INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Université Fédérale Toulouse Midi-Pyrénées

Electrical Engineering & Computer Science Department

Research Initiation Project

Estimating the position of a non-cooperative spacecraft from a 3D model

May 31, 2024

Students

Armand	Carraz-Billat	carraz-b@insa-toulouse.fr
Hugo	Girard	hgirard@insa-toulouse.fr
Marti	Jimenez	m_jimene@insa-toulouse.fr
Daichi	Malbranche	malbranc@inca-toulouse.fr
Cristian	Martinez	martinez-col@insa-toulouse.fr
Titouan	Seraud	seraud@insa-toulouse.fr

Tutor

Damien Mariyanayagam

Abstract

This article tackles space rendezvous problems and more specifically satellite pose estimation. To estimate the position and orientation of a non-cooperative spacecraft, computer vision combined with neural network algorithms offer a good accuracy. However, the training process requires a lot of data while almost no space-born images are available. In order to fill this gap, we render synthetic images but it leads to a domain gap; neural networks trained on computer-generated images are less accurate on real images. The project goal is to create a full satellite pose estimation pipeline from image generation to pose estimation via image analysis. We created a dataset generator, selected the best neural network architecture to analyze images and implemented Perspective-n-Points algorithm to estimate the pose. In the end, all pipeline elements are ready to be used but not connected.

Keywords

Machine learning, Pose estimation, Image generation, 3D modeling, Satellite, Navigation

Acknowledgements

First and foremost, we would like to thank Mr. Damien Mariyanayagam and his SII team for guiding and helping us through this project. Their help has been invaluable to us and this opportunity has been very educational. We would also like to thank Mrs. Marie-Agnès Detourbe, our English teacher, for her help and support in writing this report.

Contents

Glossary

Blender Open-source modelization software.

BSDF Bidirectional Scattering Distribution Function.

CNN Convolutional Neural Network.

Frustum Truncated pyramid. The camera frustum represents its visible volume.

Gallium arsenide Gallium arsenide (GaAs) is a III-V direct band gap semiconductor with a zinc blende crystal structure.

MAE Mean Absolute Error. Average absolute difference between the estimated values and the actual values.

MLI Multi-Layer Insulation.

MSE Mean Squared Error. Average squared difference between the estimated values and the actual values.

PnP Perspective-n-Points.

Pose Position and orientation of an object.

Python High-level, general-purpose, interpreted programming language.

Quaternion A quaternion is an extended complex number with four components.

RANSAC Random Sample Consensus.

ResNet Residual Neural Network.

SPEED+ Next-generation of Spacecraft PosE Estimation Dataset.

Tango For space scientists, it is one of the two spacecrafts of PRSIMA mission.

List of Figures

List of Tables

1 Introduction

Space rendezvous are key points of space navigation. One object must estimate precisely the position and orientation - the pose - of another one. To meet space constraints, this is generally done using an on-board camera. It is then necessary to compute a pose from a given picture. The most efficient algorithms for this problem are neural-network based. However, these networks require an enormous amount of data to be trained and spaceborn images are very rare and cannot constitute a satisfactory dataset. To remedy this, we need to create synthetic images that are faithful to real models, so that we can train the network consistently and reliably.

We first established a literature review of all the research already conducted on the topic, and all the envisioned and implemented solutions. This research inspired our work, although it sometimes took place in contexts very different from ours. This literature review can be found in the report appendices.

Our work consisted of four phases, which we allocated to each member of our team. First, we modeled the Tango satellite in 3D using Blender software, as well as the space scene in which it would be set (stars, planets, different lights, etc.). Once the model was created, we wanted to automate the rendering of synthetic images to create the dataset via a Python script that we had to imagine. We also identified several key points on the satellite so that we could find them in the generated images. Next, we designed a neural network and trained it with our synthetic images so that it could recognize the Tango satellite. Finally, we used a PnP algorithm to deduce the satellite's position and rotation on the image.

This project approach was new to each member of the group, and was very enriching for us, but also confronted us with certain problems we were not used to dealing with. In this report, we will present our methods and results step by step, while discussing the difficulties we encountered and how we adapted.

2 Project organisation

This project was carried out as part of our studies at INSA Toulouse, like all the other students in the first year of their Masters. However, the specificity of our project was the presence of a partner company, SII, which provided us with technical and human support.

2.1 Working with a company: SII

2.1.1 Company introduction

SII, or "Société pour l'Informatique Industrielle" (Society for Industrial Computing), is a French company founded in 1979 by Bernard Huvé. It specializes in engineering and technology consulting as well as digital services. It operates in various sectors, including aeronautics, defense, telecommunications, and other key industries. In 2023, they launched the ASPERA (Autonomous Spacecraft Pose Estimation with Realistic Acquisitions) challenge with the aim of obtaining solutions to this problem. Our objective was therefore to contribute to the research linked to this project.

2.1.2 Advantages of working with a company

This project was very educational for us because it was our first real experience of collaboration with professionals in the industry. Indeed, having teachers as project managers is also educational, but we were able to immerse ourselves in business life and get an idea of how to lead a project, which will be useful to us later. Also, our tutor allowed us to visit the company's premises to get a more precise idea of their scope and their current projects.

Damien gave us a lot of freedom, while being very accessible. It was very pleasant, he had an eye on what we were doing while not putting pressure on us to produce results, which left our minds free to study the other courses. He understood our situation and we are very grateful that he accommodated our schedule to accommodate meetings. We developed a relationship of trust, where everyone knew what they had to do by when, and who to contact if they were stuck.

2.2 Teamwork for a project

2.2.1 Dividing up the work

As explained in the introduction, our project could easily be divided into large, identifiable and almost independent parts. We therefore decided to assign a member of the group to each part according to the skills and desires of each person. The allocation of the tasks was done simply and everyone was able to move forward on their part of the puzzle on their own. Obviously the other members of the group were available for any questions or requests.

2.2.2 Organization of work and definition of deadlines

For the overall organization of the project, we used a project management software: Notion. This software is perfect for identifying tasks to do, sorting them, organizing them and setting deadlines. With this, each of us could have access to each other's progress, what remained to be done, and have an overall overview of the remaining tasks.

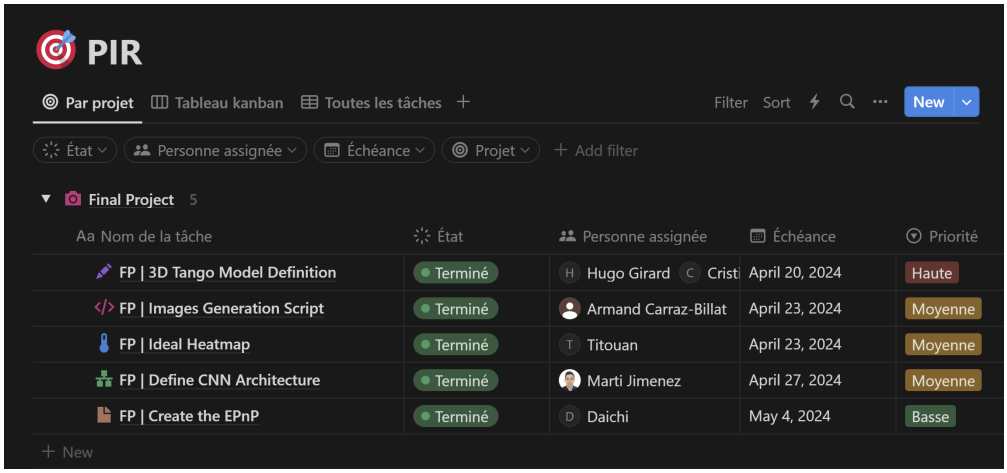

Figure 1: Excerpt from our Notion space

2.2.3 Issues encountered

At this stage of our studies, we are now used to working in teams. It was still difficult to adapt to a new team with new colleagues. We had to face a period of adaptation at the start of the project to understand how everyone worked, and to be able to communicate effectively.

Also, you must be able to adapt to unexpected events. In this case, one of the members of our group had major personal problems, which pushed us to rethink the schedule, reorganize tasks and reorganize ourselves. In the end, everything went well but we have to keep in mind that every project does not necessarily always go as planned and we know how to adapt if necessary.

3 Dataset generation

3.1 3D mesh

In order to generate artificial images of a satellite, we first needed a 3D model of the given satellite. We quickly chose to work on the same spacecraft as the 2021 Satellite Pose Estimation Competition [4], Tango.

Hence, we have to recreate a 3D model from SPEED+ dataset images [5]. Two main approaches were possible:

- Manual approach: handmade modeling using images as references.
- Neural network approach: use a multi-view stereo neural network to compute a 3D representation using images as input.

We picked the first approach because 3D modeling is easier to learn than neural network but above all because it produces a cleaner mesh which is necessary to generate realistic images. It is more important to have a mesh without artifact than to have the exact same sizes.

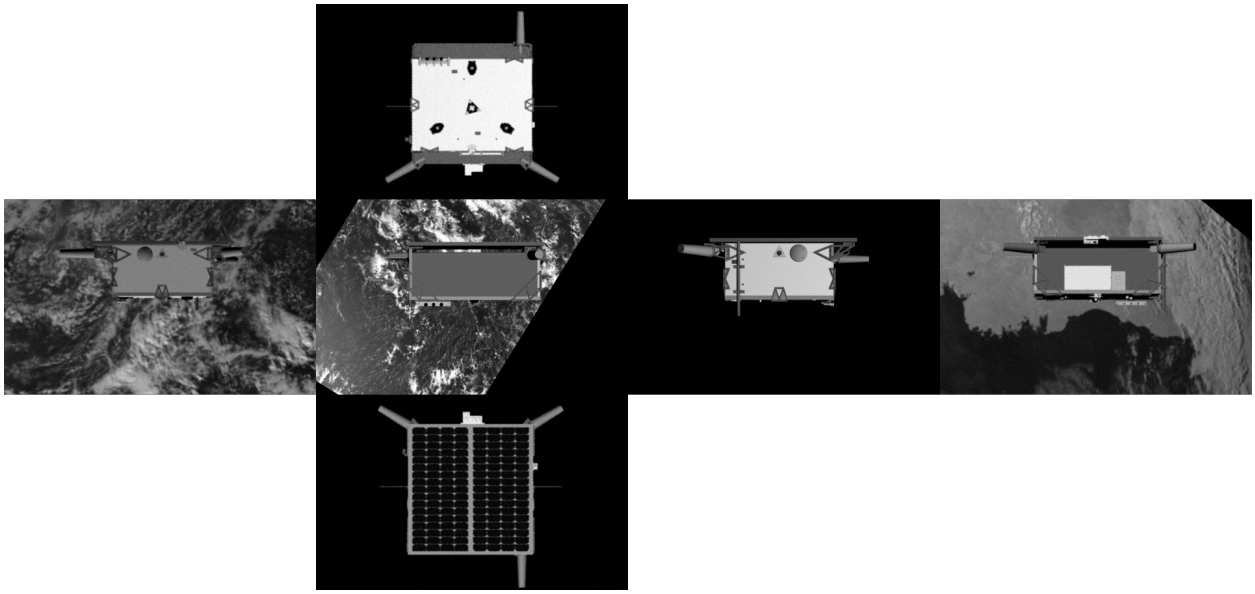

Figure 2: Image references extracted from SPEED+ dataset.

Before starting modeling, the reference images had to be chosen carefully. To simplify later work, we wanted side-view images from all six directions. However with several tens of thousands of images, identifying the best ones by hand was impossible. We then wrote a small script searching for the most axis aligned images. Since 2D image can by easily rotated, the script forgets about this specific rotation during the selection phase. Later on, during the correction phase, selected images are rotated back to the horizontal (cf. Figure 2).

Remark. Technical drawing side-views are created using orthographic projection while image from SPEED+ are in perspective. It leads to size differences which complicates the modeling process.

Once the image references were selected, we imported them into Blender and placed them. The modeling part could then begin. The satellite is made of several pieces which must be identified. We modeled them from the biggest one - the frame - to the smallest ones - small details. At the end, the 3D model is made of 27 pieces and about 10k triangles (cf. Table 1).

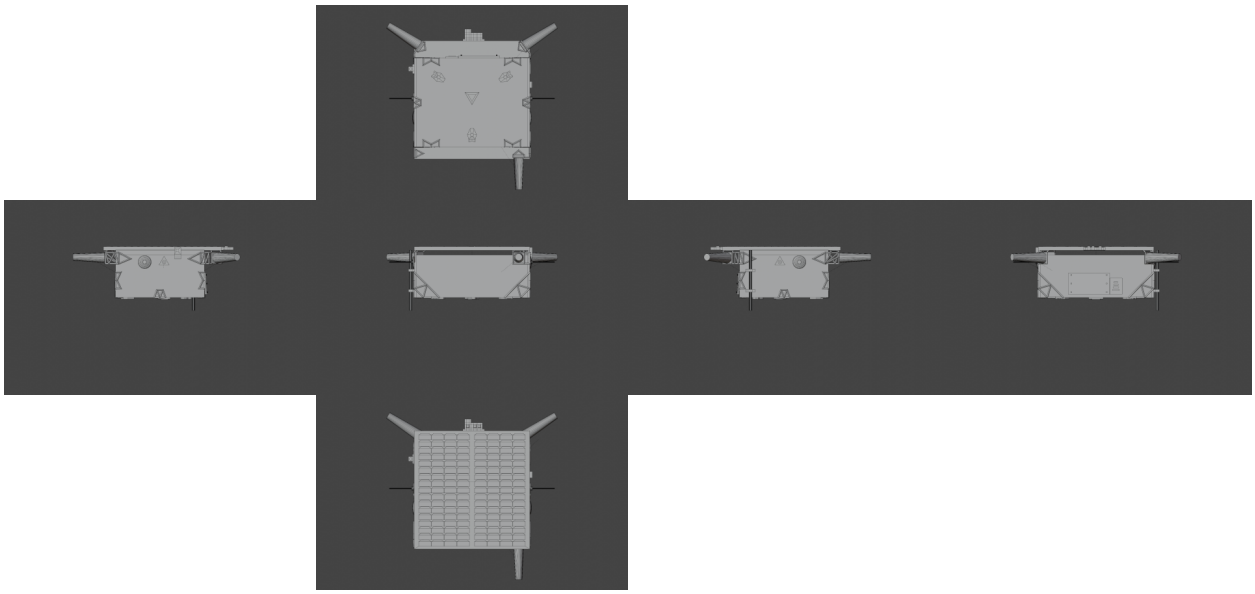

Figure 3: Side views of Tango 3D reconstructed mesh.

Objects	27
Vertices	5,587
Edges	9,765
Faces	4,395
Triangles	10,744

Table 1: Tango mesh statistics.

3.2 Textures

After constructing the mesh, some textures are added to the model. The objective is to create Blender assets that emulate the materials that make up the Tango satellite. We then apply these texturing assets to the model. Overall, this process results in a more realistic model than the bland mesh, which resembles modelling clay. This is achieved by giving each part of the model its own identity by creating a texture for each of the satellite's materials.

3.2.1 Real materials analysis

The main issue with texturing in our case is that in reality, space imagery is monochrome. We still assign realistic colors to each material, but this only results in different shades of gray. Hence, we have to play with other parameters to create the difference between materials, such as light reflection, metallic effects, or the patterns on their paint.

We see five different types of material on the real satellite, so we create the same number of textures. Here is a list of these materials:

- **MLI (Multi-Layer Insulation)** is a thermal insulation consisting of several layers of thin sheets and is often used on spacecraft. It is usually applied to the body of the spacecraft, primarily to reduce heat loss through thermal radiation. It gives some spacecraft the appearance of being covered with gold foil.

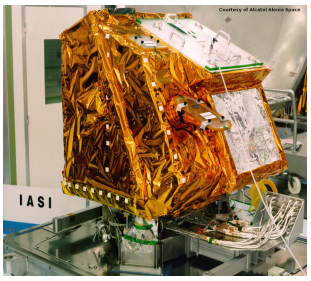

Figure 4: Part of the MetOp-C satellite covered with MLI from Thales Alenia Space [1]

- **A solar cell** is an electronic device that uses the photovoltaic effect to convert light into electricity. For space applications, it is usually made of gallium arsenide. These cells are assembled into solar panels mounted on the spacecraft. This charges its batteries using sunlight. These batteries can power the spacecraft even when it moves out of direct sunlight. Solar cells are usually of a night-blue color, and highly reflective.

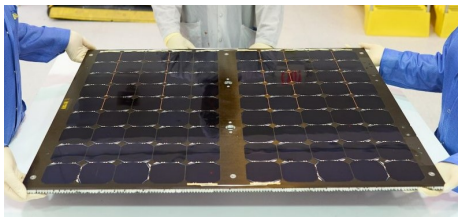

Figure 5: Space solar panel [2]

- **Unknown light colored metals** have been used on some parts of the satellite, such as solar cells junction, the sun acquisition sensor, or the sun presence sensor. We represent them with the same texture.
- **Unknown metals** have been used on some parts of the satellite, such as metrology antennas, torque rod, or GPS antennas.
- **Unknown dark colored metals** have been used on some of the joints in the reinforcement.

3.2.2 Blender textures creation

Having defined our texture needs, the creation begins. We use Blender texture nodes to generate procedural textures. These nodes simplify the process by providing easy access to all the paramaters needed to build our textures. Another option would have been to create textures from real images, but this would have taken longer to achieve the same result, as we did not have access to solar cells nor MLI blanket.

There are many texture nodes, with many parameters in Blender, but in our case, we mostly only need one: Principled BSDF (Bidirectional Scattering Distribution Function). It combines multiple layers into a single easy to use node. It can model a wide range of materials. Even in this node, we only use three parameters: base color, metallic, and roughness. Base color, as the name suggests, is the overall color of the material. Metallic gives a metallic aspect to the texture. As all of our materials are made of metal, we maximise this parameter. Roughness is the specular reflection and transmission of the material.

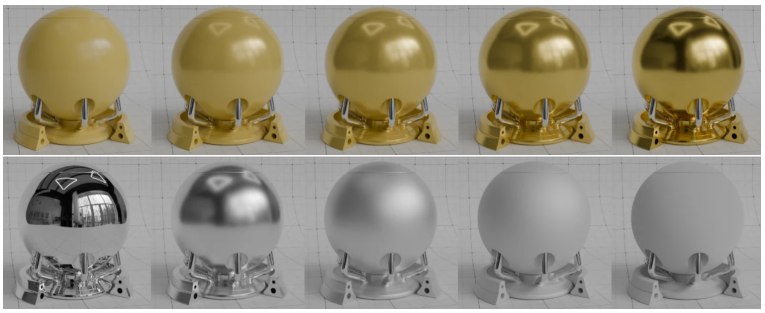

Figure 6: Metallic (on top) and roughness examples ranging from 0.0 to 1.0 [3]

Here is a list of our textures parameters and some renders:

- **Unknown metals** have medium roughness (0.5) because they seem not as reflective as the others. We use the same parameters for each type of unknown metal, except for the base color. The three base colors for these parts are light gray, gray and dark gray.
- **Solar cell** has low roughness (0.2) because it is very reflective. The base color is night-blue.
- **MLI** also has low roughness (0.2) because it is very reflective. We use a noise node, connected to a color ramp node (black to yellow) as the base color, to emulate the folds that appear on the MLI layer due to its thinness. We apply this texture directly to the body of the satellite. It would be better to create a cloth object, cover the satellite body with it and texture it properly. Although our solution is not the most qualitative, it is clearly the most optimal, as the others require much more time and knowledge.

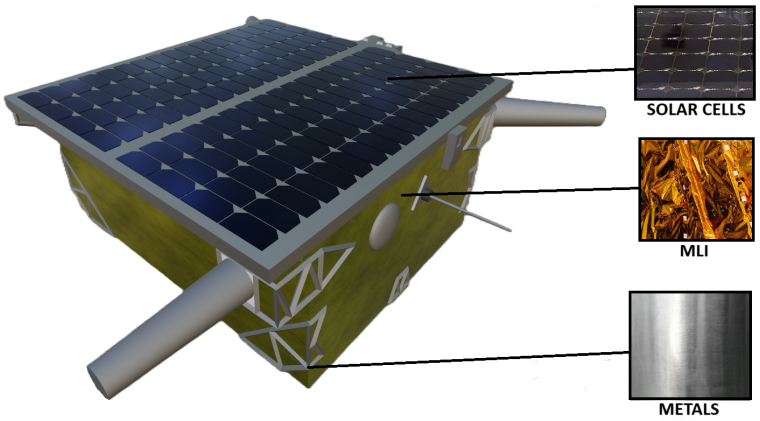

Figure 7: Final 3D model, with all textures applied

3.2.3 Skybox textures

In addition to texturing the satellite itself, creating an immersive environment around it enhances the realism of the scene. The skybox, a cubemap texture applied to a large cube surrounding the scene, plays a crucial role in simulating the vastness of space. Our skybox consists of textures representing the stars, Earth, and the Moon.

- **Stars Texture** To depict the countless stars in the cosmos, we utilized a high-resolution image of the night sky. This texture serves as the backdrop, enveloping the entire scene and expanse of space. By carefully selecting and mapping this texture onto the inner faces of the skybox cube, we create a convincing illusion of depth and infinity.

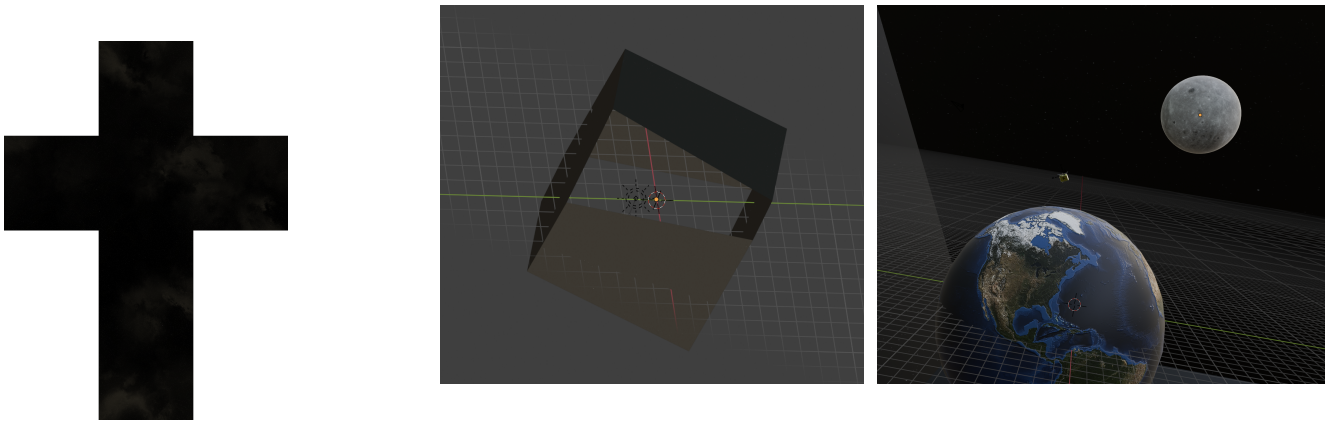

Figure 8: Stars texture applied to the skybox

- **Earth Texture** The Earth texture provides a realistic representation of our planet, visible from the perspective of the satellite in orbit. To achieve this, we utilized a high-fidelity texture map of Earth's surface, incorporating features such as landmasses, and oceans. By applying bump mapping techniques, we added depth and texture to the Earth's surface, enhancing its visual fidelity.

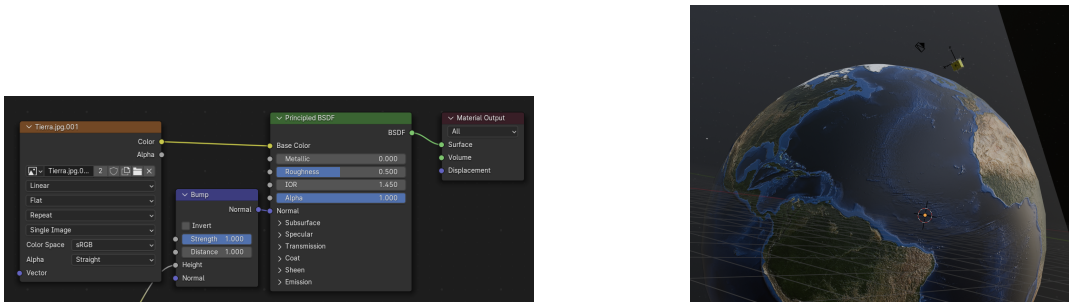

Figure 9: Earth texture with bump mapping applied

- **Moon Texture** Similarly, the Moon texture provides a detailed portrayal of Earth's celestial companion. Leveraging data from NASA, we acquired a meticulously crafted texture map, capturing the lunar surface's distinct features, craters, and more.

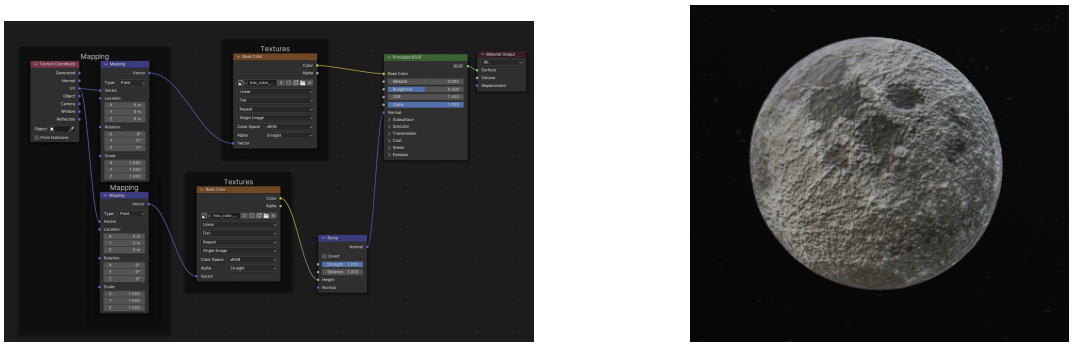

Figure 10: Moon texture positioned within the skybox

By integrating these textures into the skybox, we create a captivating backdrop that complements the satellite model, immersing viewers in the vastness of space and enhancing the overall realism of the scene.

3.3 Key points definition

In order to define the satellite pose, key points are generated. The 3D points coordinates are set relative to the center of the lower face of the satellite body. The aim is to select a few points that represent the overall shape of the satellite. Therefore, we select 11 key points on the outer vertices of the body, and on the 3 antennas.

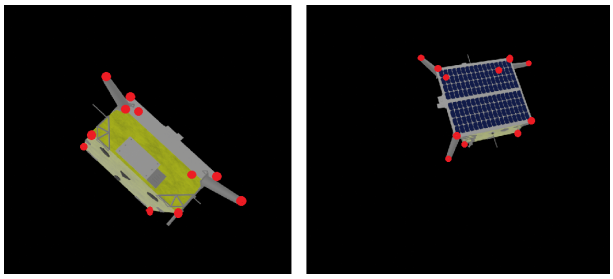

Figure 11: Keypoints (in red) placed on two generated images.

3.4 Rendering process

To generate a full dataset containing thousands of images, an automatic process must be defined. We use the open source rendering software Blender which includes a Python scripting interface. We create a script which images with its associated pose label.

3.4.1 Camera frustum

Any camera has two very important parameters: the focal and the captor size. With these two, the field of view is well defined.

The visible area of a camera is a conic region delimited by four planes: left, right, bottom and top. Any point inside this cone is visible. For image rendering, it is impossible to handle infinite distance. Hence, two bounding planes are added: near and far. Anything further than far plane or closer than near plane is cut off during the rendering process. Thus the camera frustum is a cuboid region defined by six planes.

3.4.2 Possible poses for the satellite to be visible

We first need to define what kind of random pose we want. For the rotation part, it is simple, we uniformly choose a random rotation. The position part is much more complicated. We want the satellite to be inside the camera frustum. One could simply pick a random point inside the frustum and place the satellite on it. However, if the satellite origin is too close to the frustum border, the satellite might be partially outside the image. It is a problem, we want the satellite to be entirely visible.

In order to ensure the satellite is entirely inside the camera frustum, we must reduce the available satellite positions. The possible position volume is defined by shrinking the camera frustum according to the satellite size and rotation. Thus, the rotation is chosen before the position.

Several approaches are possible for the computation of the volume of possible positions

- Exact approach. The camera frustum is reduced so when the satellite is on an extreme position, it precisely intersects the camera frustum. It requires to loop over all the satellite vertices for each image rendering.

- Bounding sphere. The satellite is approximated with a bounding sphere. The camera frustum is uniformly shrunk according to the sphere radius, which can be precomputed. For a good approximation, the satellite origin must be close to the geometry center. The more oblong the satellite is, the worse the approximation is.

- Bounding box: The satellite is approximated with an oriented bounding box. The bounding box is precomputed once.

To efficiently render images, the mesh complexity is abstracted by a bounding volume. By doing so, we are no longer dependent on thousands of vertices but only on the few parameters defining a simple shape. To avoid the bad outcomes of a bounding sphere - origin and shape dependency - the bounding box is preferred. The bounding box is precomputed once and no update is needed during the rendering process. Each time the rotation changes, a second bounding box is computed. This second bounding box is axis-aligned and is based on the first bounding box. The remaining problem is now much easier, we just have to place an axis-aligned box so it is totally inside the camera frustum (see Figure 12).

Remark. We can avoid to create a second bounding box using exact approach on the oriented bounding box. By doing so, the computations are not too heavy since the bounding box is only defined by 8 vertices.

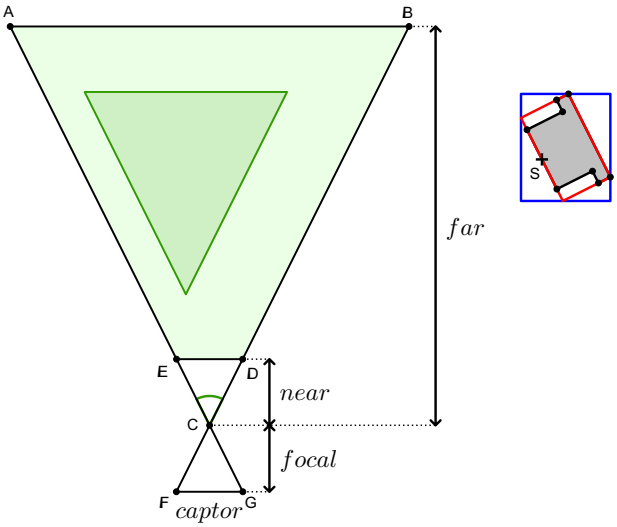

Figure 12: Side view of possible position for the satellite origin S. The camera is placed on C and its frustum is the trapezoid ABDE (in green). The satellite mesh is defined by the black vertices. The oriented bounding box is in red. The axis-aligned bounding box is in blue. This second bounding box is entirely contained in the camera field of view if and only if S is inside the dark green region.

3.4.3 Random point in a frustum

To simplify the explanations, we consider the 2D version of the problem. Adding the third dimension does not lead to any problem, top-bottom axis works the same as left-right axis. In the end, the possible positions for the satellite origin are either defined by a triangle or by a trapezoid. Since a triangle is just a trapezoid with two points coinciding, we only consider the trapezoid case. Thus, we want to generate a random point inside a trapezoid. Because a trapezoid is a quadrilateral like the square, we start by generating a point inside a square. It is as easy as generating two random numbers in $[0, 1]$. Then, using the right linear interpolations (cf. Equation 1) we get a point inside our trapezoid (cf. Figure 13).

$$(x, y) = f(\alpha, \beta) = \text{lerp}(\text{lerp}(E, F, \alpha), \text{lerp}(H, G, \alpha), \beta) \tag{1}$$

Where lerp function is defined by:

$$\text{lerp}(A, B, t) = (1 - t)A + tB \tag{2}$$

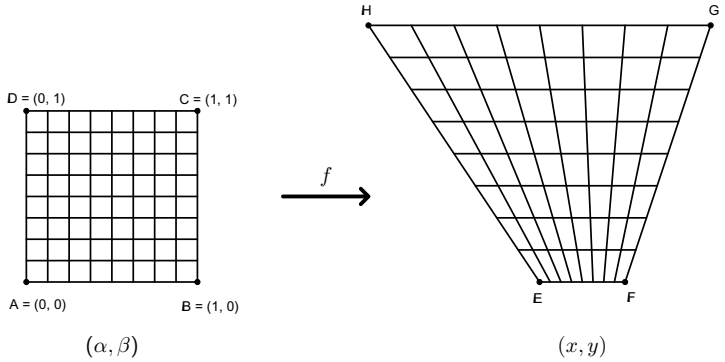

Figure 13: Linear interpolation from $[0, 1]^2$ square ABCD to trapezoid EFGH.

Remark. The random position generator defined by equation 1 does not produce uniformly spaced points. However, it produces uniformly distributed points along the camera axis, and for a given distance, along left-right axis. This behaviour is interesting for our application in the sense that there is as much probability for a point to be close to the camera as to be far away.

3.4.4 Bringing light to the scene

To render images in a realistic way, we need lights. We only consider one light - the Sun - which is modelled by a directional light. For each image, a random rotation is generated. This rotation defines the direction from where the light rays are coming from. A constant ambient light also illuminates the scene to avoid pure black color for the parts of the scene not directly illuminated by the Sun.

3.5 Identifying the position of keypoints

Once the dataset has been generated and all the translations and rotations obtained, it is necessary to use them to deduce the new positions of the initial keypoints. To do this, we use a Python script, which by matrix multiplication, gives us the 2D projected position of the keypoints after translation and rotation. The calculation of the new coordinates of the keypoints is used to train the neural network.

3.5.1 Translation and rotation

The first step is to apply translation and rotation to the keypoints.

- **Translation :** Applying a translation is easy, since it is simply the addition of two vectors of the same size.

$$\begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix}$$

- **Rotation :** Rotation management is trickier. The output is a quaternion representing the rotation. We must then transform this quaternion $q = (q_0, q_1, q_2, q_3)$ into a matrix:

$$R = \begin{bmatrix} 1 - 2q_2^2 - 2q_3^2 & 2(q_1q_2 - q_3q_0) & 2(q_1q_3 + q_2q_0) \\ 2(q_1q_2 + q_3q_0) & 1 - 2q_1^2 - 2q_3^2 & 2(q_2q_3 - q_1q_0) \\ 2(q_1q_3 - q_2q_0) & 2(q_2q_3 + q_1q_0) & 1 - 2q_1^2 - 2q_2^2 \end{bmatrix}$$

Finally, we just need to apply this matrix to the points.

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = R \cdot \begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix}$$

At the end of these transformations, we have the new 3D coordinates of each keypoint. Now we need to project these points into 2D.

3.5.2 2D Projection

To project points in perspective, we will use two modifications :

- **A first modification with an orthographic projection matrix** Here is the shape of this matrix:

$$M_{orth} = \begin{bmatrix} \frac{2}{\text{right}-\text{left}} & 0 & 0 & -\frac{\text{right}+\text{left}}{\text{right}-\text{left}} \\ 0 & \frac{2}{\text{top}-\text{bottom}} & 0 & -\frac{\text{top}+\text{bottom}}{\text{top}-\text{bottom}} \\ 0 & 0 & -\frac{2}{\text{far}-\text{near}} & -\frac{\text{far}+\text{near}}{\text{far}-\text{near}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  The values Right Left Top Bottom are the limits of our images and their size, i.e. 512x512 pixels. Near and Far are the parameters of the nearest and furthest cutting planes.

- **Then we apply a perspective projection matrix :**

$$M_{p2o} = \begin{bmatrix} \frac{f}{\text{aspect}} & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & \frac{\text{far}+\text{near}}{\text{near}-\text{far}} & \frac{2 \cdot \text{far} \cdot \text{near}}{\text{near}-\text{far}} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

  f is the camera focal length and aspect is the aspect ratio of the 512x512 image.

  It should also be noted that this matrix is designed for a camera oriented towards the negative Z axis, as this is the default setting in Blender. In our case, the camera is oriented towards the negative Y. The matrix will therefore need to be modified accordingly.

  Following these two transformations, we will have moved from 3D coordinates to 2D coordinates, which will enable us to identify keypoints for the PnP algo.

3.6 Difficulties encountered

One significant challenge we faced was our lack of prior experience with Blender, the software chosen for modeling and texturing. Learning to navigate Blender and creating realistic textures posed initial difficulties. Modeling the satellite accurately and finding appropriate textures required experimentation and problem-solving.

However, these challenges provided valuable learning opportunities. Through perseverance and collaboration, we overcame the obstacles and gained proficiency in Blender. Despite the initial hurdles, the project ultimately enhanced our skills in 3D modeling and texture creation.

4 Pose estimation

In recent years, Convolutional Neural Network (CNN)-based approaches have emerged as a powerful alternative for pose estimation tasks. These methods leverage the capability of deep learning models to learn complex patterns and relationships directly from raw image data. By training on large datasets of annotated images, CNNs can effectively infer the pose of objects with high accuracy and efficiency.

4.1 Motivation of our model approach

Existing methods for pose estimation, particularly those based on Convolutional Neural Networks (CNNs), encounter several challenges that hinder their effectiveness in real-world applications. One significant challenge is the time-consuming process of predicting heatmaps, which are often used to locate keypoints in the image. The generation of these heatmaps typically involves multiple convolutional layers and can be computationally intensive, leading to delays in inference time, especially for high-resolution images or real-time applications.

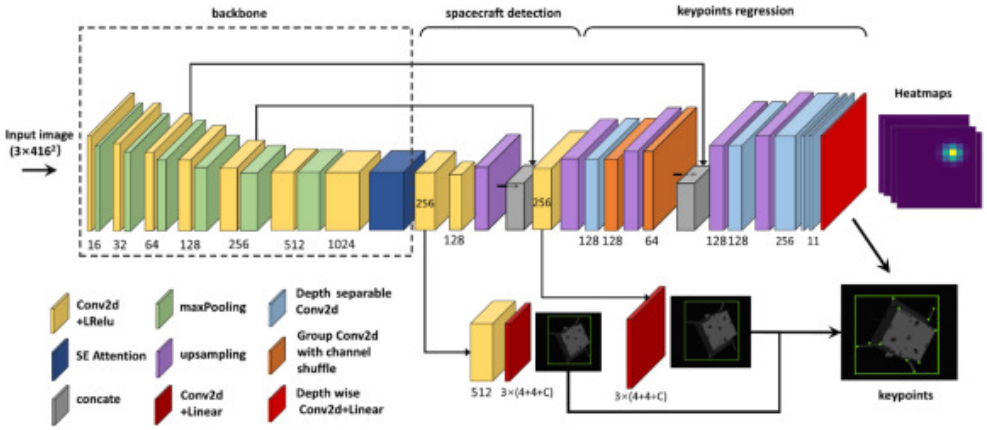

Figure 14: Example of heatmap prediction model

Moreover, while directly predicting the pose from image data can be advantageous in terms of simplicity and end-to-end learning, it often comes with the drawback of opacity. That is, without the visualization of intermediate representations such as heatmaps, it becomes challenging to understand where the model has identified keypoints and where it might have gone wrong. This lack of interpretability can hinder the debugging and refinement of the model, making it challenging to diagnose and correct errors effectively.

4.2 Our modelling approach for the CNN

In our approach, we introduce a novel method of directly predicting the coordinates of keypoints along with binary indicators of whether they are inside the frame. Unlike traditional methods that rely on predicting heatmaps or intermediate representations, our model directly outputs the spatial coordinates of keypoints, providing a more interpretable and transparent solution. By incorporating information about whether keypoints are in the frame, we enhance the model's ability to understand the context of the detected features.

We adopt a bottleneck ResNet-50 architecture as the backbone of our model. This choice offers several advantages for our pose estimation task. Firstly, the bottleneck architecture effectively reduces the computational burden by employing smaller convolutional kernels within each residual block, resulting in a more lightweight and efficient network. Additionally, ResNet-50's deep architecture allows for the extraction of hierarchical features, enabling the model to capture intricate patterns and spatial relationships in the input images.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 15: Architectures for bottleneck ResNets (Our architecture is written the 50 layer columns)

4.3 Bottleneck ResNet-50 architecture

The bottleneck ResNet-50 is a variant of the original ResNet (Residual Network) introduced by He et al. in their seminal paper "Deep Residual Learning for Image Recognition, 2015". ResNet-50 is a deep convolutional neural network renowned for its effectiveness in image classification tasks and its ability to handle very deep networks without suffering from the vanishing gradient problem.

Here are the principal caracteristics of this architecture:

- **Bottleneck Design**: The term "bottleneck" refers to the specific design of residual blocks within the network. Instead of using a series of 3x3 convolutional layers in each block, as in traditional ResNet architectures, the bottleneck ResNet-50 utilizes a bottleneck structure. This structure consists of three convolutional layers: 1x1, 3x3, and 1x1 convolutions. The 1x1 convolutions are used to reduce and then increase (or restore) dimensions, while the 3x3 convolution serves as the main processing layer. This design significantly reduces computational complexity while maintaining representational power.

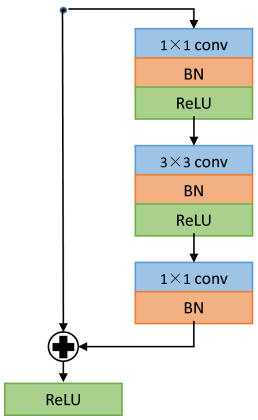

Figure 16: Bottleneck Block

- **Skip Connections:** One of the key features of ResNet-50 is its use of skip connections or shortcuts. These connections bypass one or more layers and add the original input to the output of deeper layers. This mechanism helps to mitigate the vanishing gradient problem by allowing gradients to flow more easily during training. As a result, the model can learn more effectively from the data and converge faster, which is advantageous in scenarios where large datasets are not readily available, such as satellite pose estimation.

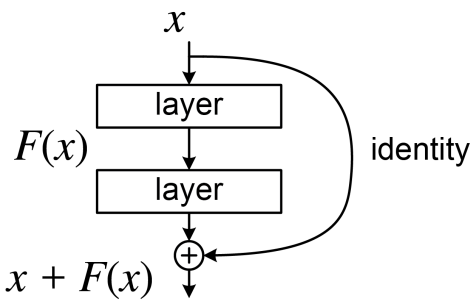

Figure 17: Diagram of Skip Connections

- **Global Average Pooling:** Towards the end of the network, a global average pooling layer is applied to collapse the spatial dimensions of the feature maps into a single vector. This pooling operation provides a compact representation of the features extracted by the preceding layers, which is then fed into the fully connected layers for final prediction.

We chose the bottleneck ResNet-50 architecture for our pose estimation pipeline because of its efficient design. By leveraging the bottleneck structure, skip connections, and global average pooling, the model can effectively learn from satellite imagery data while maintaining computational efficiency. This architecture optimizes our pipeline by striking a balance between model complexity and performance, making it well-suited for real-world applications where resource constraints are a concern.

4.4 Model Output and Evaluation

Output of the Model: The output of our model consists of predicted coordinates for each keypoint in the satellite imagery, along with an "inside" indicator. For each keypoint, the model predicts the x and y coordinates within the image, indicating the location of the keypoint. Additionally, it predicts whether the keypoint is inside the object of interest (e.g., a satellite panel) or not. These predictions are represented as a set of coordinates and binary indicators for each keypoint.

Evaluation of the Performance: The performance of the model is evaluated using standard metrics for regression tasks, such as mean squared error (MSE) or mean absolute error (MAE), to measure the discrepancy between the predicted coordinates and the ground truth coordinates. Additionally, accuracy metrics can be computed for the "inside" indicators to assess the model's ability to correctly classify keypoints as inside or outside the object. The evaluation process involves comparing the model's predictions with annotated ground truth coordinates to quantify its accuracy and robustness.

4.5 Using Perspective n Point to estimate the pose of the satellite using key points given by the CNN

Now that the CNN has learned to identify key points of the satellite, we can use those key points to estimate the pose of the satellite. To do so, we use the Perspective n Points algorithm, known as PnP. The PnP takes in 2D coordinates of the key points of the satellite, and estimates its 3D pose, in other words its rotation and translation in the camera referential. In our project, the implementation of the PnP is coded in Python. The actual PnP calculations are done using the PnP solvers of the cv2 library.[6] Here is how we calculate the pose of the satellite based on the keypoints 2D coordinates the CNN gives us.

4.5.1 Bypassing inaccurate coordinates by defining inliers and outliers using RANSAC

The two main functions called to do the job are solvePnPRansac and solvePnP,from the cv2 library. Both are solvers for pose calculation based on PnP algorithm, but solvePnPRansac introduces the concepts of inliers and outliers. In a dataset, outliers are data that are too extreme or incorrect to be used, and inliers are data that we want to use. In our pose estimation problem, the key points coordinates are given by our CNN. But the CNN can be mistaken and produce inaccurate or incorrect coordinates; those are our outliers. RANSAC, standing for random sample consensus, is used to classify which point is an inlier and which point is an outlier when fitting the coordinates to a pose with PnP. By integrating RANSAC to our PnP solver, we are able to deduce the pose of the satellite even if some of the coordinates given by our CNN are incorrect. RANSAC estimates which coordinates are inliers, meaning they are reliable, and which coordinates are outliers to be discarded. Then it gives the inliers coordinates to solvePnP, and solvePnP estimates a pose based on reliable coordinates only.

4.5.2 Defining parameters representing the 3D model of the satellite and the camera parameters

The solvePnPRansac and solvePnP functions both need the 3D representation of the satellite, so it can "recognize" it by fitting the 3D representation it has. In addition, both solvers need parameters referring to the camera, the intrinsic matrix and the distortion coefficients. Those two parameters define the internal parameters of the camera. They need to be adjusted to the camera model that will be onboard the satellite.

To sum up, we use a PnP solver of the cv2 library to compute the pose of the satellite based on key points given by the CNN. We start by using a solver with Random Sample Consensus to extract inliers, then we use those inliers to estimate a reliable pose of the satellite. The 3D model of the satellite on which we estimate the pose on, and the internal parameters of the camera onboard the satellite that estimates the pose, are parameters that we give to the PnP.

4.6 Difficulties encountered

The main difficulty in the PnP section was understanding the concepts of PnP and Ransac, how they work and how to use them. We had to dive into documentations, articles and courses to grasp the concepts and learn how to use those mathematical tools to archieve our goals.

5 Results and comparison

We would have liked to be able to compare the performance obtained with the performances of the SPEED+ dataset and the results of other teams of engineers who have worked on this topic. After finishing all the project stages and putting them together, all we needed to do was to train the model. Unfortunately, we do not have machines powerful enough for this training, and the machines on the SII company side were used or not available when we needed them.

We will therefore have our model trained in the coming days, but it is not certain that we will have it before the deadline for this project.

6 Conclusion

In the end, we built an architecture capable of generating a dataset and using it to train an AI to then estimate the pose of a satellite, all starting from an image. This solution can be used on board satellites, provided that there are machines powerful enough on board to be able to carry out its energy-intensive steps in a short time.

This project gave us all a lot of technical skills while working in a context that we are not used to experiencing. Also, the subject of satellites and space is a different subject from what we are used to studying and it is nice to be able to learn for our studies while discovering new things.

In addition, it was frustrating to have so little time to move forward on the project, having so many other things to do on the side for our studies. A project of this magnitude would have deserved to be able to delve into it 100%.

References

[1] R. Amalvy, "Le satellite metop-c part avec la fusée soyouz... et des technologies made in france," Nov. 2018. [Online]. Available: https://www.usinenouvelle.com/editorial/en-images-le-satellite-metop-c-part-avec-la-fusee-soyouz-et-des-technologies-made-in-france.N765844

[2] "Rocket lab website." [Online]. Available: https://www.rocketlabusa.com/space-systems/solar/

[3] Blender Documentation Team, *Blender 4.1 Manual*. Blender Documentation Team, 2024, ch. Principled BSDF. [Online]. Available: https://docs.blender.org/manual/en/latest/render/shader_nodes/shader/principled.html

[4] T. H. Park, M. Märtens, M. Jawaid, Z. Wang, B. Chen, T.-J. Chin, D. Izzo, and S. D'Amico, "Satellite pose estimation competition 2021: Results and analyses," *Acta Astronautica*, vol. 204, pp. 640–665, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094576523000048

[5] T. H. Park, M. Märtens, G. Lecuyer, D. Izzo, and S. D'Amico, "Speed+: Next-generation dataset for spacecraft pose estimation across domain gap," *2022 IEEE Aerospace Conference (AERO)*, 2021. [Online]. Available: https://arxiv.org/abs/2110.03101

[6] OpenCV documentation team, *OpenCV manual 4.10.0-dev*. OpenCV documentation team, 2014, ch. Perspective-n-Point (PnP) pose computation. [Online]. Available: https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html