POLITECNICO
MILANO 1863

Analysis and validation of spaceborne synthetic imagery using a Vision-Based pose initialization algorithm for non-cooperative spacecrafts

Francescodario Cuzzocrea
April 28, 2021

D-ORBIT
NEW SPACE SOLUTIONS

Introduction

Introduction

Pose determination for close range proximity operations using monocular camera

Purposes:

- Formation Flying (FF)
- On-Orbit Servicing (OOS)
- Active Debris Removal (ADR)

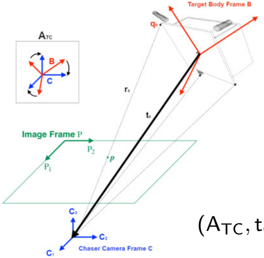
Advantages:

- Reliability
- Simplicity
- Low weight
- Low power

Challenge $\longrightarrow$ Estimate the pose of an uncooperative target from a monocular image

Monocular algorithm requirements

- Target 3D model availability
- On-Board real-time execution
- Robustness to illumination
- Robustness to background
- Robustness to variability of pose



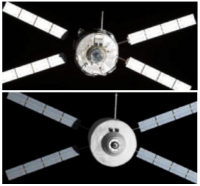
$$(A_{TC}, t_C) = ?$$

→ Need a representative data-set to develop image processing algorithms

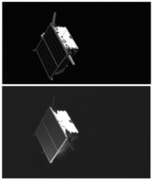
→ A lack of realism can lead to wrong assumptions and thus incoherent results

Available Solutions

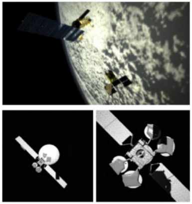
ESA PANGU



SPEED Data-Set



Airbus SurRender



URSO Data-Set

Motivation

Data-set requirements

→ Emerges the need for a tool capable of rendering images for CV algorithms training which must:

- be affordable for small research center or small companies
- be capable of rendering different kinds S/C
- be able to accomodate solar-system sized scenes
- be able to also render composite background

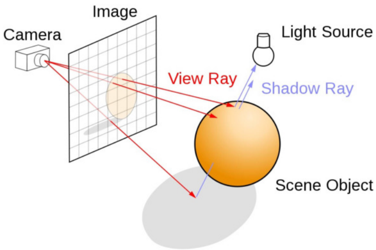
Aim of the thesis → Develop a tool capable of producing a data-set of images representative of in-orbit real operative conditions and use them to implement and test a State-of-Art pose initialization algorithm

Synthetic Image Generation

POV-Ray represents a viable solution

⟶ It simulates the physics of light!

- Support meshes for objects
- Customizable textures support
- Customizable optical properties
- Customizable light properties
- Customizable camera parameters
- Accepted by ESA in past studies



⟶ Noise can be modeled and added *a posteriori* to enhance fidelity.

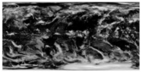
Earth and Light Modeling

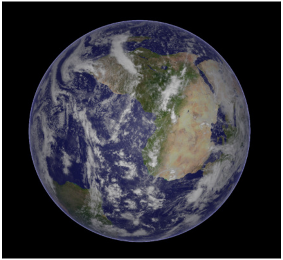
→ Use three different layers to differentiate optical parameters and textures for seasonality variations

→ Include 25 km tick atmosphere with scattering properties


Terrain


Landmask

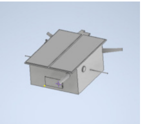

Clouds

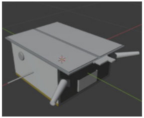


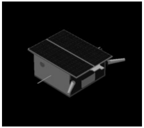
→ The Sun has been modeled using POV-Ray **area_light** feature

3D Model of the Spacecraft

⟶ Based on the TANGO spacecraft from the PRISMA mission to perform comparison with SPEED


CAD


Blender


POV-Ray

- CAD is created using Autodesk Inventor
- Imported in Blender to exploit POV-Ray add-on
- Textures are applied
- Optical parameters are defined
- SDL code is refactord for MATLAB implementation

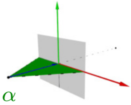
⟶ Process has to be carried out only one time per S/C

Simulate a pin-hole camera

→ POV-Ray allows to set camera aperture angle α and Aspect Ratio:

$$CCD_{size} = d_u \cdot N_u$$

$$\alpha = 2 \cdot \arctan\left(\frac{CCD_{size}}{2 \cdot f_x}\right)$$



$$A.R. = \frac{N_u}{N_v}$$

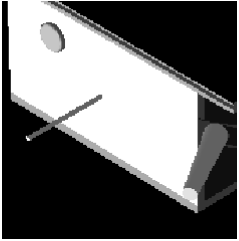
N_u	Number of horizontal pixels
N_v	Number of vertical pixels
f_x	Horizontal focal length
d_u	Horizontal pixel length

→ Under the assumption of square CCD sensor having square pixels

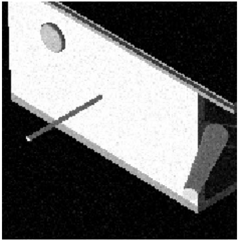
→ If camera model is known, distortion can be added in post-processing

Add Gaussian white noise and speckle noise

⟶ To enhance the fidelity noise is added to the image



Normal



Noisy

Gaussian: $\sigma^2 = 0.003$
Speckle: $\sigma^2 = 0.004$

MATLAB/Simulink Integration

Simulink Model

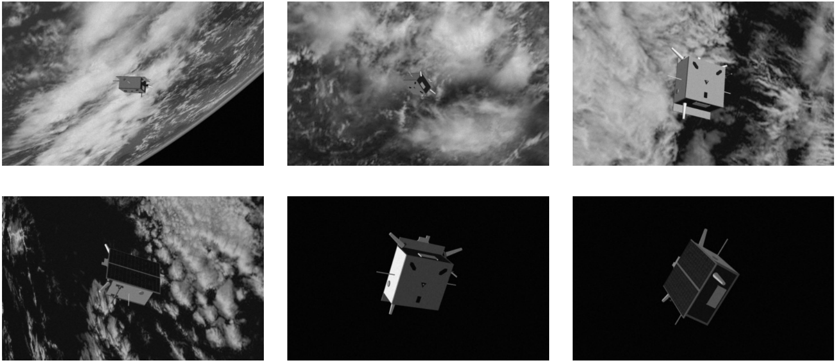
- → Simulate uncontrolled behavior on LEO orbit
- → Easily extendable to include controlled behavior too
- → Chaser constrained to be between 3 to 18 meters far from the target
- → Extendable to simulate precise chaser approach

MATLAB Functions

- → Write `.pov` SDL files for each image
- → Annotate ground truth pose for each image
- → Sets relevant POV-Ray options and run it
- → Can re-parse ground truth pose for each image

Results

Samples

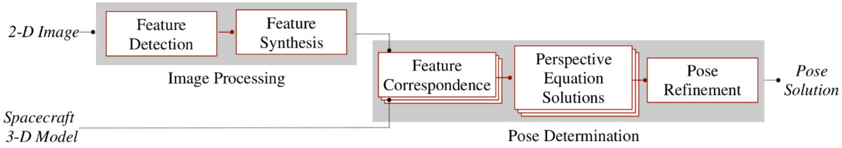


WARNING: Custom POV-Ray version needed to replicate results

The SVD Architecture for Pose Initialization

Solve the pose initialization problem

→ General architecture



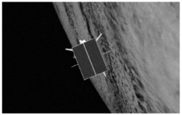
→ Enhance current State-of-the-art

- Introduces WGE technique which allows to distinguish S/C from background and allows to determine a ROI
- Uses 2D/3D feature groups to solve feature correspondence problem

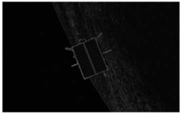
Image processing subsystem - 1

→ Weak Gradient Eliminator

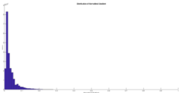
Most of the gradient intensities are weak and corresponds to the feature in the background or on the spacecraft surface → Treshold normalized gradient



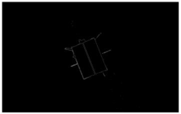
Original image



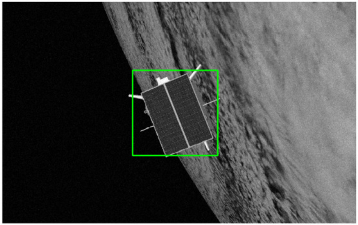
Normal gradient



Histogram

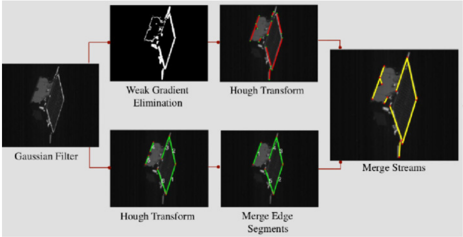


Tresholding



Found by limiting CDFs of the vertical and horizontal gradient of WGE filtered image

Image processing subsystem - 2



Feature Detection

→ Two streams of features are extracted using Hough transform:
  - WGE stream
  - S&H stream

→ Multiple truncated edges are merged into one line

- The knowledge of the ROI allows to tune Hough hyperparameters adaptively to obtain desired behavior
- Edges outside the ROI are rejected

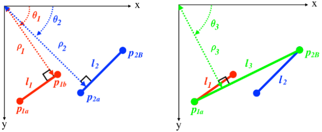
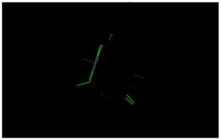


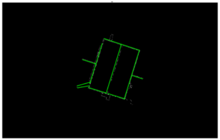
Image processing subsystem - 3

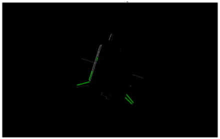
→ **Custom implementation**

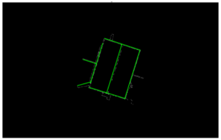
- ROI it's enlarged by adding the 5% of the mean edge length of the detected ROI to not penalize images presenting a black background
- Prior applying Hough transform to the Sobel image, to eliminate the pixel chunks belonging to smaller reflective elements from the output of Hough, all connected components (objects) that have fewer than P pixels from the input binary image are removed
- The edge merging procedure is applied to both streams

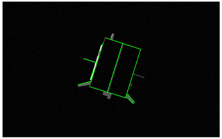
Image processing subsystem - 4

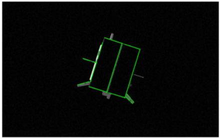

WGE stream


S&H stream


WGE merge


S&H merge

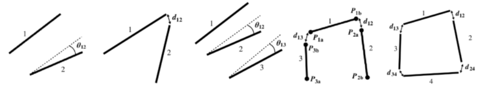

WGE + S&H


Merge stream

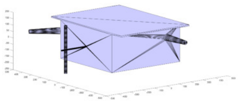
Perceptual Grouping

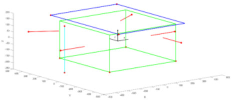
$\longrightarrow$ **Allows to reduce the search space of the correspondence problem**

2D perceptual groups are defined on the basis of some geometrical constraints



Parallel Pair | Proximity Pair | Parallel Triad | Proximity Triad | Closed Polygonal Tetrad

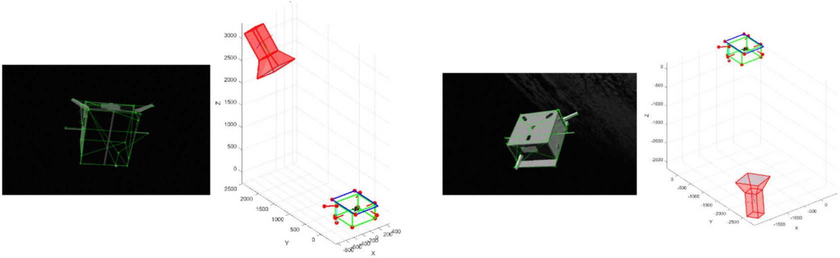




Carried out on a wireframe model too by applying 3D geometry definitions instead of 2D ones

Successful pose initialization

⟶ Obtained using P3P solver coupled with a RANSAC algorithm for outlier rejection instead of ePnP

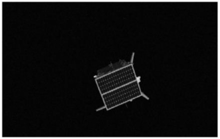


Figures are showing the map superposed to the image and the 3D camera pose in map coordinates

Results

SPEED VS Implemented Tool

→ Speed


Unfiltered Image

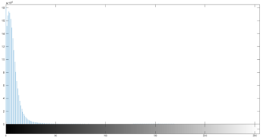
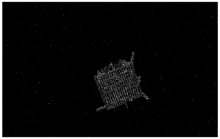
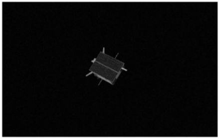


Image Histogram


Sobel Image

→ Implemented Tool


Unfiltered Image

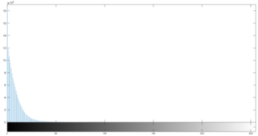
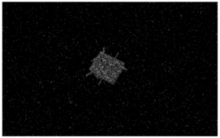
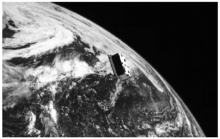


Image Histogram


Sobel Image

SPEED VS Implemented Tool

→ Speed



Unfiltered Image

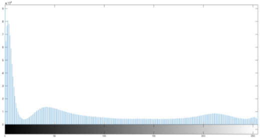
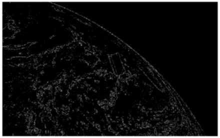
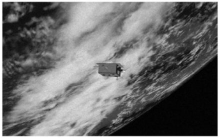


Image Histogram



Sobel Image

→ Implemented Tool



Unfiltered Image

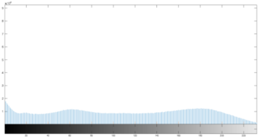
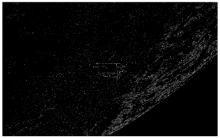


Image Histogram



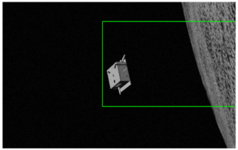
Sobel Image

SVD Algorithm

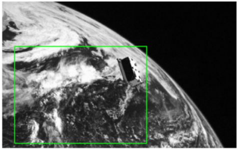
Failures

ROI Detection ⟶ Due to composite background

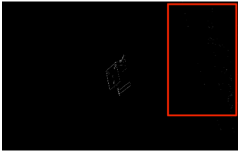
Implemented Tool | SPEED



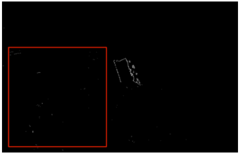
Detected ROI



Detected ROI



Tresholded Gradient

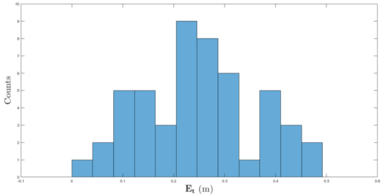


Tresholded Gradient

Rotational and translational errors

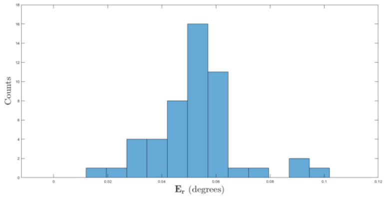
Translational Error

→ Mean error: 0.2506 m



Rotational Error

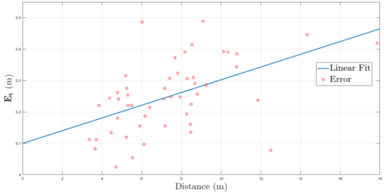
→ Mean error: 0.0528°



Errors WRT inter-spacecraft distance

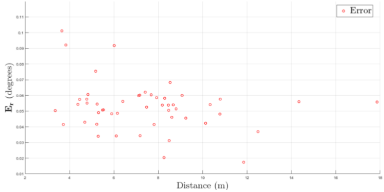
Translational Error

→ Increases with distance



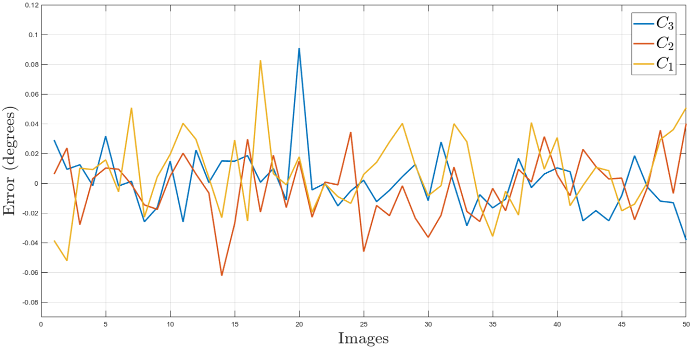
Rotational Error

→ No recognizable trend



Camera axis error

⟶ No preferred axis



Conclusions

Results Summary

Image Generation Tool

- → Image toolbox is fully automated
- → Controlled and uncontrolled behavior can be simulated as well as well as approach
- → Generated images are compatible with data-sets claimed to be representative by the scientific community

SVD Algorithm

- → Successfully tested SVD algorithm
- → Results seems to be promising

Improvements

Image Generation Tool

- → Use better textures for the S/C and the Earth
- → Use a more rigorous method to set optical parameters
- → Improve noise model
- → Patch POV-Ray code to allow GPU cores usage for rendering

SVD Algorithm

- → Employ separated Hough transforms to detect different geometric shapes
- → Improve edge detection procedure (solar panels, distance are an issue)
- → Compare different pose solvers
- → Hardware-in-the-loop experiments to evaluate the computational time on real H/W.

The End

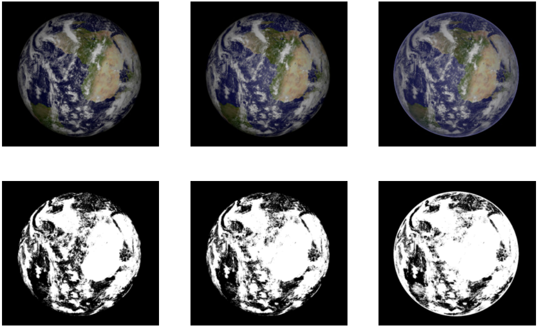
Thanks for your attention!

If you have any question, feel free to ask!

Additional Material

Earth Modeling

→ Binary image Comparison



Light Modeling

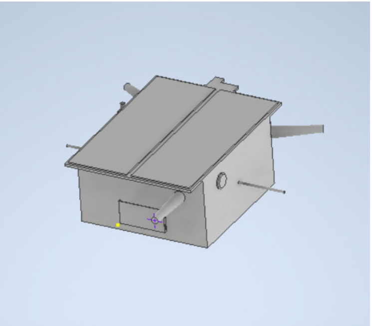
⟶ The Sun can be modeled using POV-Ray **area_light** feature:

- it create a cluster of point-like light sources distributed on a disc
- the radius of the disc is set to the radius of the Sun, and placed at the exact distance which the Sun has from the Earth
- the **orient** option makes every object see the Sun's disc as oriented toward it, from any position around it

3D Model of the Spacecraft

→ Based on the TANGO spacecraft from the PRISMA mission:

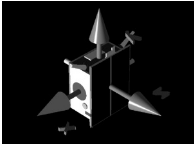
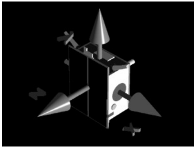
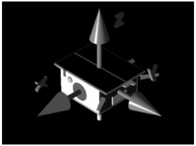
- 570 mm x 759 mm solar panel
- 560 mm x 550 mm x 300 mm body
- 4x 204 mm antennas



Reference Frames and Ground Truth Pose

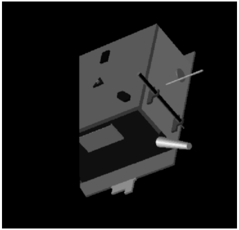
→ POV-Ray by default works using a Left Handed Coordinate System

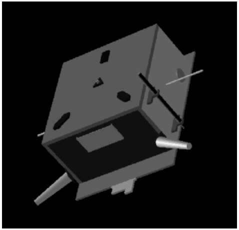
- Flip it to use a Right Handed Coordinate System
- Compute camera attitude by the knowledge of camera position and camera looking direction
- Compute ground truth pose by knowledge of camera attitude and target attitude

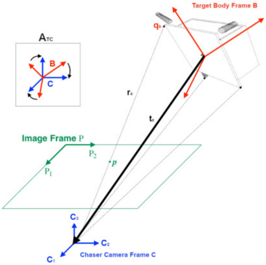
Bounding issue

This system compartmentalizes all finite objects in a scene into invisible rectangular boxes that are arranged in a tree-like hierarchy. Before testing the objects within the bounding boxes the tree is descended and only those objects whose bounds are hit by a ray are tested.


Bounding on


Bounding off

Pose estimation problem



<u>2D/3D true perspective equations</u>

$$\mathrm{r_C} = [x_C \quad y_C \quad z_C]^T = \mathrm{A_{TC}}\,\mathrm{q_B} + \mathrm{t_C}$$

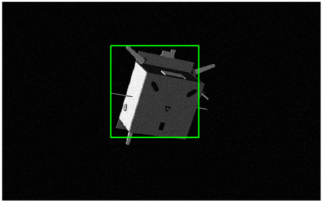
$$\mathrm{p} = \left[\frac{x_C}{z_C} f_x + C_x, \ \frac{y_C}{z_C} f_y + C_y\right]$$

$$\longrightarrow (\mathrm{A_{TC}}, \mathrm{t_C}) \longleftarrow$$

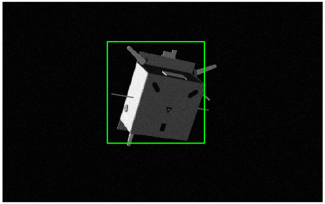
- Highly non linear and can have ∞ solutions if undercostrained
- No *a priori* knowledge of the correspondence between $\mathrm{q_B}$ and p
- Image has to be corrected for pixel's non-quadratism and lens distortion

Image processing subsystem

ROI it's enlarged by adding the 5% of the mean edge length of the detected ROI to not penalize images having a black background from a too strict tresholding of the normalized gradient.



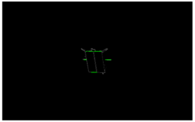
Detected ROI

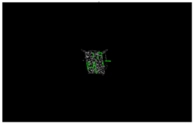


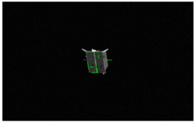
Enlarged ROI

Failures

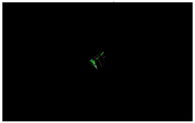
Edge Detection ⟶ Due to solar panels or distance

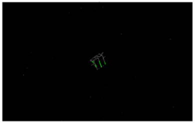

WGE Stream

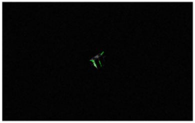

S&H Stream


Merging streams


Tresholded Gradient


WGE Stream


S&H Stream


Merging Streams