A Monocular Visual Odometry Experiment Using ORB-SLAM Library

Farhad Mirkazemi

Monocular Visual Odometry

- The Act of Determining Moving Target Position and Attitude Using Single Camera Image Sequences

- Unique Only Up to A Scale Factor

General Algorithm

- Capture Images as Input: $I_k \rightarrow k \in \{1,2,\dots\}$
- Determine Camera Intrinsics and Lens Distortion Parameters

$$w[x\ y\ 1] = [X\ Y\ Z\ 1]\begin{bmatrix} R \\ t \end{bmatrix} K$$

$$K = \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ p_x & p_y & 1 \end{bmatrix}$$

LensDistortion

- Radial distortion occurs when light rays bend more near the edges of a lens than they do at its optical center

$x_{\text{distorted}} = x(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6)$

$y_{\text{distorted}} = y(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6)$

- Tangential distortion occurs when the lens and the image plane are not parallel

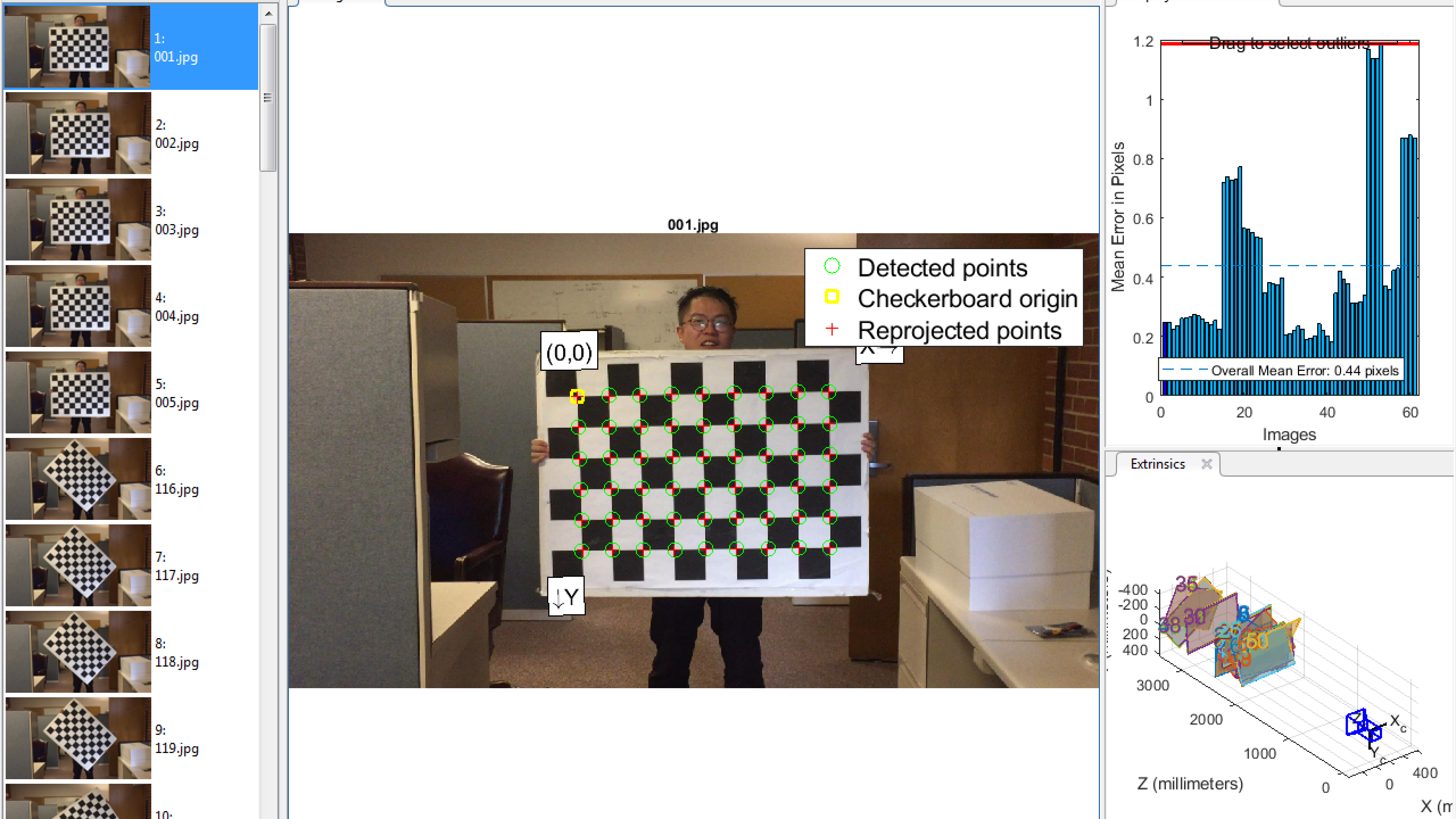
$x_{\text{distorted}} = x + (2 * p_1 * x * y + p_2 * (r^2 + 2 * x^2))$

$y_{\text{distorted}} = y + (p_1 * (r^2 + 2 * y^2) + 2 * p_2 * x * y)$

Matlab Camera Calibration Algorithm

- Solve for the intrinsics and extrinsics in closed form, assuming that lens distortion is zero

- Estimate parameters simultaneously, including the distortion coefficients, using nonlinear least-squares minimization (Levenberg–Marquardt algorithm) setting the initial estimate of the intrinsics and extrinsics to the preceding step and the distortion coefficients to zero



1: 001.jpg
2: 002.jpg
3: 003.jpg
4: 004.jpg
5: 005.jpg
6: 116.jpg
7: 117.jpg
8: 118.jpg
9: 119.jpg
10:
001.jpg
Detected points
Checkerboard origin
Reprojected points
(0,0)
↓Y
Drag to select outliers
Mean Error in Pixels
Overall Mean Error: 0.44 pixels
Images
Extrinsics
Z (millimeters)

My Iphone 6 Camera Parameters

Parameter	***Estimated Value***
Image Size$_x$	1920
Image Size$_x$	1080
f_x	1890.1946
f_y	1887.7924
p_x	934.0193
p_y	524.4038
k_1	-0.0935
k_2	0.5849
k_3	00376
p_1	-0.0030
p_2	-0.0029

General Algorithm

- Use Feature-Based Algorithm to Detect Points of Interest on Consecutive Images I_k, I_{k+1}

- ORB_SLAM uses ORB Features which are oriented multi-scale FAST (oFAST)corners with a 256 bits descriptor associated

ORB Keypoints

- FAST and its variants are the method of choice for finding keypoints in real-time systems that match visual features.

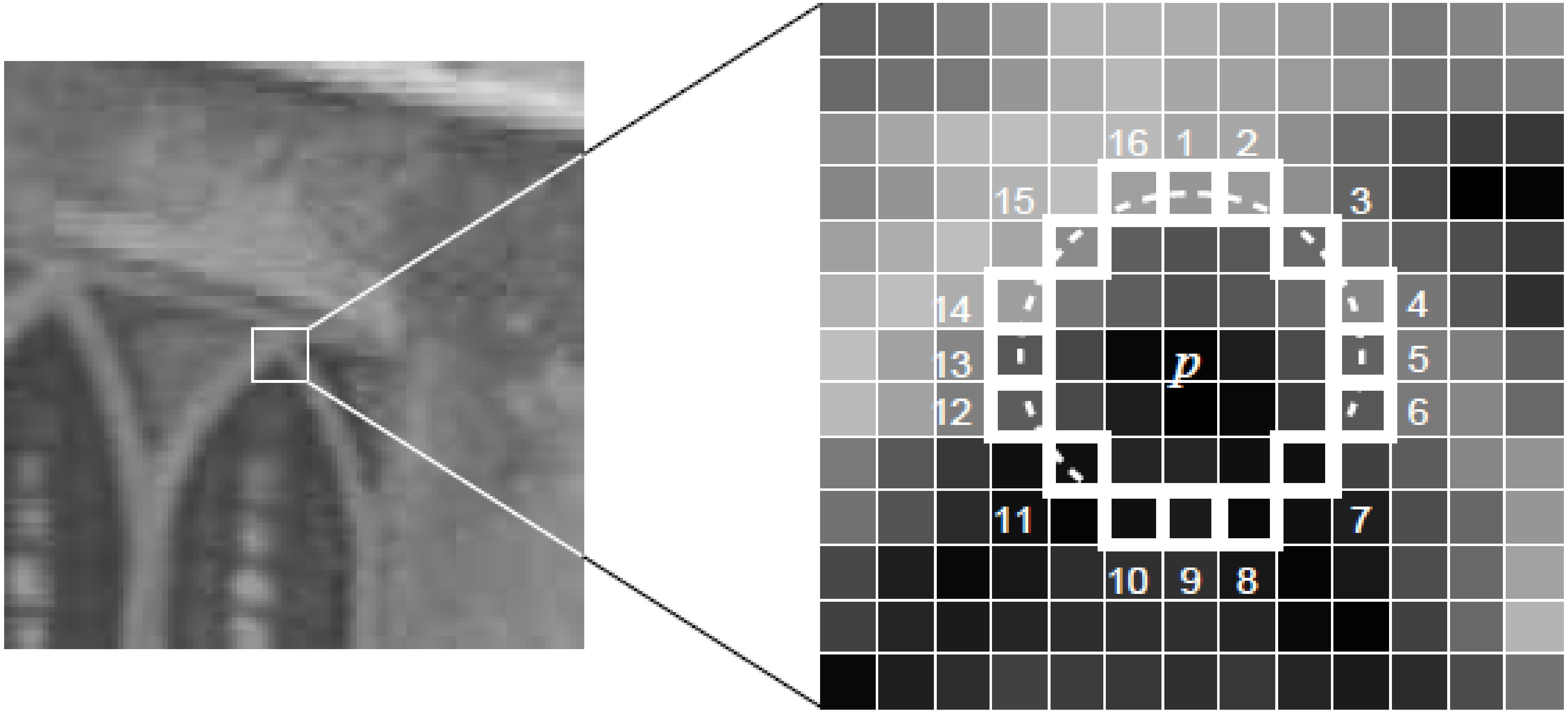
- It is efficient and finds reasonable corner keypoints, although it must be augmented by a Harris corner filter to reject edges and provide a reasonable score.

- Many keypoint detectors include an orientation operator (SIFT and SURF are two prominent examples), but FAST does not.

FAST: Features from Accelerated Segment Test

- A segment test criterion operates by considering a circle of Sixteen,(ORB uses 9) pixels around the corner candidate p
- The original detector classifies p as a corner if there exists a set of n contiguous pixels in the circle which are all brighter than the intensity of the candidate pixel I_k plus a threshold t, or all darker than $I_k - \mathrm{t}$
- Further Machine Learning Algorithm process to ensure higher speed for lower size of the circle

FAST



Harris Filter Augmentation

- For a target number N of keypoints, we first set the threshold low enough to get more than N keypoints, then order them according to the Harris measure, and pick the top N points
- FAST does not produce multi-scale features. ORB employs a scale pyramid of the image(8 scale), and produce FAST features (filtered by Harris) at each level in the pyramid

Reminder

- Compute spatial derivatives I_x, I_y
- Construct structure tensor M:

$$M = \sum \begin{bmatrix} {I_x}^2 & I_x I_y \\ I_y I_x & {I_y}^2 \end{bmatrix}$$

- Harris Response Calculation

$$\lambda_{min} = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} = \frac{Det(M)}{Trace(M)}$$

- Finding the local maxima within the 3 by 3 window

ORB Corner Orientation

- ORB uses a simple but effective measure of corner orientation, the **intensity centroid**
- Find moments of a patch:

$$m_{pq} = \sum_{x,y} x^p y^q \, I(x, y)$$

- Find centroid:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right)$$

- Construct a vector from the corner’s center, O, to the centroid, OC .
- Orientation of the Patch is:

$$\theta = atan2(m_{01}, m_{10})$$

ORB Descriptor

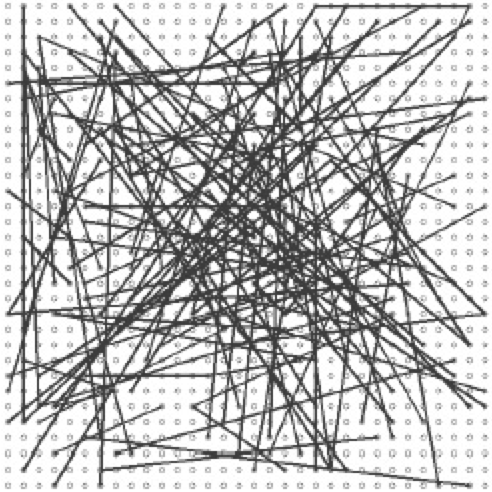
- ***BRIEF: Binary Robust Independent Elementary Feature***
- *Consider the binary test:*

$$t(p;x,y) = \begin{cases} 1 & : p(x) < p(y) \\ 0 & : p(x) > p(y) \end{cases}$$

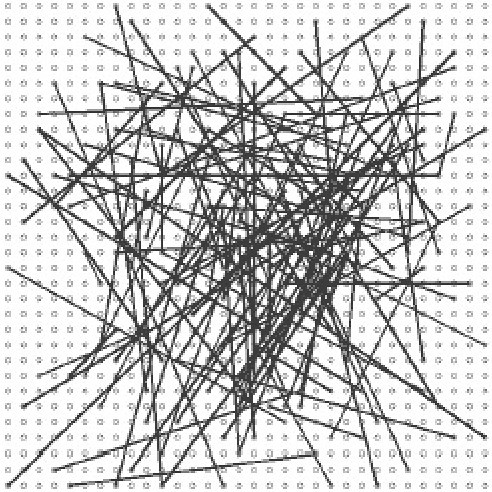
- *The feature is defined as a vector of n-binary test:*

$$f_n(p) = \sum_{i=1:n} 2^{i-1} t(p;x_i,y_i) \qquad \& \, n = 256$$

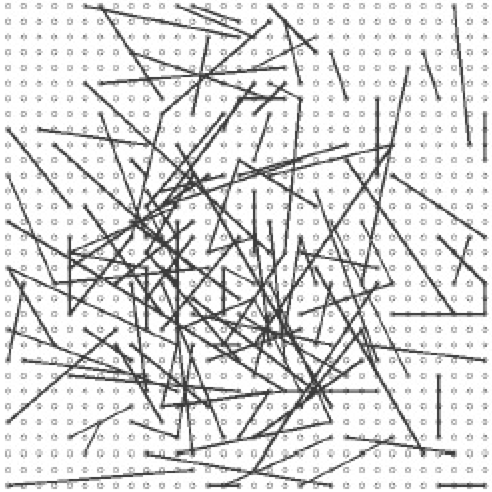
- ***rBRIEF:*** *To be invariant to in-plane rotation*
- $s = \begin{bmatrix} x_1, x_2 \ldots x_n \\ y_1, y_2 \ldots y_n \end{bmatrix} \rightarrow s_\theta = R_\theta s$
- $g_n(p,\theta) = f_n(p) | (x_i, y_i) \in s_\theta$
- ORB discretizes the angle to increments of 2π/30 (12 degrees), and construct a lookup table of precomputed BRIEF patterns. As long as the keypoint orientation θ is consistent across views, the correct set of points Sθ will be used to compute its descriptor.



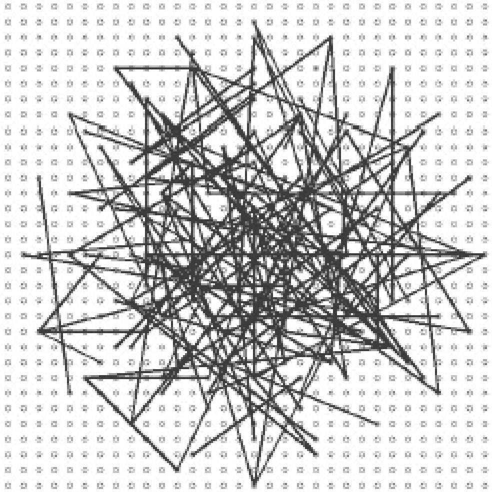
G I



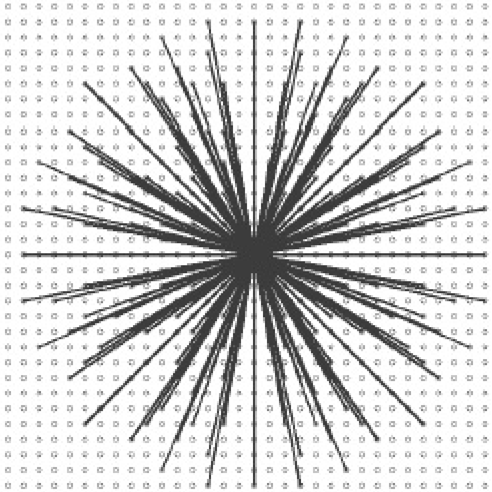
G II



G III



G IV



G V

Initialization

8-point in Each Iteration RANSAC

$$x_t^T F_{t,t-1} x_{t-1} = 0$$

Find Initial Correspondences

Parallel Computation of Homography and Fundamental Matrix

$$x_t = H_{t,t-1} x_{t-1}$$

Normalized DLT

Initialization

$$R_H = \frac{S_H}{S_H + S_F}$$

$$R_H > 0.45$$

If the Scene is Planar

Choose H

If not

Choose F

Structure From Motion Recovery

Full Bundle Adjustment

$$S_M = \sum_i \rho\left(d^2\left(x_i, H^{-1}x_i'\right)\right) + \rho\left(d^2\left(x_i', Hx_i\right)\right)$$

$$\rho(d^2) = \begin{cases} \Gamma - d^2 & d^2 < T_M \\ 0 & d^2 > T_M \end{cases}$$

ORB Tracking

- Tracking using motion-only Bundle adjustment as below, having all points in local area is fixed and only the camera pose is to be optimized

$$e_{i,j} = x_{i,j} - \pi_j(T_{iw}X_{wj})$$

$$\pi_j\left(T_{iw}X_{wj}\right) = \begin{bmatrix} f_{i,u} + \frac{x}{z}c_{i,u} \\ f_{i,u} + \frac{y}{z}c_{i,u} \end{bmatrix}$$

$$[x_{i,j} \quad y_{i,j} \quad z_{i,j}]^T = R_{iw}X_{w,j} + t_{iw}$$

$$Cost = \rho(e_{i,j}^T \Omega e_{i,j})$$

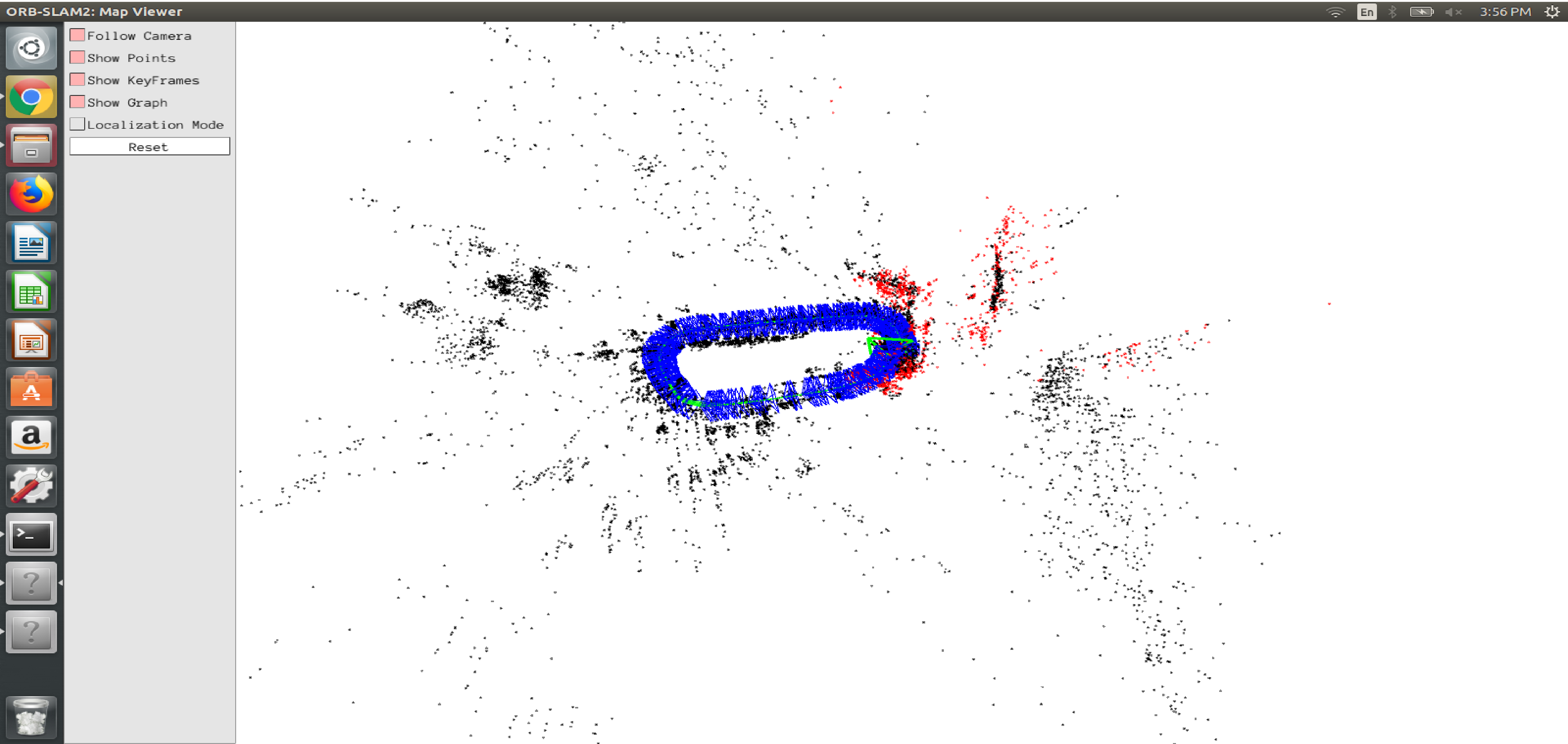
ORB Tracking

- If tracking was successful for last frame, the algorithm uses a constant velocity motion model to predict the camera pose and perform a guided search of the map points observed in the last frame.

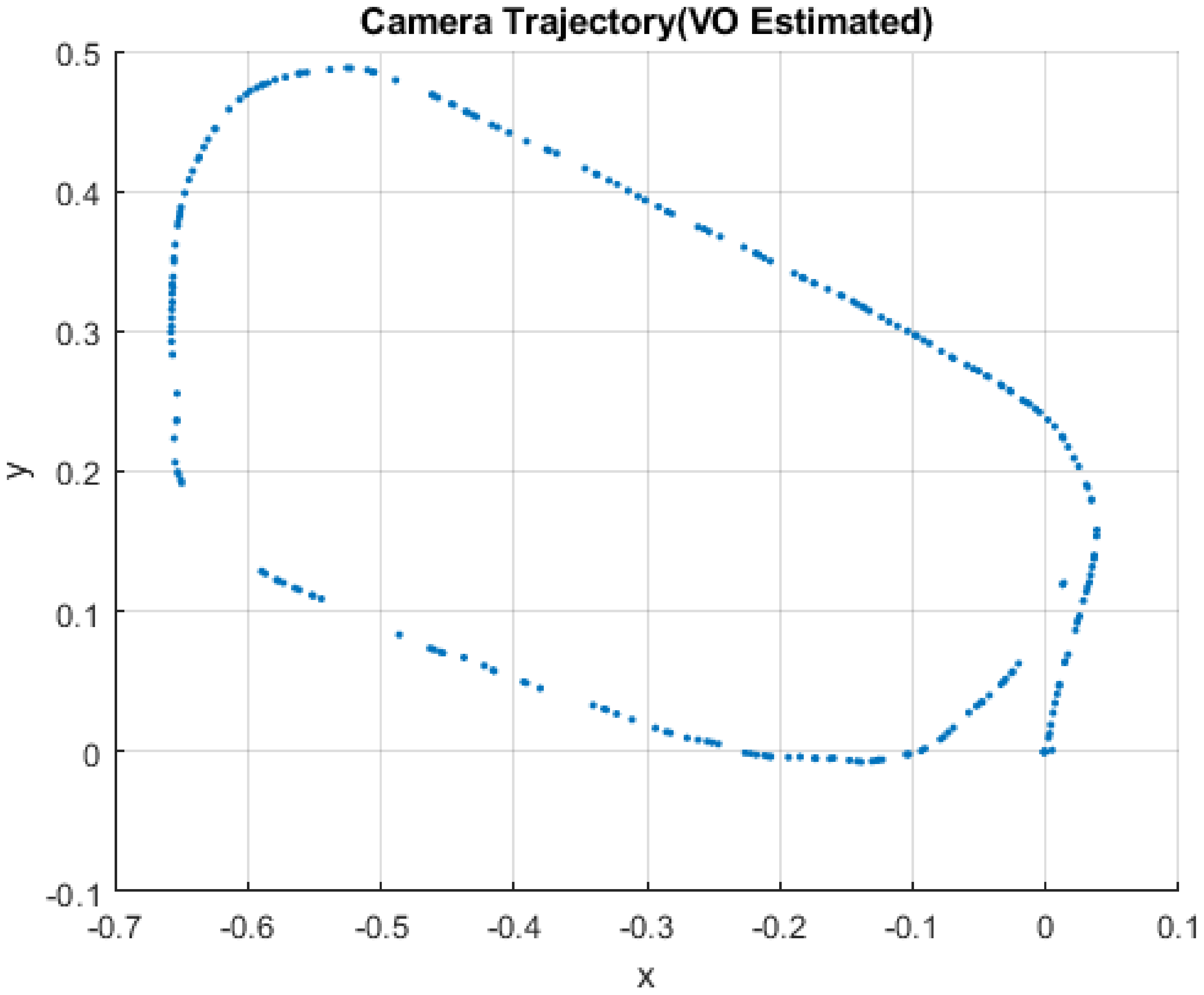
- If not enough matches were found, it uses a wider search of the map points around their position in the last frame. The pose is then optimized with the found correspondences

Results

Results


ORB-SLAM2: Map Viewer
En
3:56 PM
Follow Camera
Show Points
Show KeyFrames
Show Graph
Localization Mode
Reset

Results


Camera Trajectory(VO Estimated)
0.5
0.4
0.3
0.2
0.1
0
-0.1
y
-0.7
-0.6
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
x

References

- *[1] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, “ORB-SLAM: a ´ versatile and accurate monocular SLAM system,” IEEE Trans. Robot., vol. 31, no. 5, pp. 1147–1163, 2015.*
- *[2] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: an efficient alternative to SIFT or SURF,” in IEEE International Conference on Computer Vision (ICCV), Barcelona, Spain, November 2011, pp. 2564– 2571*