



POLITECNICO
MILANO 1863

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

Francescodario Cuzzocrea
April 20, 2021



Introduction

Table of contents

- ① Introduction
- ② Synthetic Image Generation
- ③ The SVD Architecture for Pose Initialization
- ④ Results
- ⑤ Conclusions

Aim of the thesis → Develop a data-set of images representative of in-orbit real operative conditions and test them using a Computer Vision (CV) algorithm.

Nowadays, different classes of missions envision a major role for close-range proximity operations

Purposes:

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

The main challenge when performing close-range operations in OOS and ADR missions is when the target S/C may be **uncooperative** (no markers or other specific supportive means). Thus, when operating in close-proximity the time evolution of the target's **pose** must be estimated in autonomy by exploiting the sensors available on the S/C actively performing the servicing maneuvers.

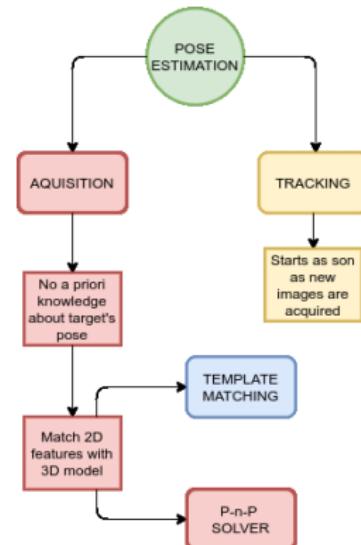
Monocular Based Pose Estimation

Monocular Solution

Requirements

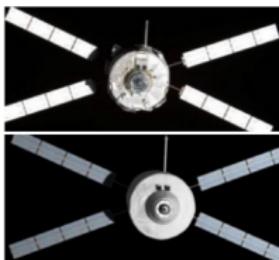
- TARGET 3D MODEL AVAILABILITY
- ON-BOARD REAL-TIME EXECUTION
- ROBUSTNESS TO ILLUMINATION
- ROBUSTNESS TO BACKGROUND
- ROBUSTNESS TO VARIABILITY OF POSE

→ Need a **representative** data-set to develop complex **CV** algorithms



Synthetic Image Generation: commercial solutions

ESA PANGU



Need to be ESA project contractor

- OpenGL image rendering
- Can make use of GPU cores
- Closed-Source

AIRBUS SURRENDER

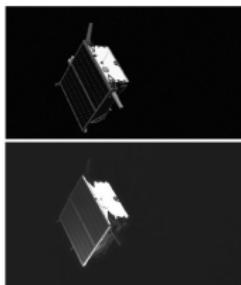


Need to buy a license from Airbus

- Ray tracing image rendering
- Fully parallelized
- Closed-Source

Synthetic Image Generation: academical solutions

SPEED DATA-SET



Only the Tango S/C is represented in the images.

- TRON facility for real images
- OpenGL image rendering
- Closed-Source

URSO DATA-SET



Lack of photometric accuracy

- Implemented using Unreal Engine 4
- Developed to train neural networks
- Open-Source rendering engine

Synthetic Image Generation

Introduction

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
- The images have to be **representative** of a real data-set taken by a true camera!
- A lack of realism can lead to **wrong** assumptions and thus **incoherent** results!!!

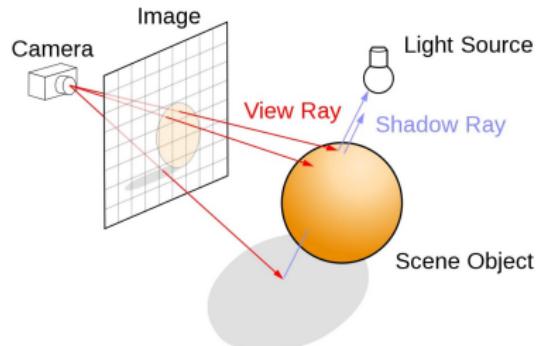
Introduction

Ray Tracing

POV-Ray represents a viable solution

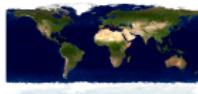
- Support **mesh** and objects
- Customizable optical properties
- Customizable light properties
- Customizable textures support
- Customizable camera parameters
- Used by ESA before PANGU

- It simulates the physics of light!
- Noise can be modeled and added *a posteriori* to enhance fidelity.



Earth Modeling

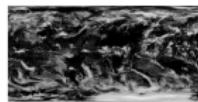
- Use three different layers to differentiate optical parameters and textures for seasonality variations



Terrain



Landmask



Clouds

Same optical
parametersDifferent
optical
parameters

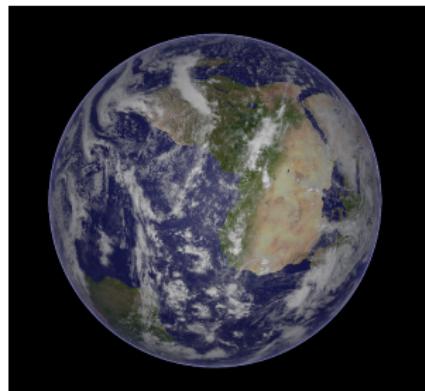
- The Sun has been modeled using POV-Ray **area_light** feature

Earth Modeling

- Add 25 km thick atmosphere augment realness and challenge edge detection CV algorithms



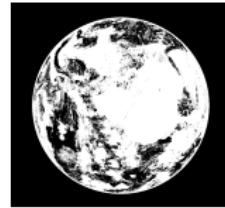
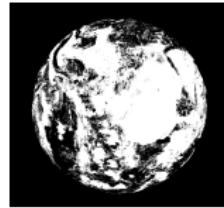
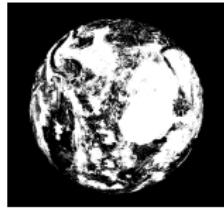
No atmosphere



Atmosphere

Earth Modeling

→ Binary image Comparison

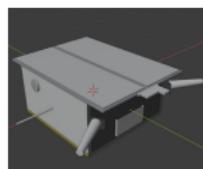


3D Model of the Spacecraft

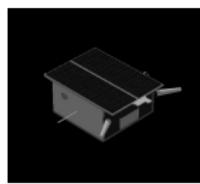
→ Based on the TANGO spacecraft from the PRISMA mission:



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 refactored for MATLAB implementation

→ Process has to be carried out only one time

Simulate a real camera

→ POV-Ray allows to set camera aperture angle:

$$A.R. = \frac{N_u}{N_v}$$
$$CCD_{size} = d_u \cdot N_u$$
$$\alpha = 2 \cdot \arctan \left(\frac{CCD_{size}}{2 \cdot f_x} \right)$$

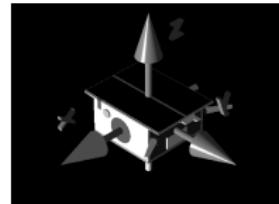
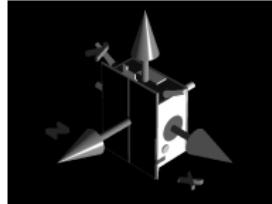
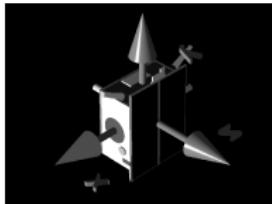
N_u	Number of horizontal pixels
N_v	Number of vertical pixels
f_x	Horizontal focal length
f_y	Vertical focal length
d_u	Horizontal pixel length
d_v	Vertical pixel length

→ Under the assumption of square CCD sensor having square pixels.

Reference Frames and Ground Truth Pose

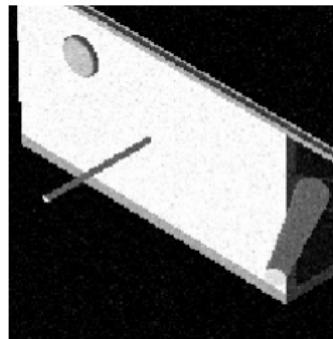
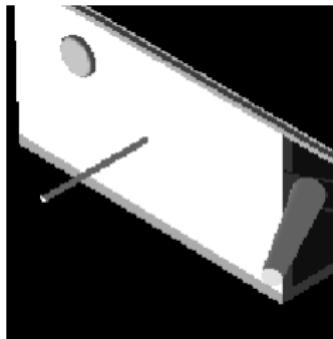
→ POV-Ray by default works using a Left Handed Coordinate System

- 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



Add Gaussian with noise and speckle noise

- To enhance the fidelity noise is added to the image



Speckle: $\sigma^2 = 0.004$

Gaussian: $\sigma^2 = 0.003$

Simulink Model

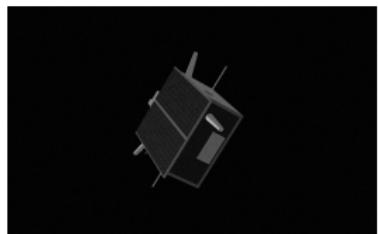
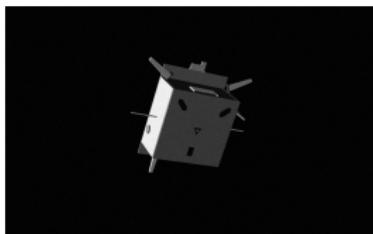
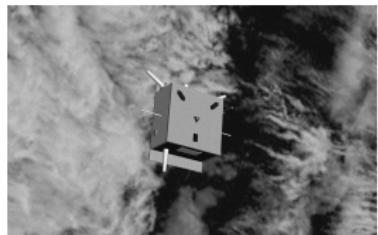
- Simulate uncontrolled behavior on LEO orbit
- Easily extendable to include controlled behavior too
- Chaser constrained to be between 3 to 18 meters far from the target
- Extendable to simulate precise chaser approach

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

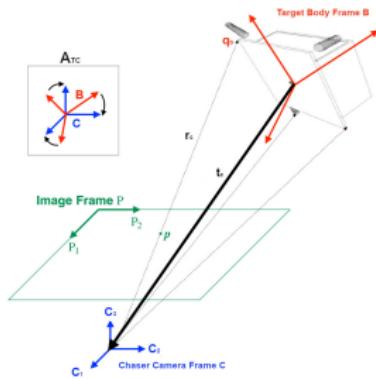


WARNING: Custom POV-Ray version needed to replicate results!

The SVD Architecture for Pose Initialization

Introduction

Pose estimation problem



2D/3D true perspective equations

$$\mathbf{r}_C = [x_C \quad y_C \quad z_C]^T = A_{TC} \mathbf{q}_B + \mathbf{t}_C$$

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

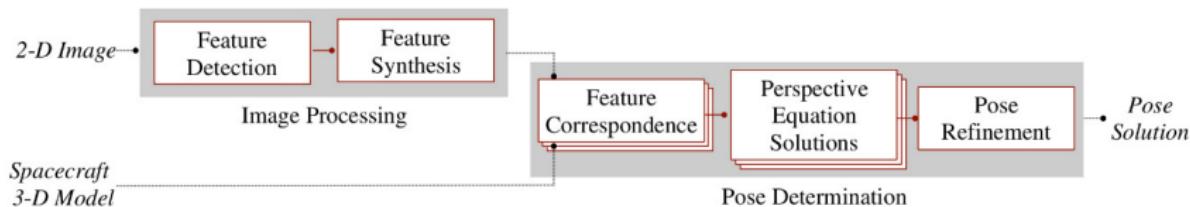
$$\rightarrow (A_{TC}, t_C) \leftarrow$$

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

Feature-based pose estimation

General architecture

→ Custom re-implementation of SVD algorithm from the ground up



→ Enhance current State-of-the-art

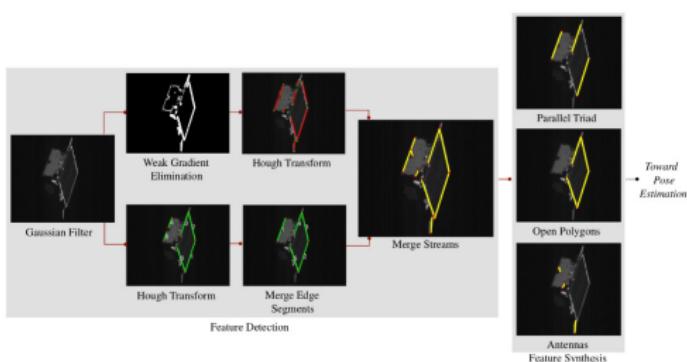
- Introduces WGE technique which allows to distinguish S/C from background and allows to determine a ROI
- Uses 2D/3D feature groups to solve feature correspondence problem

The SVD Architecture for Pose Initialization

Feature-based pose estimation

20/35

Image processing subsystem - 1



- Two streams of features are extracted using Hough transform:
 - WGE stream
 - S&H stream
- Multiple truncated edges are merged into one line
- Features are organized in perceptual groups

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

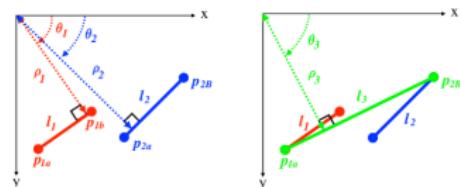
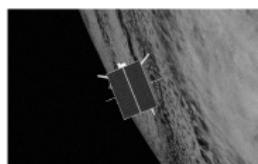


Image processing subsystem - 2

→ Weak Gradient Eliminator

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



Original image

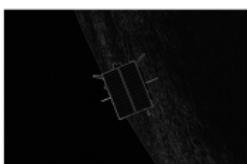


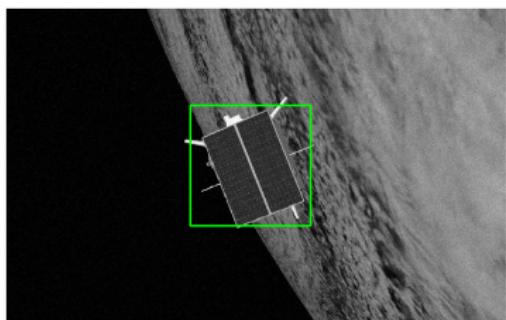
Image gradient



Histogram



Filtered image



Feature-based pose estimation

Image processing subsystem - 3

→ Custom implementation

- ROI it's enlarged by adding the 5% of the mean detected edge length 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

The SVD Architecture for Pose Initialization

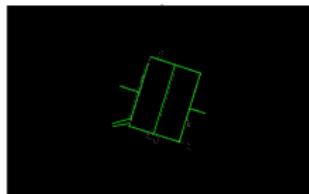
Feature-based pose estimation

23/35

Image processing subsystem - 4



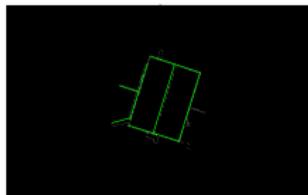
WGE stream



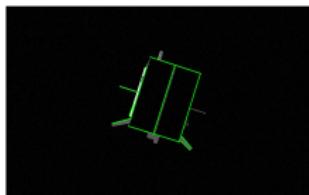
S&H stream



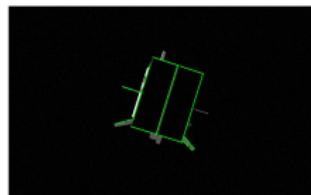
WGE merge



S&H merge



WGE + S&H

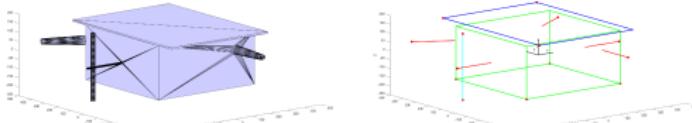
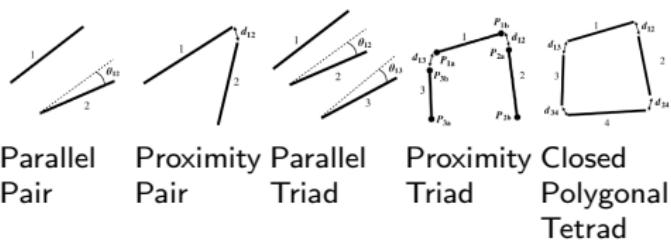


Merge stream

Perceptual Grouping

→ Allows to reduce the search space of the correspondence problem

Perceptual groups are defined on the basis of some geometrical constraints

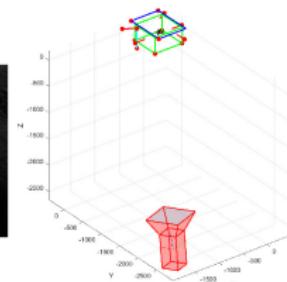
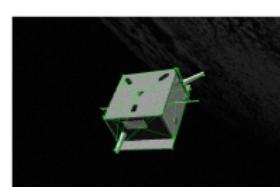
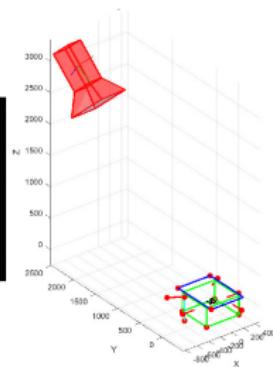
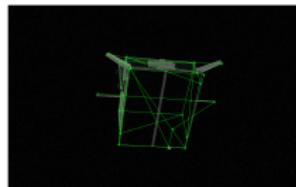


Carried out on a wireframe model too by applying 3D geometry definitions instead of 2D ones

Result

Successful pose initialization

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

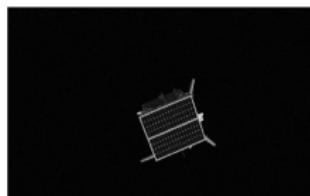


Figures are showing the map superposed to the image and the 3D camera pose in map coordinates

Results

Qualitative Assessment

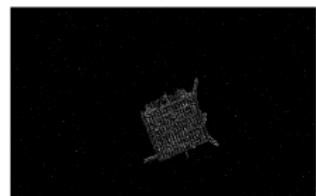
SPEED VS Implemented Tool



Unfiltered Image



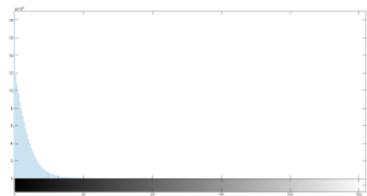
Histogram



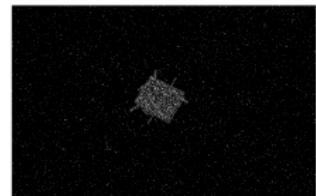
Sobel Image



Unfiltered Image



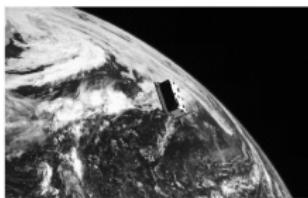
Histogram



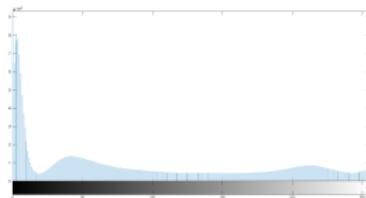
Sobel Image

Qualitative Assessment

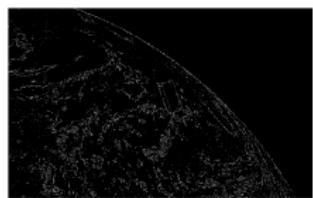
SPEED VS Implemented Tool



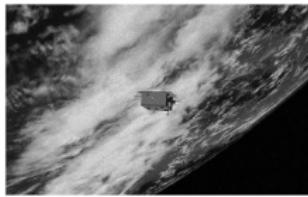
Unfiltered Image



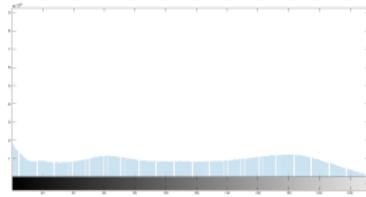
Histogram



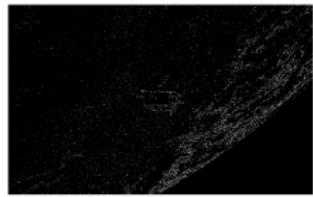
Sobel Image



Unfiltered Image



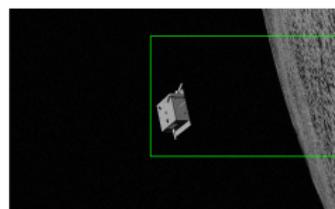
Histogram



Sobel Image

Failures

ROI Detection → Due to composite background



Detected ROI



Normalized Gradient



Thresholded Gradient



Edge Detection

Failures

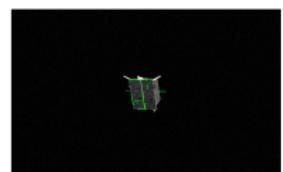
Edge Detection → Due to solar panels or distance



WGE Stream



S&H Stream



Merging streams



Thresholded Gradient



WGE Stream



S&H Stream

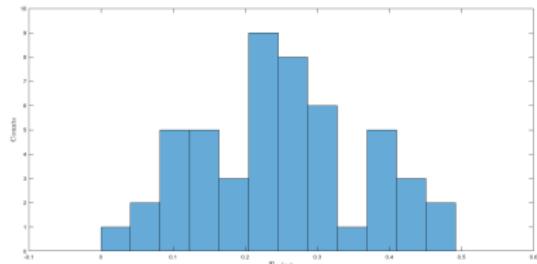


Merging Streams

Rotational and translational errors

Translational Error

→ Mean error: 0.2506 m

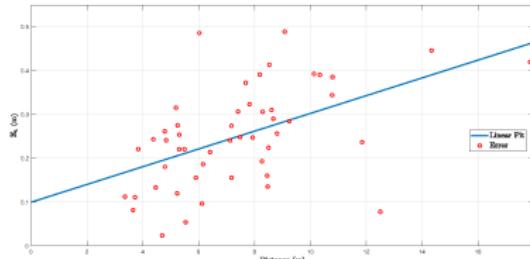


SVD Algorithm

Errors WRT inter-spacecraft distance

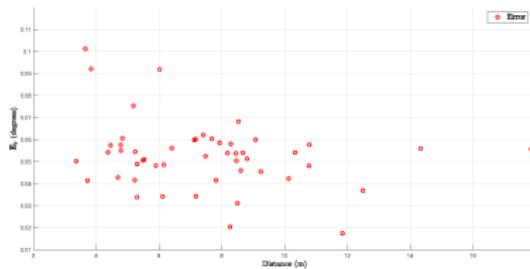
Translational Error

→ Increases with distance



Rotational Error

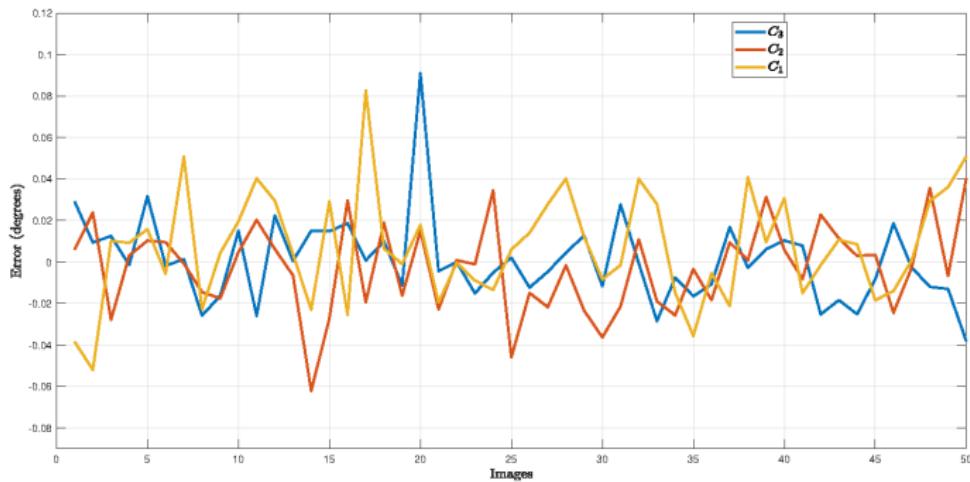
→ No recognizable trend



SVD Algorithm

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 seems are promising

SVD Algorithm

- Successfully tested SVD algorithm
- Results seems to be promising

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.

Thanks for your attention

Additional Material

Light Modeling

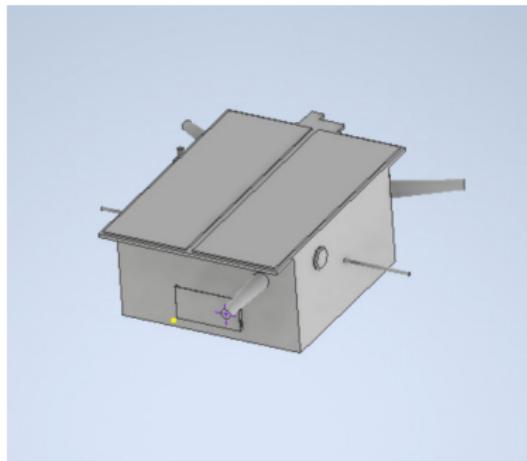
→ The Sun can be modeled using POV-Ray **area_light** feature:

- it creates 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

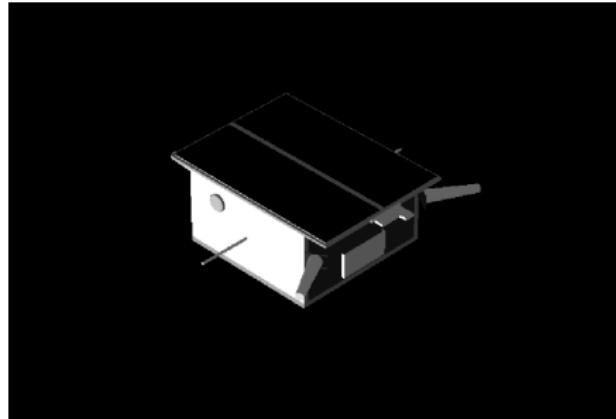
→ Based on the TANGO spacecraft from the PRISMA mission:

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



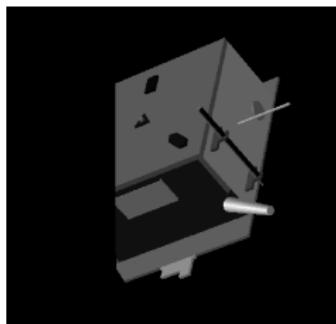
3D model rendering with POV-Ray

Blender POV-Ray add-on code has to be refactored to allow using one single S/C object

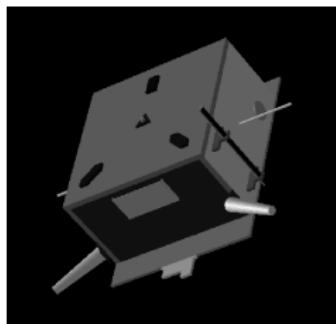


Bounding issue

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