
- the body-fixed reference frame is plotted in yellow, according to the estimated attitude; the origin of the frame is in correspondence of the center of the bottom surface of Tango
- the two text boxes on the left side of the image report the predicted pose, along with the corresponding errors with respect to the Ground Truth

Let us now visualize some more examples, over a wide variety of conditions in terms of distance, illumination and background.

[(6)] the rejected low-confidence predictions are not included

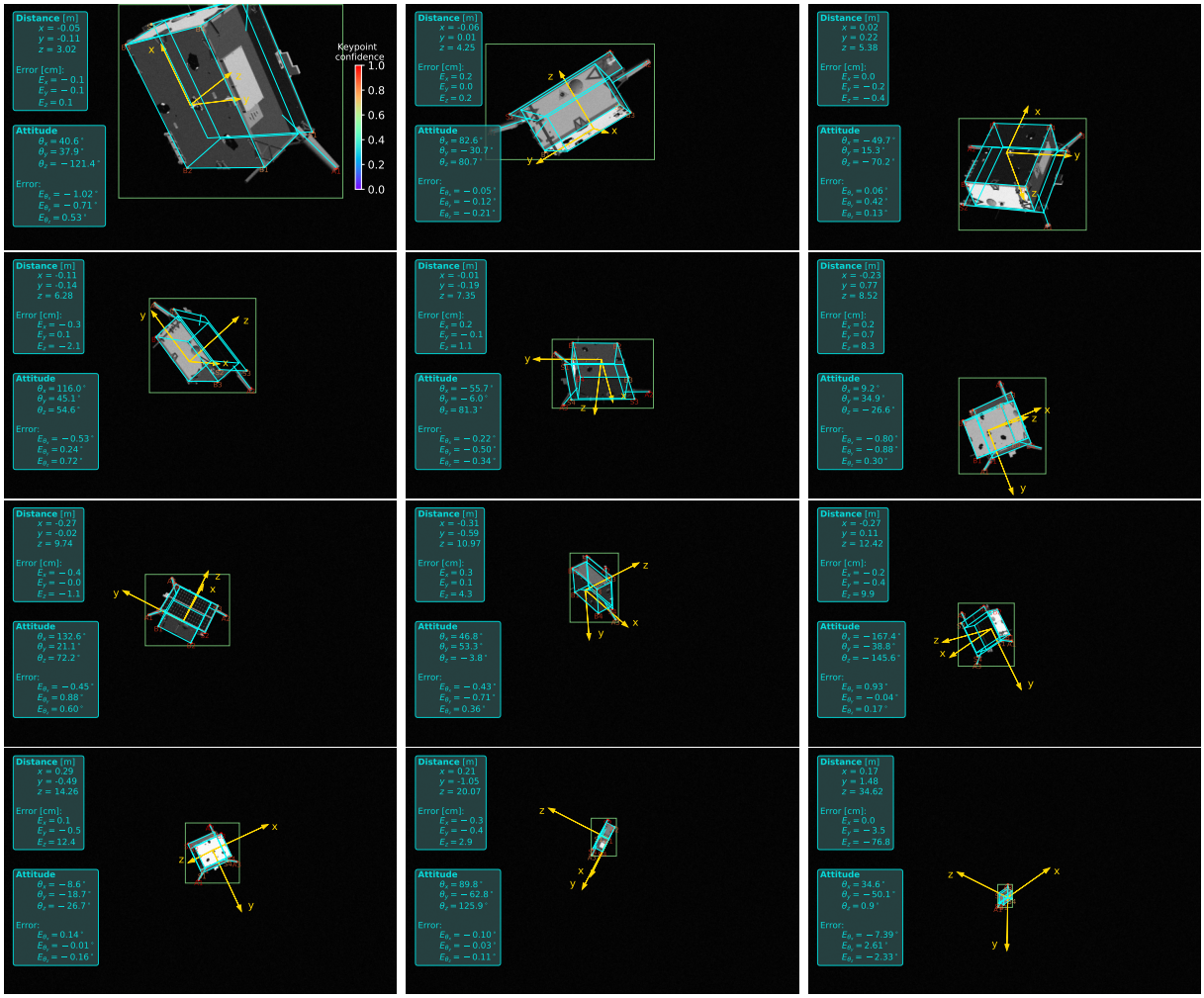


Figure 4.18: *Prediction visualization mosaic of test images with black background and increasing inter-spacecraft distance*

In Figures 4.18 and 4.19 a total of 24 random test images is displayed with the corresponding inference results. In particular, the random images were sampled from image batches, each with a given range of relative distances, and are here sorted in order of ascending distance. The 12 test images in Figure 4.18 have a black background, while the 12 remaining images in Figure 4.19 were all rendered with Earth in the background (some of them in eclipse condition).

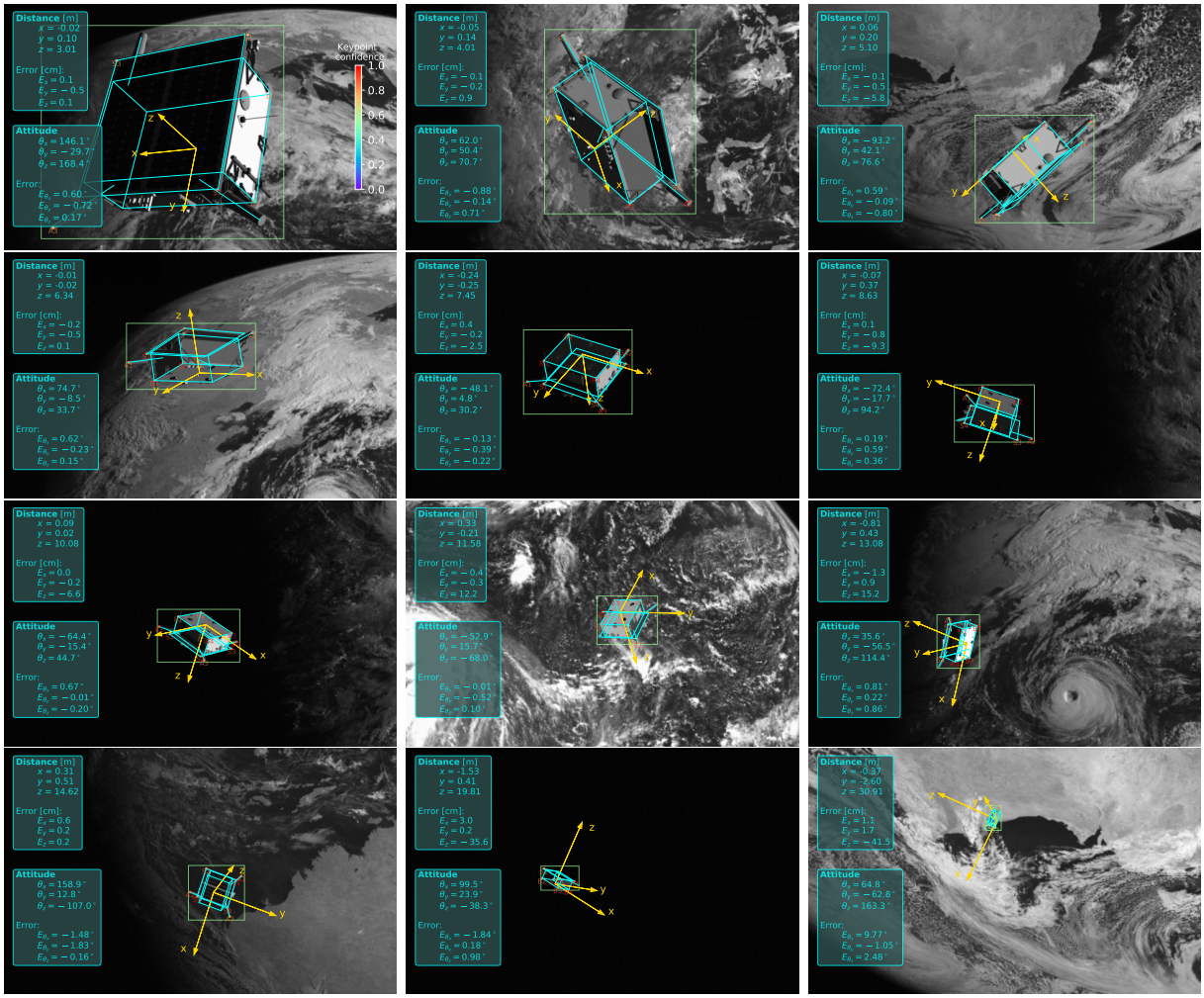


Figure 4.19: *Prediction visualization mosaic of test images with Earth background and increasing inter-spacecraft distance*

A total of 13 pose outliers out of 2400 test images has been detected and partially corrected (in terms of translation only). We also depicted 6 of the corresponding visualizations in Figure 4.20. All these images are characterized by a mid-to-large relative distance. As it can be immediately seen from the color of the dots, the keypoints are here predicted by LRN with very low confidence and, as a result, only the minimal set of 7 landmarks is retained.

The most common scenario in these cases is the one in which a given keypoint is mistaken for another one that is visually similar. This is particularly evident in the top right image of Figure 4.20 (`img002961.jpg`), in which most of the retained landmarks are detected with exceptional accuracy, except for two of them: what actually is the A1 antenna is mistaken for the A2 antenna and the detected S1 solar panel edge should have been the S3 landmark. This leads to an inconsistency which, nonetheless, the EPnP algorithm tries to fit, thus resulting into a completely wrong pose estimation.

The RoI-based approximation employed for correcting the relative translation vector yields substantial improvements, although the accuracy of the boresight component is highly dependent on the range between chaser and target.

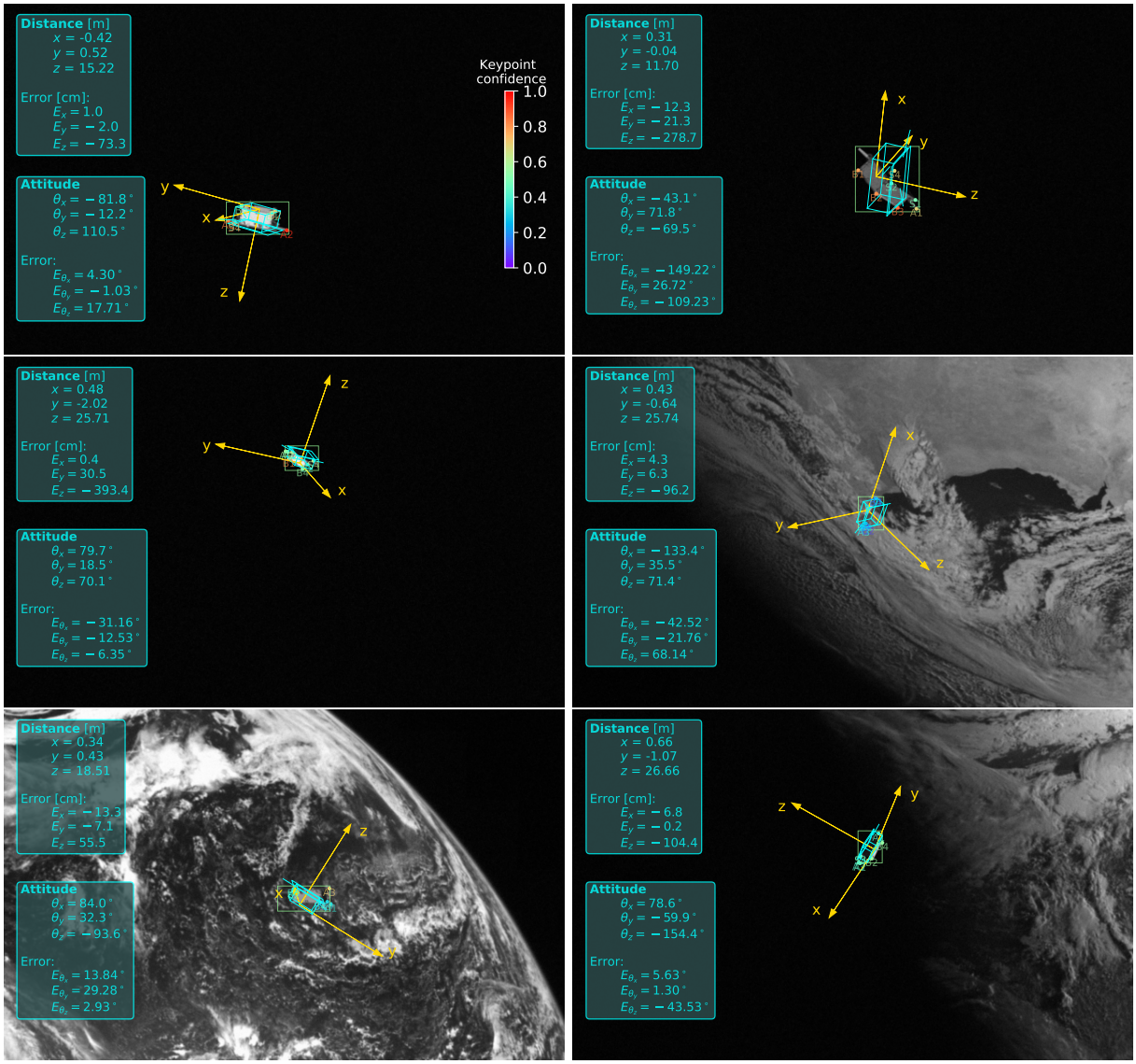


Figure 4.20: *Prediction visualization of 6 out of 13 pose outliers*

Chapter 5

Conclusions & future work

5.1 Conclusions

The main contribution of this work is the development of a deep learning-based pipeline capable of estimating the relative pose of an uncooperative spacecraft from a single monocular image, provided the knowledge of the target's 3D model and with no need of any other a-priori information.

Our discussion started with a survey of current state-of-the-art pose estimation techniques, either feature-based or deep-learning based, with a particular focus on the techniques employed by the top ranking teams that participated in the SLAB/ESA Pose Estimation Challenge. This was supplemented by an introduction to Convolutional Neural Networks (CNNs) and to algorithms for the solution of the Perspective-n-Point (PnP) problem.

At this point, we presented the architecture of our Relative Pose Estimation Pipeline (RPEP) which is composed of three main subsystems.

i) Spacecraft Localization Network (SLN). Its aim is to identify in the image the RoI, in which the S/C is located. This allows cropping out irrelevant portions of the image, so as to avoid unnecessary computation. SLN is a Convolutional Neural Network (CNN) based on the `YOLOv5` architecture. For this subsystem alone, the measured Average Precision is $AP_{50}^{95} = 98.51\%$, with a mean IoU of 95.38%.

ii) Landmark Regression Network (LRN). It processes the output of the previous subsystem in order to detect the position in the RoI of pre-defined semantic keypoints of the S/C. This CNN is based on the `HRNet32` architecture. The Average Precision of the landmark regression task, measured in terms of OKS thresholds, is $AP_{50}^{95} = 98.97\%$.

iii) Pose solver. This third and last subsystem receives as input the landmarks detected by LRN and seeks for the best pose fit of the known 3D wireframe model of the satellite, that minimizes the reprojection error. Our algorithm is based on the EPnP method for computing an initial pose estimate, which is iteratively refined using the Levenberg-Marquardt Method (LMM). The pose solver is also in charge of flagging and partially correcting possible pose outliers.

The performance of our pipeline has been tested on the synthetic images from the Spacecraft PosE Estimation Dataset (SPEED). The latter consists of 15300 images of the

Tango satellite and is the first and only publicly available ML dataset for spacecraft pose estimation.

Our architecture demonstrated to outperform the baseline developed by SLAB within the framework of the Pose Estimation Challenge. In particular, a SLAB synthetic score of 0.04673 has been achieved in the post-mortem competition, which means that our RPEP virtually ranks 3rd in original Pose Estimation Challenge. In addition, the same error metric evaluated on the real test set of SPEED corresponds to 0.12726, which, as of November 24th 2020, is the 2nd best score ever obtained since the beginning of the original competition in February 2019.

A global error metric alternative to the SLAB score has also been presented, which we called the Median Normalized Pose Error (MNPE). This score is characterized by two main peculiarities: it accounts for full normalization, both of position and attitude error; it evaluates median accuracy, which is more representative of actual performance in a nominal situation, i.e. in the absence of extreme conditions that could impair the quality of our estimation.

From the analysis of the results obtained on the test images in SPEED, it was concluded that the accuracy of our estimation strongly correlates with two main factors.

- Inter-spacecraft distance: there will clearly be a progressive drop in performance as the range between chaser and target increases.

- Presence of Earth in the image background: it is intuitive that images with a black background, due to the sharp contrast between the RoI and the rest of the image, will result into features that are easier to detect and hence higher accuracy of the estimated pose.

This means that pose estimation may be particularly challenging in the event of long-range images with cluttered backgrounds, which is indeed the case of our pose outliers. The end-to-end performance of our pipeline, evaluated across the entire test set, corresponds to an absolute translation error of 10.36 cm (mean) and 3.58 cm (median), while the quaternion error is 2.24° (mean) and 0.81° (median).

5.2 Future work

We will now provide a few directions for future work, that are necessary steps in the roadmap to spaceborne implementation of a fully vision-based relative navigation system. They are listed here below.

i) Performance evaluation in a dynamic rendezvous scenario. The output of the pipeline, which still processes individual frames, is fed to a navigation filter, which accumulates information from sequential images to provide a more accurate dynamic estimate of the pose. A detailed evaluation of the uncertainty in our raw estimates coming from the RPEP is clearly of paramount importance.

ii) Implementation of an algorithm for identifying individual keypoint outliers, hence removing them from the subset of landmarks processed by the pose solver. One of

the main drawbacks of our current architecture, is that whenever a pose outlier is detected, no action is taken in order to correct the attitude.[(1)]

iii) Validation of the architecture on actual spaceborne imagery.

iv) Implementation of data augmentation techniques such as the Neural Style Transfer, in order to randomize the S/C's texture in the images used for training our CNNs. This is of particular importance to address the issue of mismatch in terms of textures and reflective properties, between the synthetic imagery used during offline training and the actual flight imagery processed during online inference. Randomizing the textures of our training images would largely improve the robustness to such mismatches.

v) Low-level C/C++ inference implementation of our pipeline.

vi) Evaluation of the runtime on space-grade hardware or on an off-the-shelf microcomputer such as the `Raspberry Pi`.

vii) In order to support mission scenarios in which the target is unknown or partly know,[(2)] further generalization of the problem is clearly needed. This would expectedly come at the expense of the outstanding accuracy demonstrated in this dissertation.

[(1)] in a dynamic scenario, one may actually implement a navigation filter that, whenever a pose outlier is flagged, performs the propagation step but skips the update step, without necessarily having to identify the inconsistent keypoint detection(s)

[(2)] e.g. debris removal (completely unknown object) or de-orbiting of a dismissed satellite (might be damaged or exhibit surface degradation)

Bibliography

[Boc20] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection". In: *arXiv preprint arXiv:2004.10934* (2020).

[Bod12] Per Bodin et al. "The prisma formation flying demonstrator: Overview and conclusions from the nominal mission". In: *Advances in the Astronautical Sciences* 144.2012 (2012), pp. 441–460.

[Can86] John Canny. "A computational approach to edge detection". In: *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986), pp. 679–698.

[Cha95] Yves Chauvin and David E Rumelhart. *Backpropagation: theory, architectures, and applications*. Psychology press, 1995.

[Che19] Bo Chen et al. "Satellite pose estimation with deep landmark regression and nonlinear pose refinement". In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2019, pp. 0–0.

[Dai16] Jifeng Dai et al. "R-fcn: Object detection via region-based fully convolutional networks". In: *Advances in neural information processing systems*. 2016, pp. 379–387.

[DAm14] Simone D'Amico, Mathias Benn, and John L Jørgensen. "Pose estimation of an uncooperative spacecraft from actual space imagery". In: *International Journal of Space Science and Engineering 5* 2.2 (2014), pp. 171–189.

[Dho89] Michel Dhome et al. "Determination of the attitude of 3D objects from a single perspective view". In: *IEEE transactions on pattern analysis and machine intelligence* 11.12 (1989), pp. 1265–1278.

[Dud72] Richard O Duda and Peter E Hart. "Use of the Hough transformation to detect lines and curves in pictures". In: *Communications of the ACM* 15.1 (1972), pp. 11–15.

[ESA] ESA. *https://www.esa.int/Safety_Security/Clean_Space/ESA_commissions_world_s_first_space_debris_removal*.

[Gir14] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[Gir15] Ross Girshick. "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[Kel] Kelvins-ESA. *https://kelvins.esa.int/satellite-pose-estimation-challenge/*.

[Kin14] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[Kis20] Mate Kisantal et al. "Satellite Pose Estimation Challenge: Dataset, Competition Design and Results". In: *IEEE Transactions on Aerospace and Electronic Systems* (2020).

[Kri12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[LeC98] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[Lep09] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. "Epnp: An accurate o (n) solution to the pnp problem". In: *International journal of computer vision* 81.2 (2009), p. 155.

[Liu18] Shu Liu et al. "Path aggregation network for instance segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8759–8768.

[Low12] David Lowe. *Perceptual organization and visual recognition*. Vol. 5. Springer Science & Business Media, 2012.

[Low87] David G Lowe. "Three-dimensional object recognition from single two-dimensional images". In: *Artificial intelligence* 31.3 (1987), pp. 355–395.

[Mac67] James MacQueen et al. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA. 1967, pp. 281–297.

[Par19] Tae Ha Park, Sumant Sharma, and Simone D'Amico. "Towards Robust Learning-Based Pose Estimation of Noncooperative Spacecraft". In: *arXiv preprint arXiv:1909.00392* (2019).

[Pro20] Pedro F Proença and Yang Gao. "Deep learning for spacecraft pose estimation from photorealistic rendering". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 6007–6013.

[Qia99] Ning Qian. "On the momentum term in gradient descent learning algorithms". In: *Neural networks* 12.1 (1999), pp. 145–151.

[Red16] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[Red17] Joseph Redmon and Ali Farhadi. "YOLO9000: better, faster, stronger". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7263–7271.

[Red18] Joseph Redmon and Ali Farhadi. "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018).

[Red20] Nola Taylor Redd. "Bringing satellites back from the dead: Mission extension vehicles give defunct spacecraft a new lease on life-[News]". In: *IEEE Spectrum* 57.8 (2020), pp. 6–7.

[Ree16] Benjamin B Reed et al. “The restore-L servicing mission”. In: *AIAA space 2016*. 2016, p. 5478.

[Ren15] Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[San18] Mark Sandler et al. “Mobilenetv2: Inverted residuals and linear bottlenecks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520.

[Sha17] S Sharma, C Beierle, and S D’Amico. “Towards Pose Determination for Non-Cooperative Spacecraft Using Convolutional Neural Networks”. In: *Proceedings of the 1st IAA Conference on Space Situational Awareness (ICSSA)*. 2017, pp. 1–5.

[Sha18a] Sumant Sharma, Connor Beierle, and Simone D’Amico. “Pose estimation for non-cooperative spacecraft rendezvous using convolutional neural networks”. In: *2018 IEEE Aerospace Conference*. IEEE. 2018, pp. 1–12.

[Sha18b] Sumant Sharma, Jacopo Ventura, and Simone D’Amico. “Robust model-based monocular pose initialization for noncooperative spacecraft rendezvous”. In: *Journal of Spacecraft and Rockets* 55.6 (2018), pp. 1414–1429.

[Sha20] Sumant Sharma and Simone D’Amico. “Neural Network-Based Pose Estimation for Noncooperative Spacecraft Rendezvous”. In: *IEEE Transactions on Aerospace and Electronic Systems* (2020).

[Sob68] Irwin Sobel and Gary Feldman. “A 3x3 isotropic gradient operator for image processing”. In: *a talk at the Stanford Artificial Project in* (1968), pp. 271–272.

[Sun19a] Ke Sun et al. “Deep high-resolution representation learning for human pose estimation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2019, pp. 5693–5703.

[Sun19b] Ke Sun et al. “High-resolution representations for labeling pixels and regions”. In: *arXiv preprint arXiv:1904.04514* (2019).

[Tan20] Mingxing Tan, Ruoming Pang, and Quoc V Le. “Efficientdet: Scalable and efficient object detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 10781–10790.

[Tie12] Tijmen Tieleman and G Hinton. “Divide the gradient by a running average of its recent magnitude. COURSERA Neural Netw”. In: *Mach. Learn* 6 (2012), pp. 26–31.

[Uij13] Jasper RR Uijlings et al. “Selective search for object recognition”. In: *International journal of computer vision* 104.2 (2013), pp. 154–171.

[Ult] Ultralytics. *YOLOv5 (https://github.com/ultralytics/yolov5)*.

[Wan20] Chien-Yao Wang et al. “CSPNet: A new backbone that can enhance learning capability of cnn”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 390–391.