Robotic Vision Project 2

Group 21

Karoline Amalie Halvorsen
Anders Thallaug Fagerli
Odin Aleksander Severinsen

Cite [1]

Goal

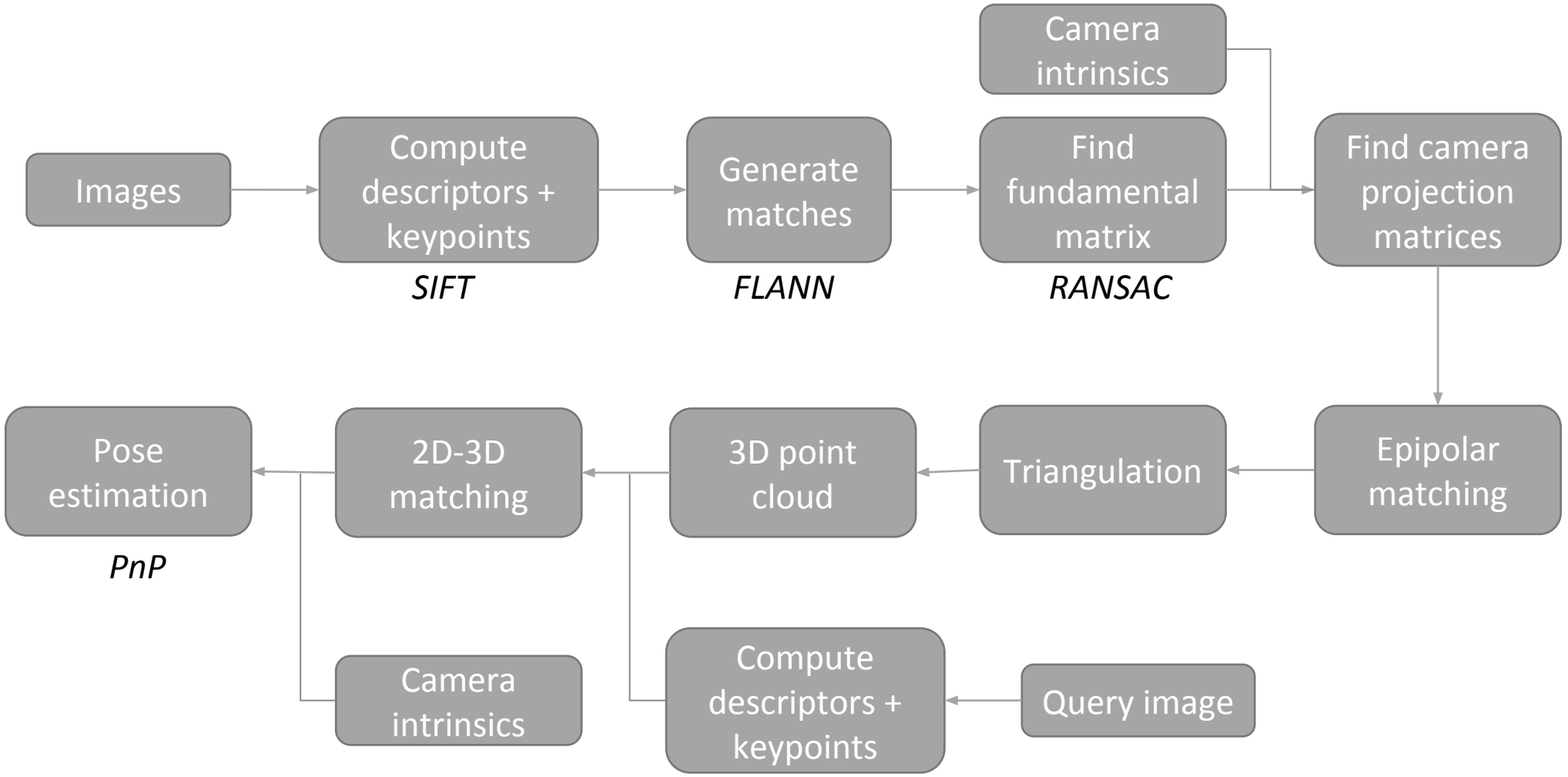
Implement a 3D structure-based localization system and determine the exact position and orientation from a query photo

Background

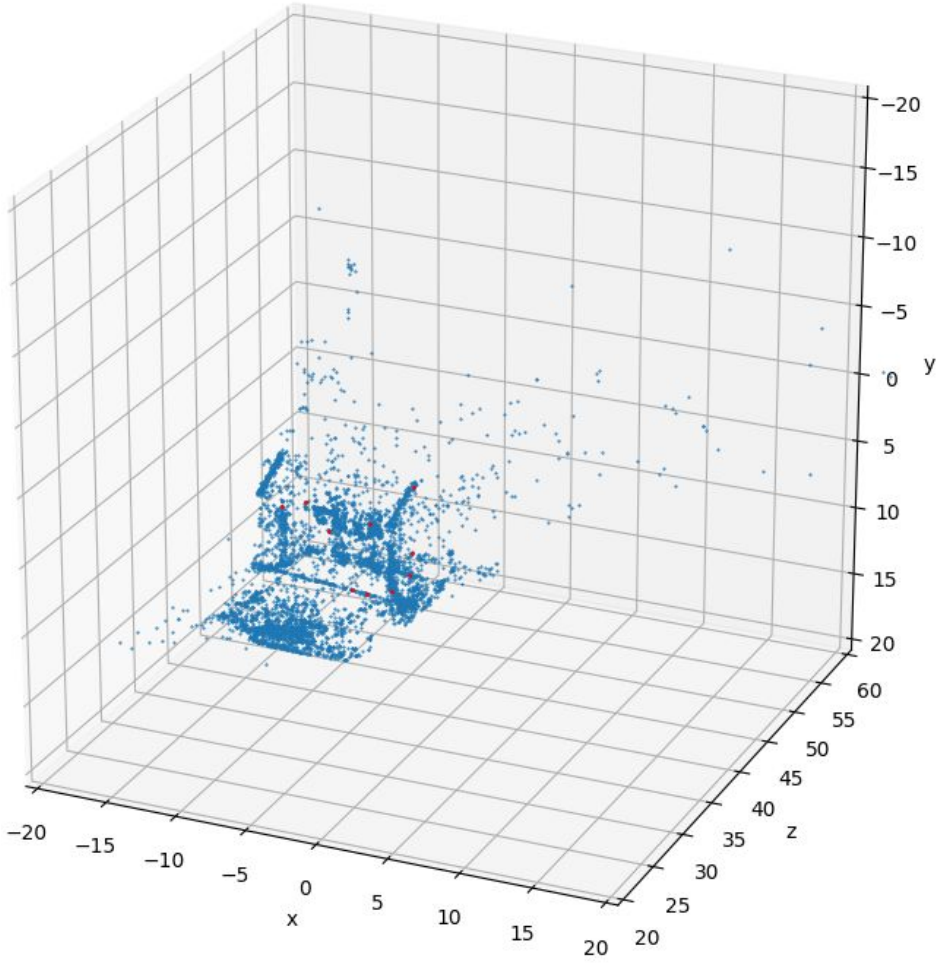
Software:

- Python + OpenCV
- MATLAB
- COLMAP

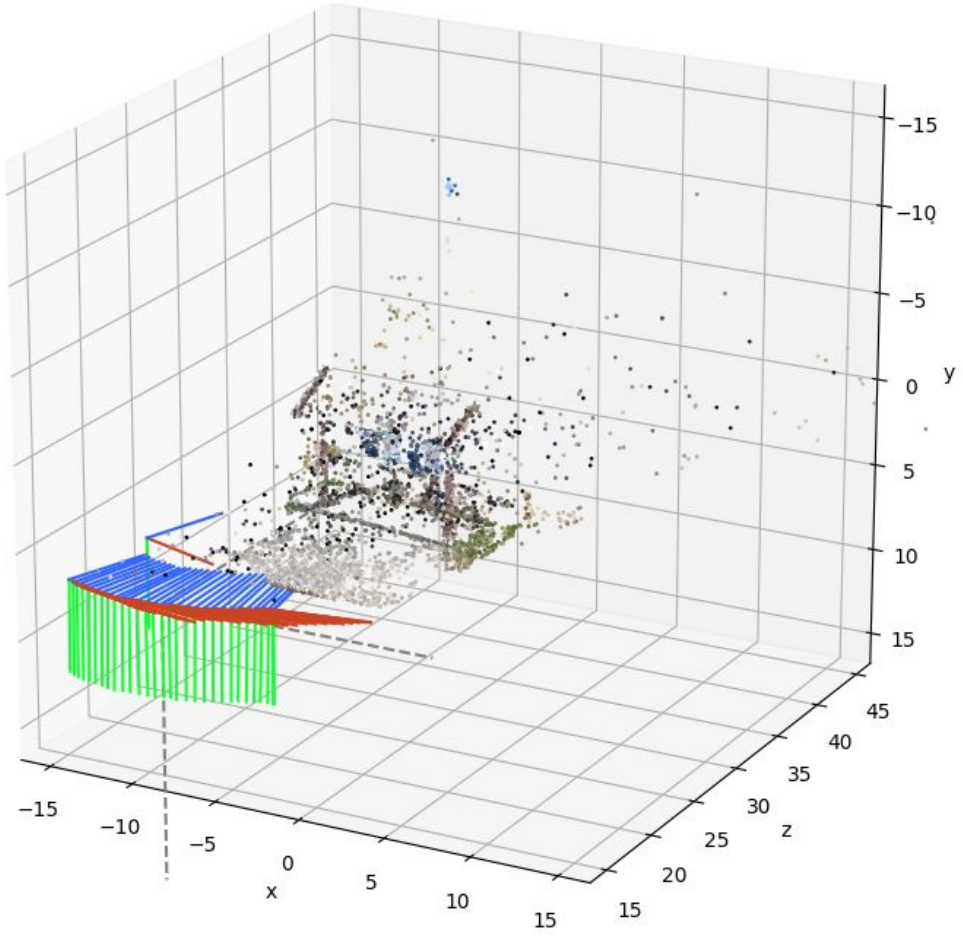
Flowchart of Project



Core functionality -
2D - 3D inliner matches for a query image



Core functionality - localization of query image with the 3D model



How we calibrated Camera:

- OpenCV:
  - Cv2.findChessboardCorners
  - Cv2.calibrateCamera
- MATLAB
  - Camera Calibrator app

- COLMAP
  - fx: 4608, fy: 4608, cx: 1920, cy: 1080

3D Reconstruction

1. Python

SfM with OpenCV

2. COLMAP

3. MATLAB

SfM with Computer Vision Toolbox

1. Python

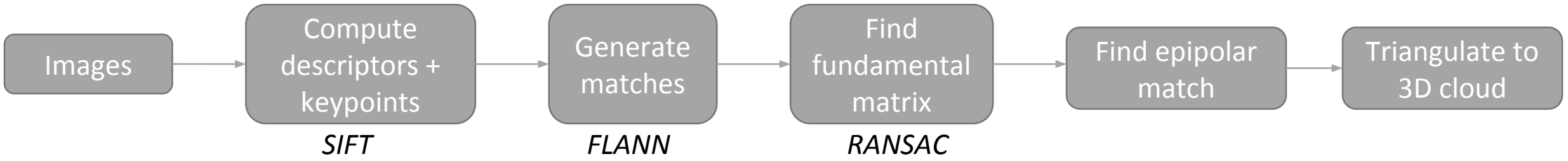
SfM with OpenCV

How did you acquire the 3D model and ensure its accuracy?

1. Python

SfM with OpenCV

- Camera intrinsic
- Descriptors- filter Lower
- Fundamental matrix with RANSAC
-



1. Python

SfM with OpenCV

Keypoints + Descriptors

Images → Compute descriptors + keypoints → Generate matches → Find fundamental matrix → Epipolar matching → Triangulate to 3D cloud

SIFT *FLANN* *RANSAC*

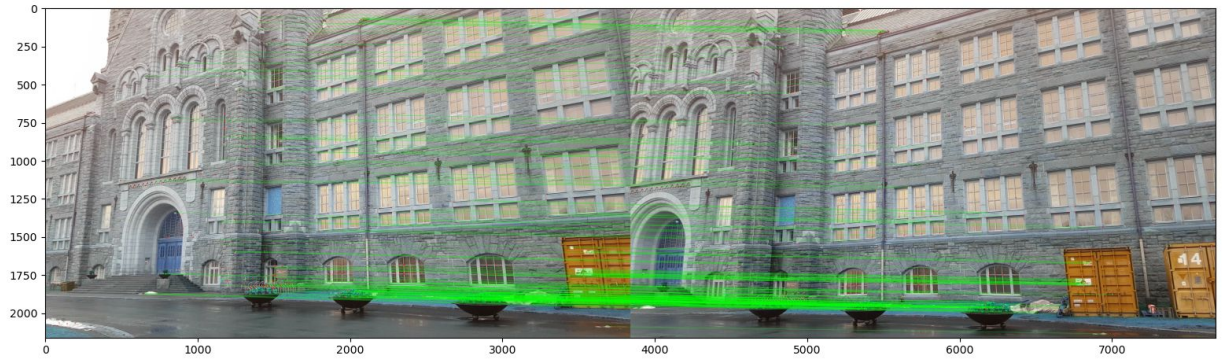
Feature matching

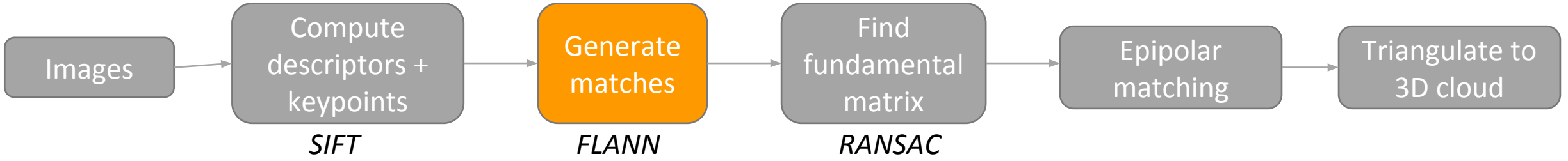
900 features
0.8 threshold Lowe test
3 trees
Search param: 50

1. Python

SfM with OpenCV

- cv2.Flann.knnMatch
- K = 2
- Ratio test: Lowe



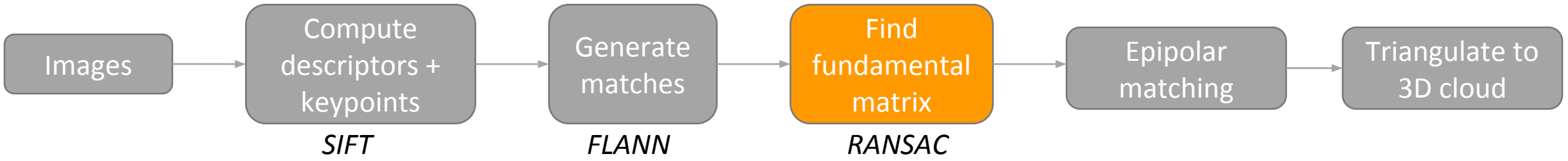


1. Python

SfM with OpenCV

Fundamental matrix

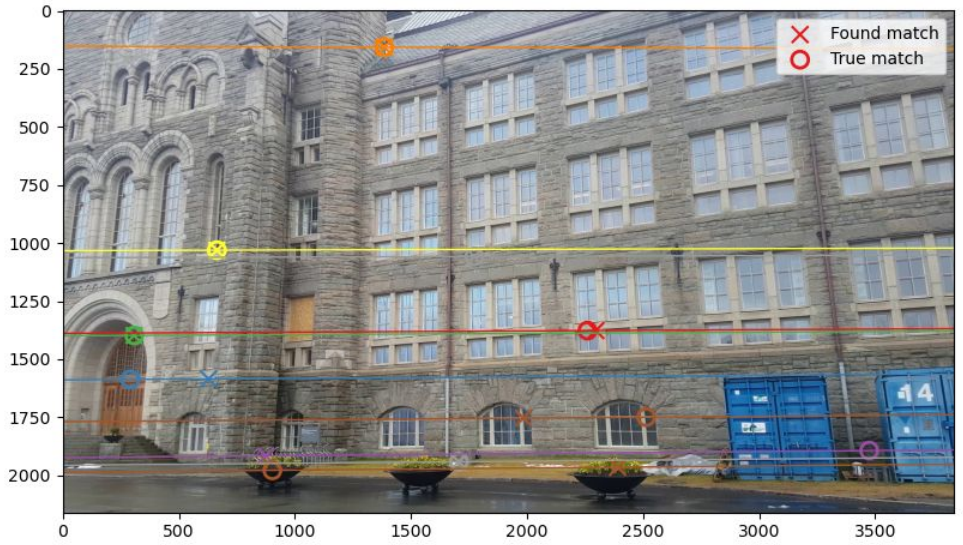
- With LMEDS or RANSAC
- More robust than 8-point algorithm



1. Python

SfM with OpenCV

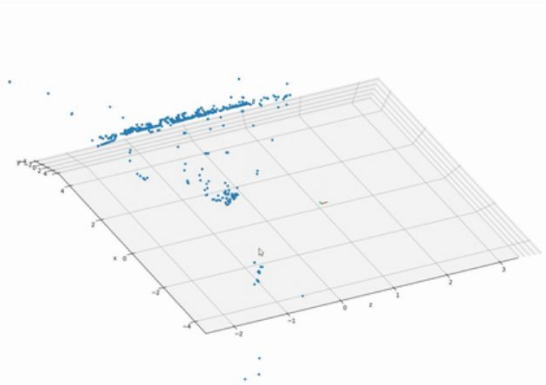
Epipolar matching

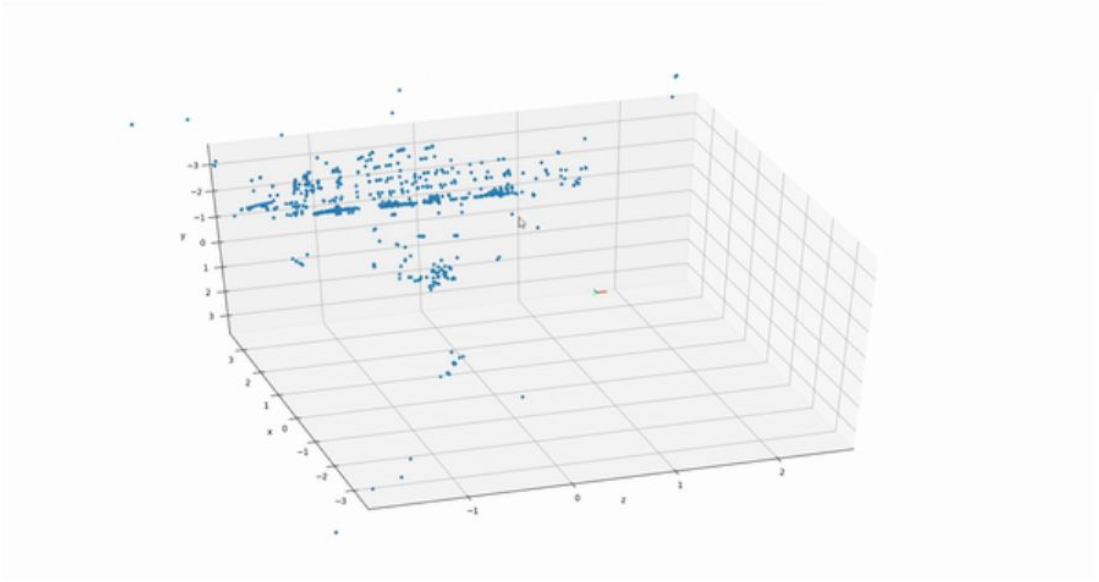


Images → Compute descriptors + keypoints → Generate matches → Find fundamental matrix → Epipolar matching → Triangulate to 3D cloud

SIFT *FLANN* *RANSAC*

Triangulation





1. Python

SfM with OpenCV

Fundamental matrix

Essential Matrix

Rotation + translation

Camera projection matrices

Images → Compute descriptors + keypoints → Generate matches → Find fundamental matrix → Find epipolar match → Triangulate to 3D cloud

SIFT

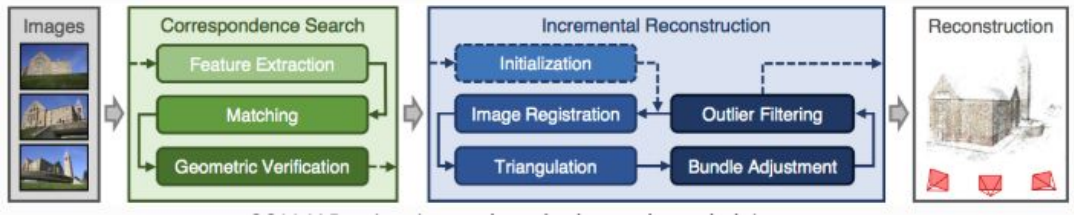
FLANN

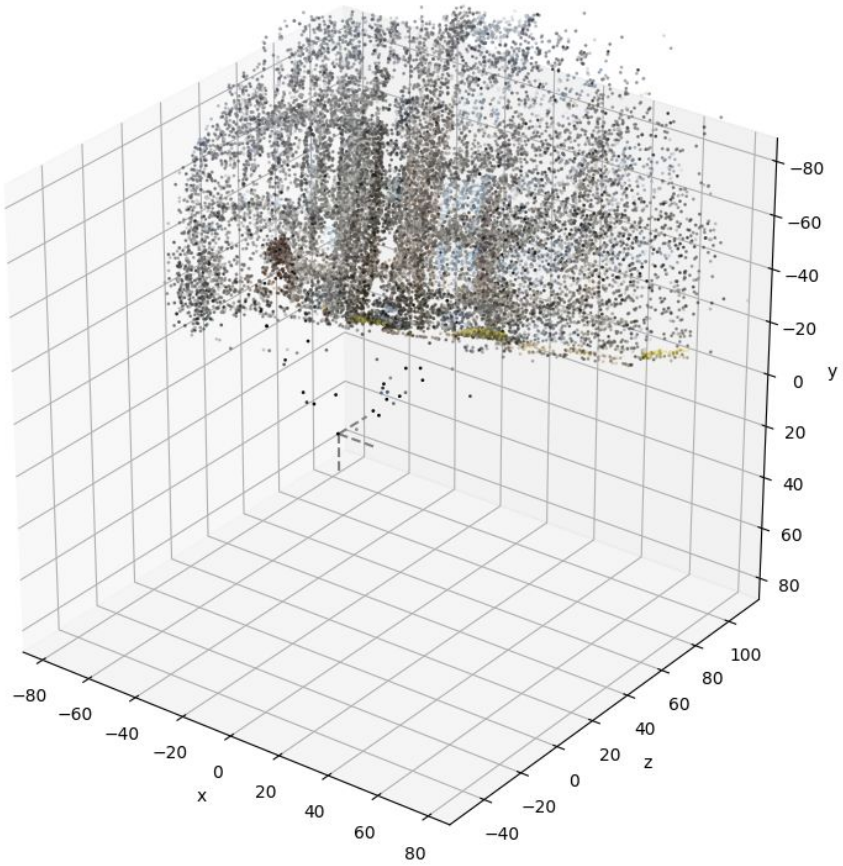
RANSAC

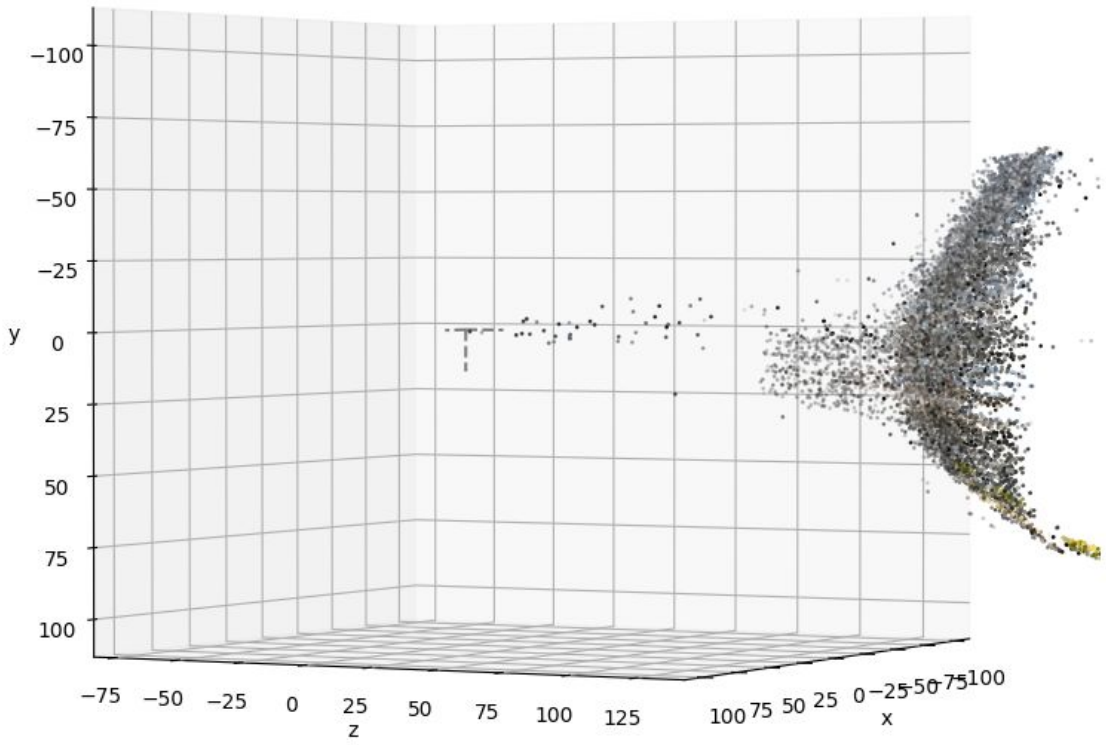
2. COLMAP

COLMAP

- Import descriptors from e.g OpenCV methods
  - Limited to 128 dimensional descriptor vector
- Feature matching by an exhaustive search
- 3D reconstruction by triangulation
- Pose estimation by bundle adjustment



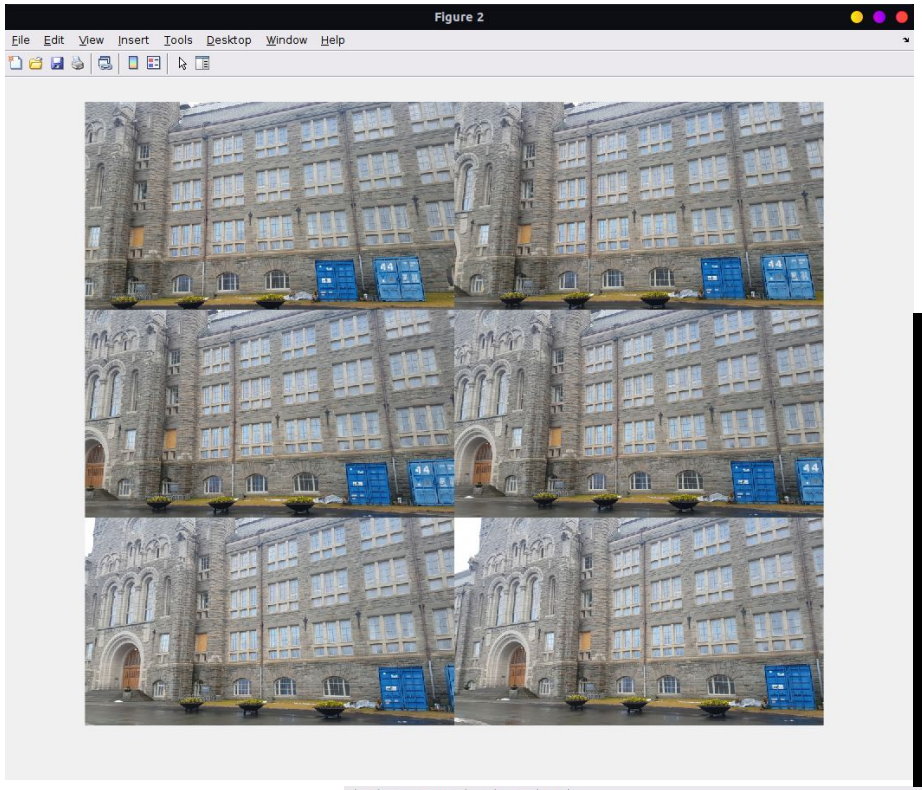

−80
−60
−40
−20
0
20
40
60
80
y
−80
−60
−40
−20
0
20
40
60
80
100
z
−40
−20
0
20
40
60
80
x

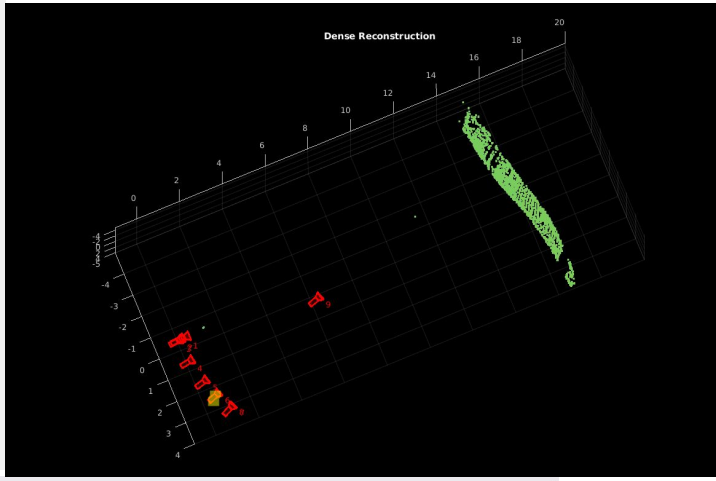

−100
−75
−50
−25
0
25
50
75
100
y
−75
−50
−25
0
25
50
75
100
125
z
100
75
50
25
0
−25
−50
−75
−100
x

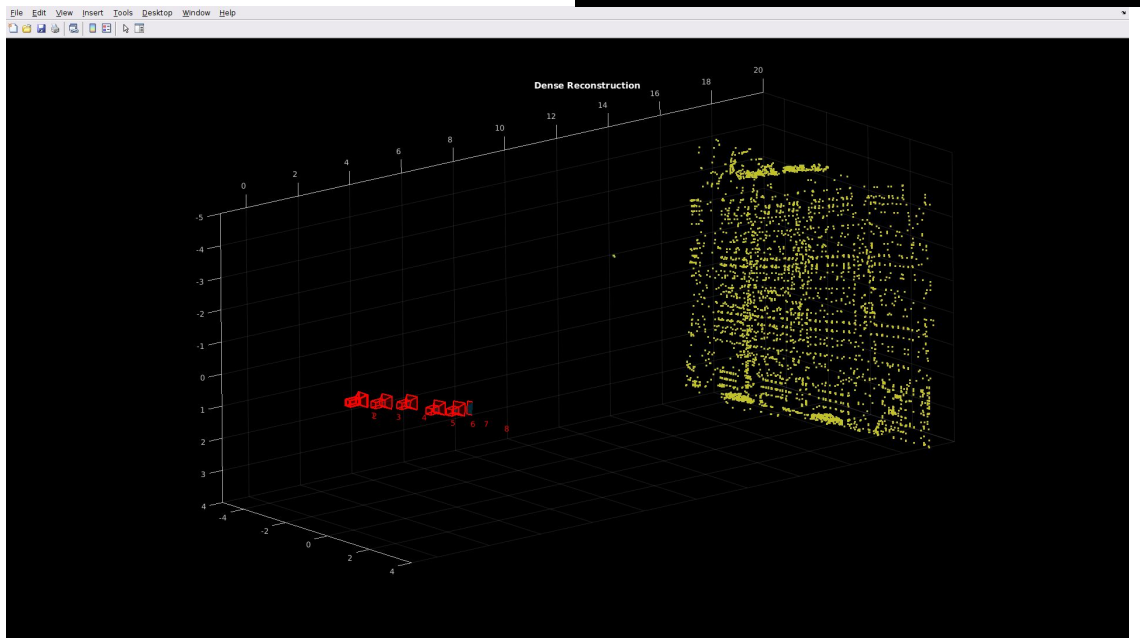
3. MATLAB

MATLAB

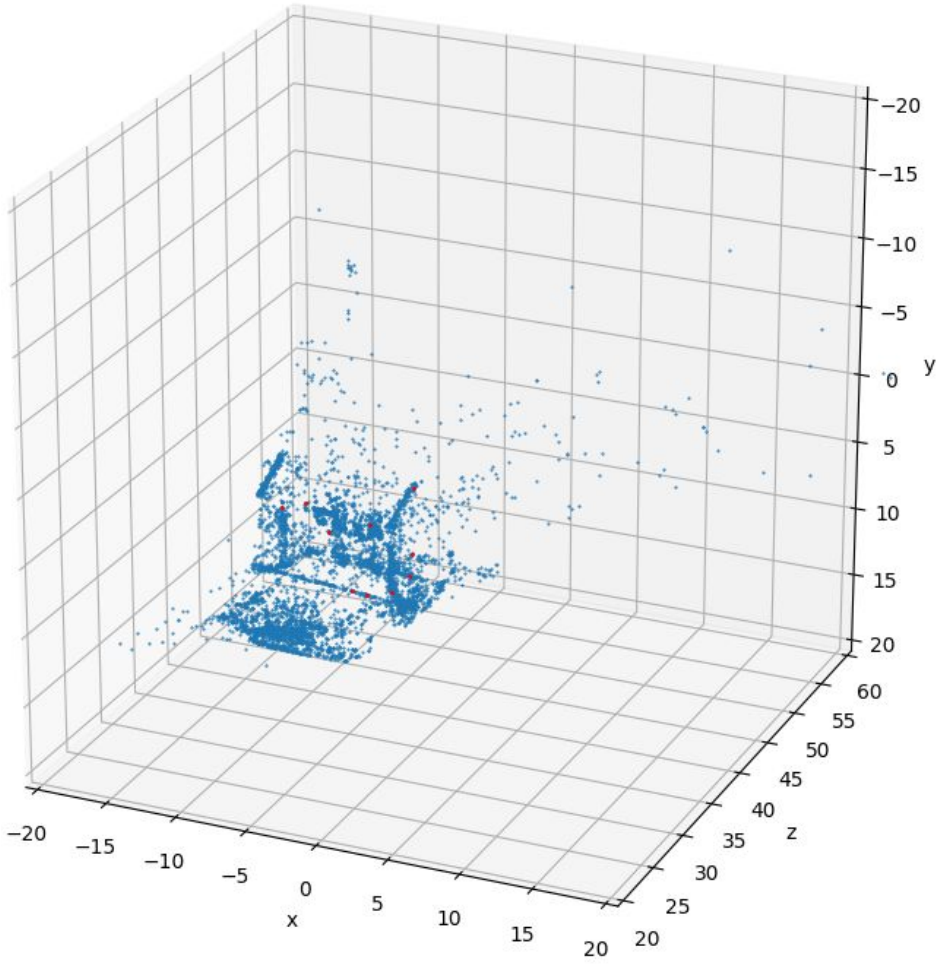
1. Build pose graph from consecutive frames, to scale
2. Retrieves scale by bundle adjustment
3. Triangulates a dense point cloud from the refined poses
4. Gathers features from the point tracks identified by Matlab



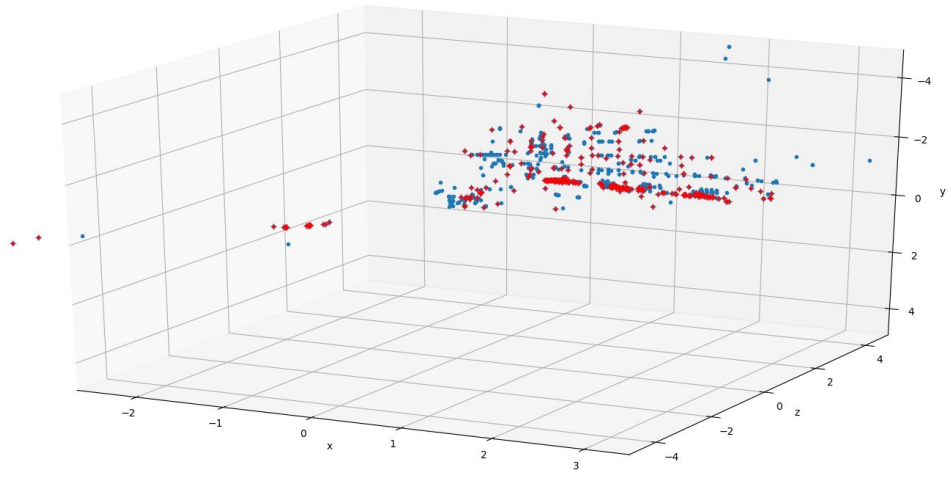




2D - 3D matching

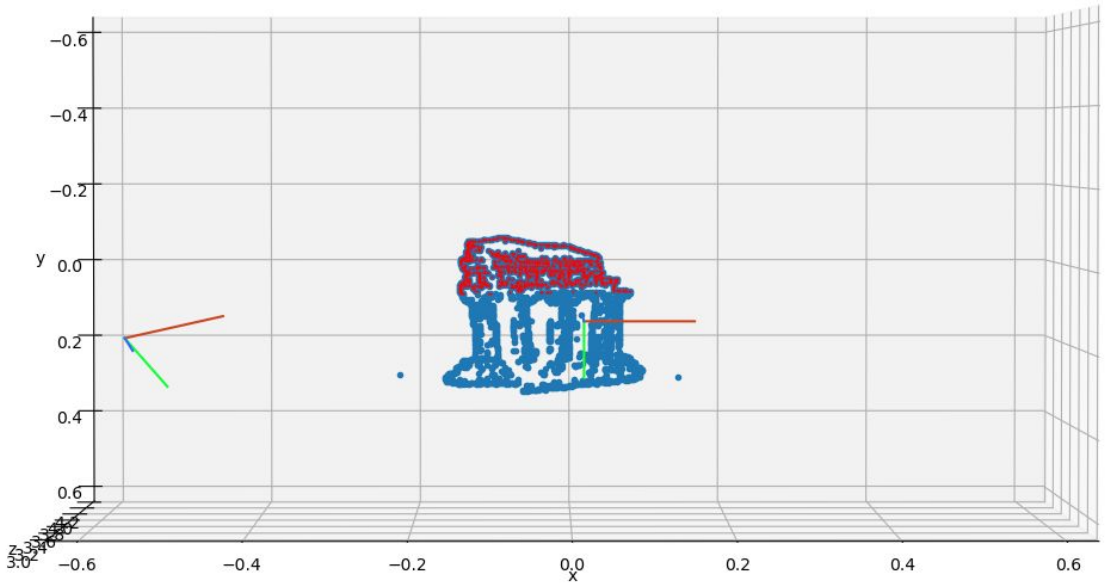


2D - 3D feature matching



Find Homography

Used method 0, that uses all the points. With Rancas to estimate



Pose estimation

1. Python

PnP with RANSAC

- Runs DLT inside a RANSAC loop to filter out outliers and compute P.
- Minimizes the reprojection error with Levenberg-Marquardt initialized with the DLT estimate

2. MATLAB

PnP with MSAC

- Uses MSAC for outlier removal in DLT
- Finds pose with P3P

Extra:

- solvePnP: Levenberg M: minimizing the reprojection error

Reference:

[1]: https://www.universeoptics.com/robotic-vision/

https://opencv.org/

https://colmap.github.io/

https://se.mathworks.com/help/vision/examples/structure-from-motion-from-multiple-views.html

https://piazza.com/class/k3o58n5yru0281