Auto-Calibration with Stop Signs

Yuhan Liu, Yunhai Han

Aug 13, 2020

Autonomous Vehicle Laboratory

Cognitive Robotics

Overview

calibrate intrinsics using estimated stop sign corners

- System Pipeline
- Line Refinement Threshold
- Kalman Filter

System Pipeline

The system pipeline can be illustrated as:

- Detection(bounding box for stop sign)
- Color Segmentation to find the edge of the sign
- Edge Detection to refine edges (early line detection)
- Line refinement with perpendicular search
- Estimation of intersection points for $N > 4$ points
- Homography estimation for calibration
- Update of camera calibration

Currentlt, all of this can be done automatically. In other words, the input file is the raw **rosbag** file and the output files are the calibration results. But, the manual work are needed in two parts:

System Pipeline

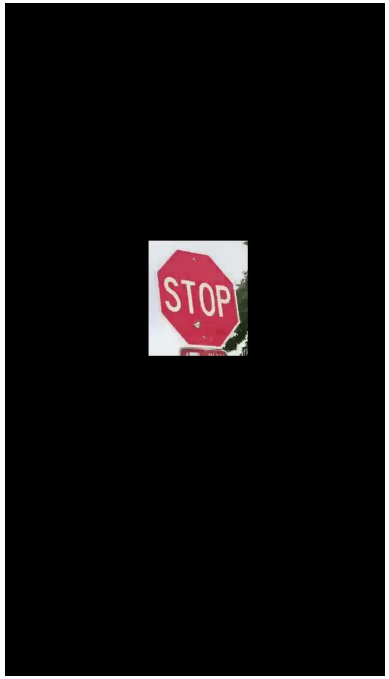
- Separate the whole video into several separate clips which capture stop signs.
- Manually remove some bad-quality images. For example, we obtain in total 21 sets of corner points(camera6), we have to manually remove one of them. The reason is that the estimated line does not match the real line well(due to some issues that Yuhan introduced before) and this would make the calibration result much worse.

We have also obtained the calibration results when the system is fully automatic(except for the part:separate the whole video into clips). The main point is to introduce a threshold to limit the modification along the normal line. Yuhan will talk a bit about this part.
However, the overall performance is a little worse than before(When more manual work is involved).

System Pipeline

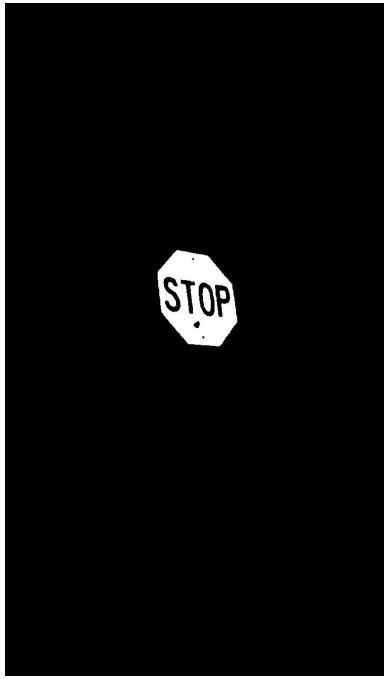
Video Frame

Current
- Mask RCNN*
(5.5 s/frame)



Bounding Box

Current
HSV



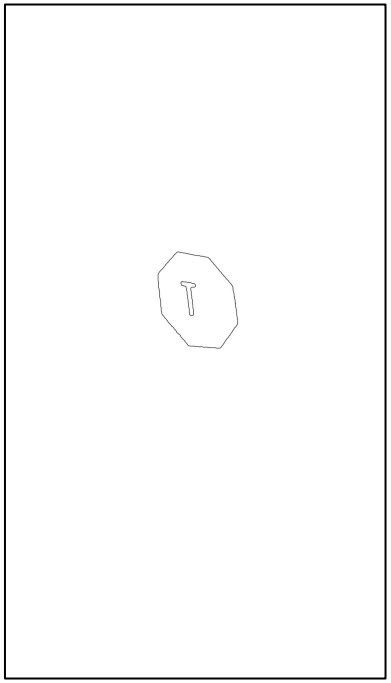
Pixel-wise Mask

*: Detectron2, Facebook AI Research

System Pipeline

Current
Canny Edge

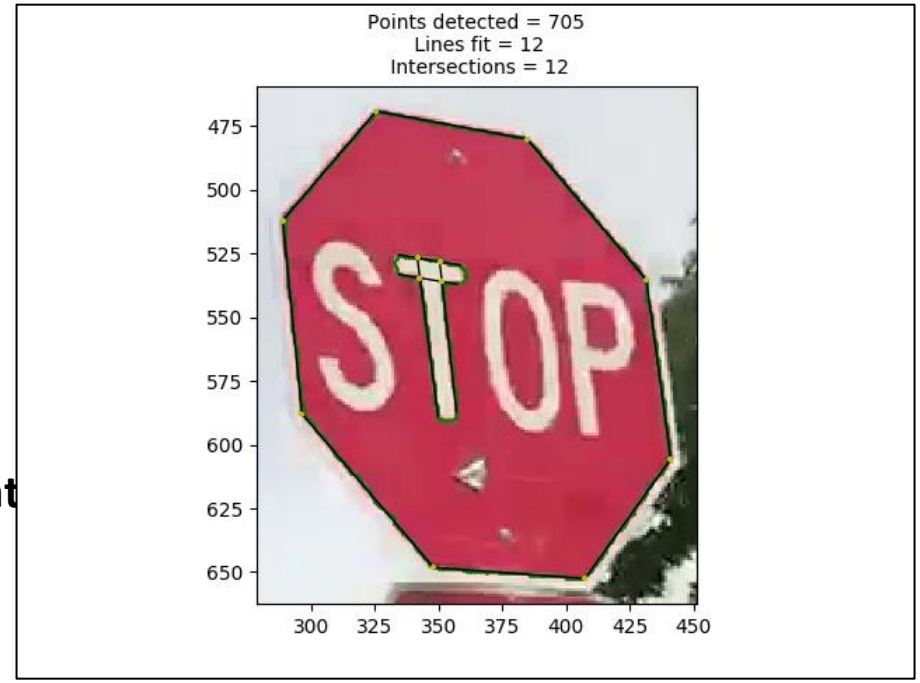
+Devernay Sub-Pixel Correction*



Edge Points

Current
SVD +RANSAC
(20 s/frame)

+Line Improvement with Threshold



Line Estimation + Intersections

*: R.G. Gioi & G. Randall(2017), "A Sub-Pixel Edge Detector: an Implementation of the Canny/Devernay Algorithm"

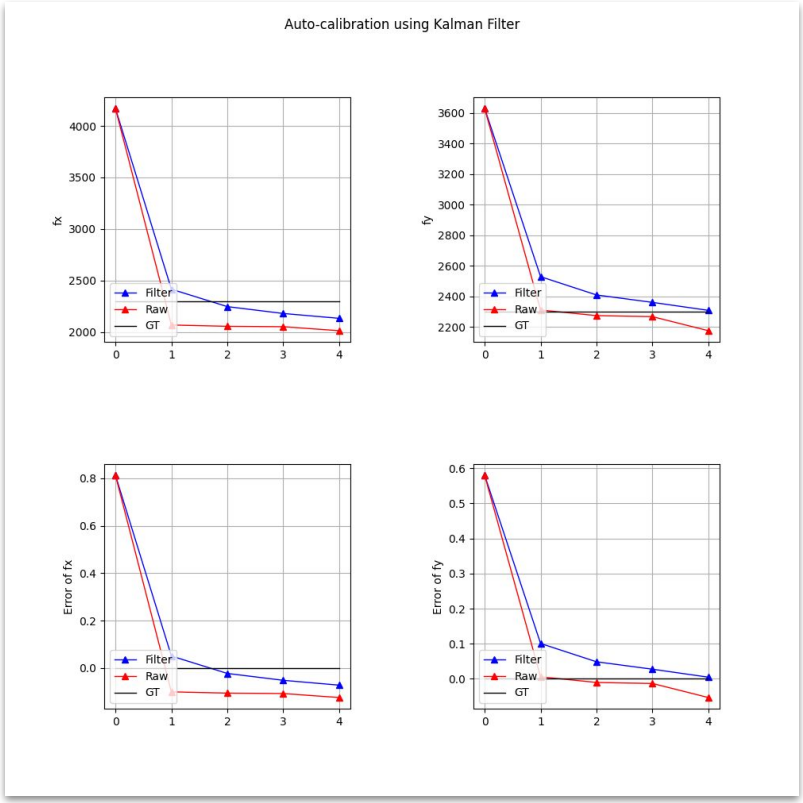
System Pipeline

Current
Zhang's Calibration*

$$\begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Intrinsic Matrix

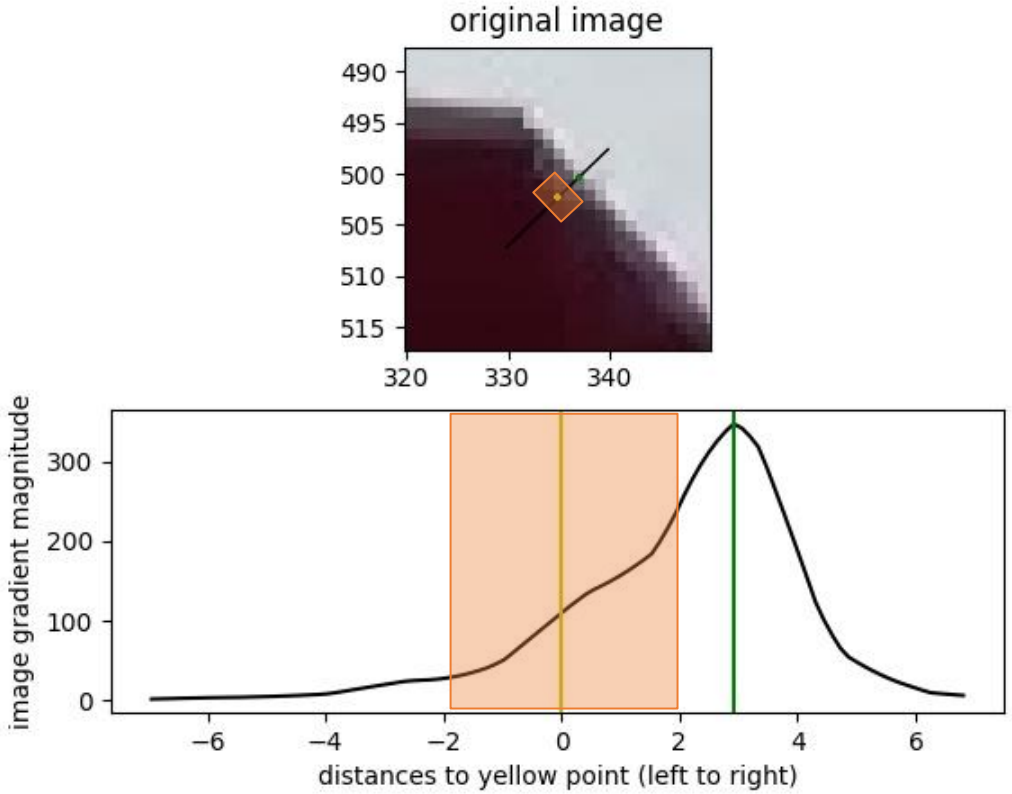
Current
Kalman Filter



Updated Calibration

*: Z. Zhang, "A flexible new technique for camera calibration"

Line Refinement Threshold



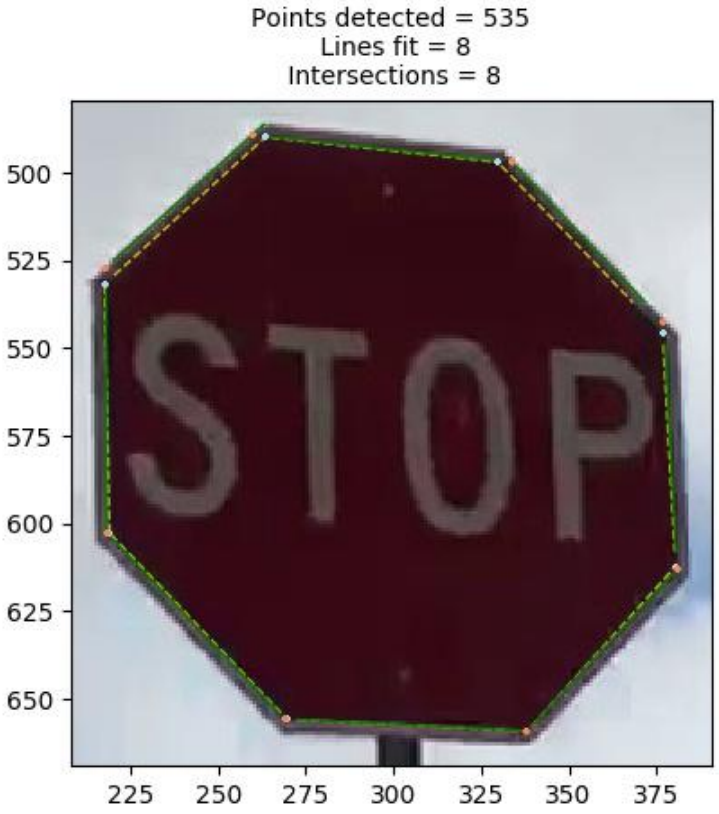
Estimated Line Segment Length

Estimated White Border Width

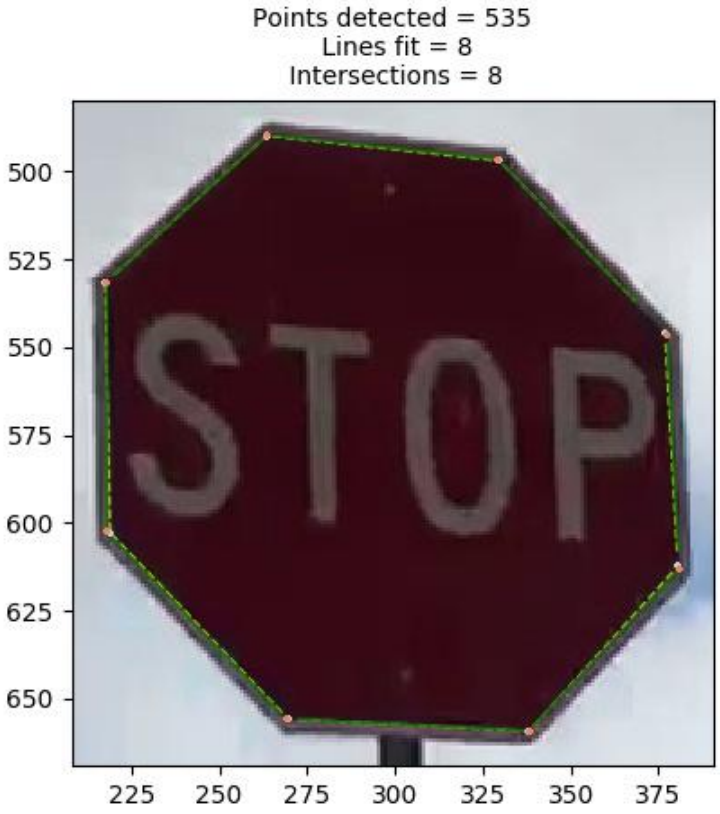
Threshold = 70% Border Width

Line Refinement - Line Threshold

More than Half of the Points beyond Threshold => Improved Line Abandoned



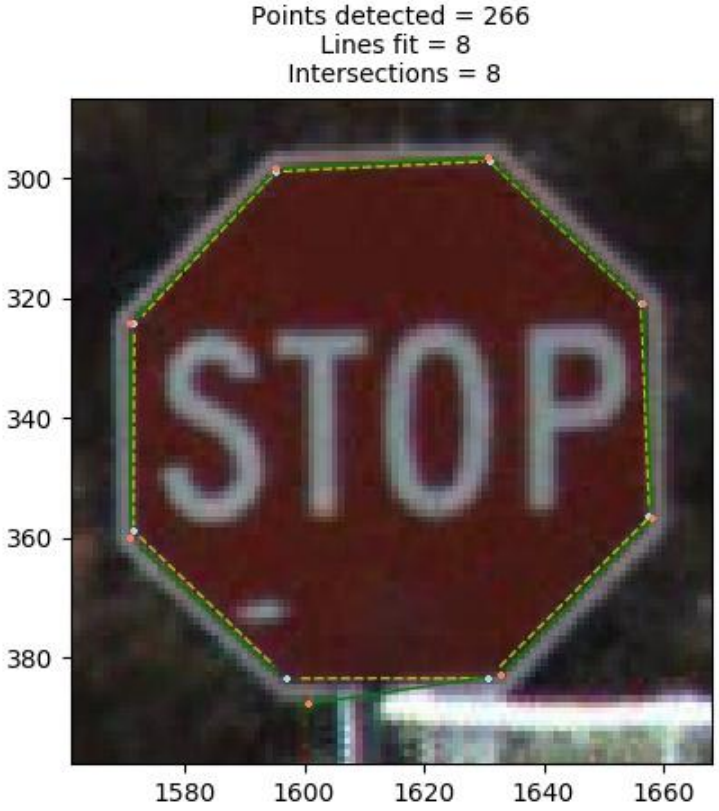
No Threshold



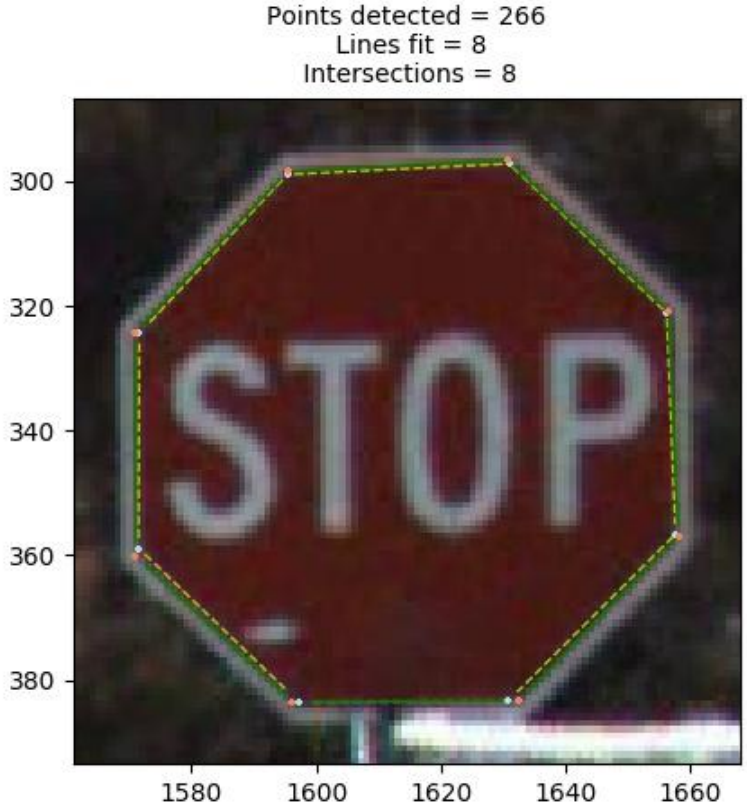
Line Thresholded

Line Refinement - Line Threshold

More than Half of the Points beyond Threshold => Improved Line Abandoned



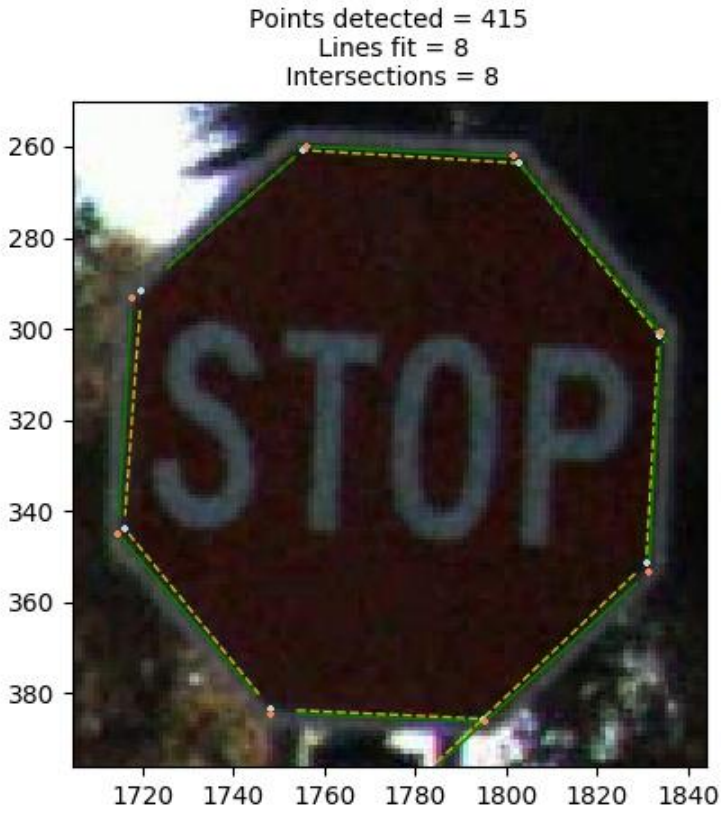
No Threshold



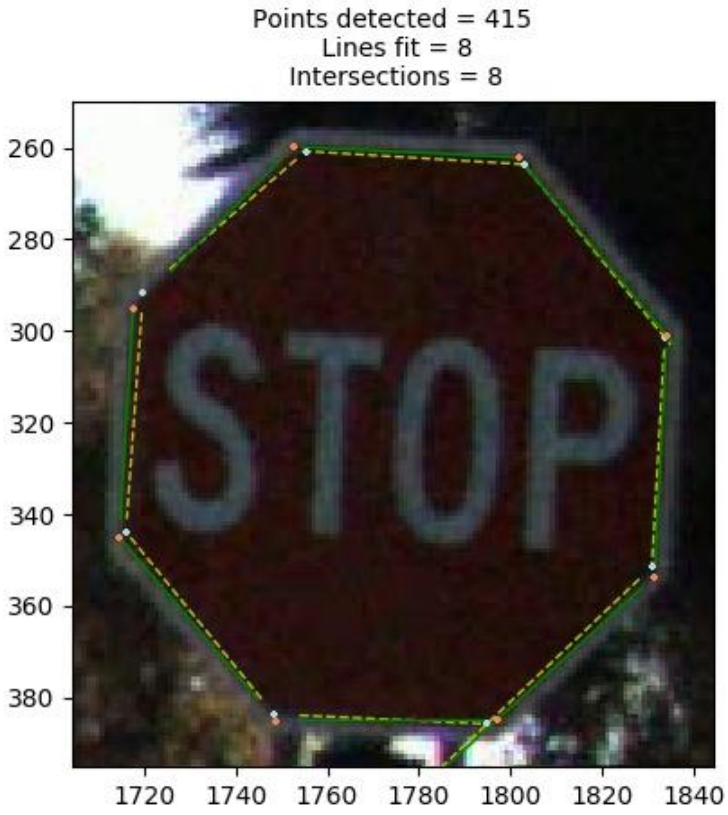
Line Thresholded

Line Refinement - Point Threshold

Point beyond Threshold => Improved Point Abandoned



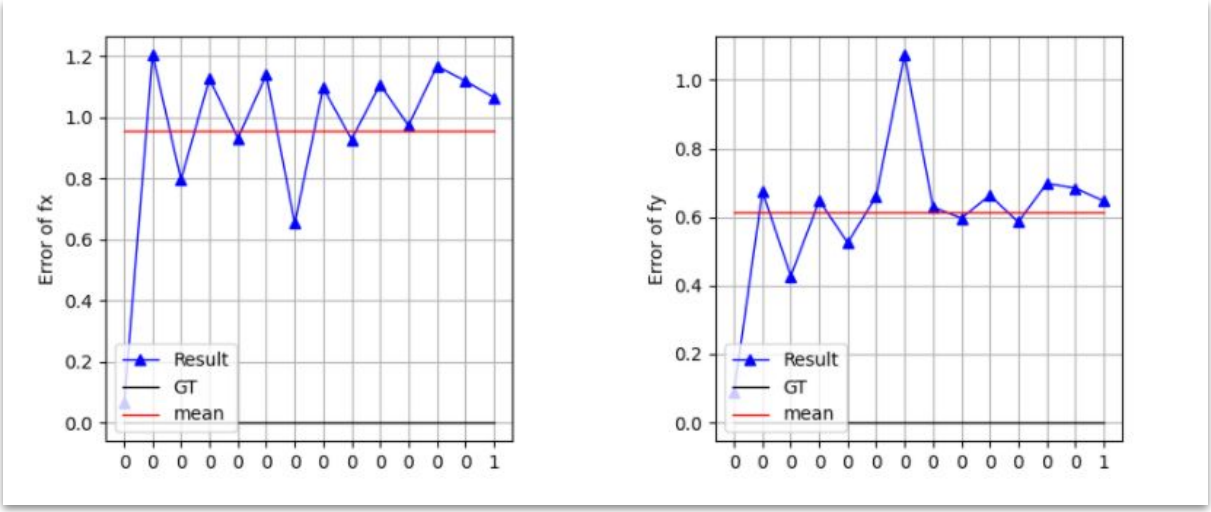
Line Threshold



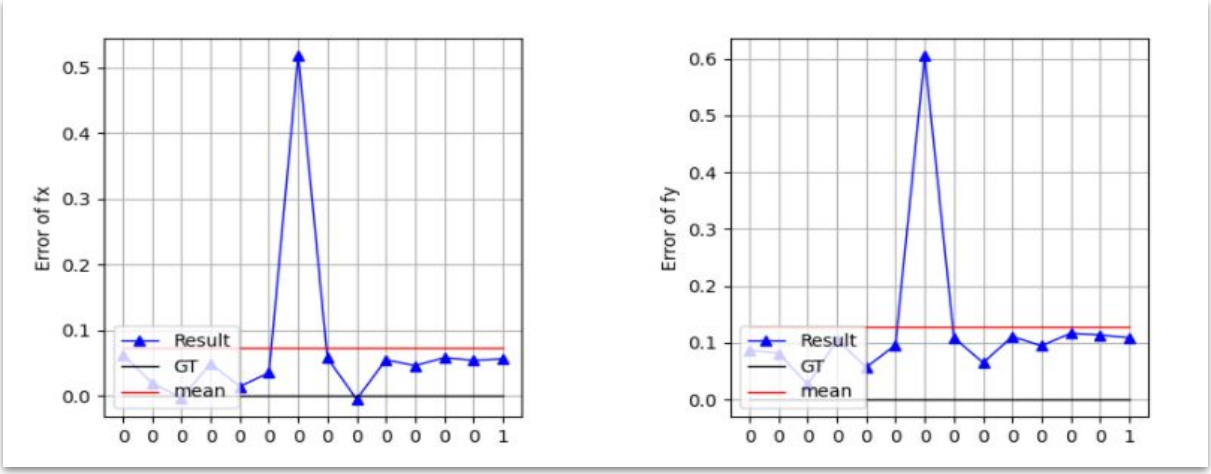
Point Thresholded

Performance

No Threshold



Line Threshold



Kalman Filter

We implement Kalman Filter as professor mentioned in the report. Since both the state transition equation and measurement equation are linear and no control inputs are involved, it is very easy for implementation.

We think of three different modes on how to implement Kalman filter on the images we collected.

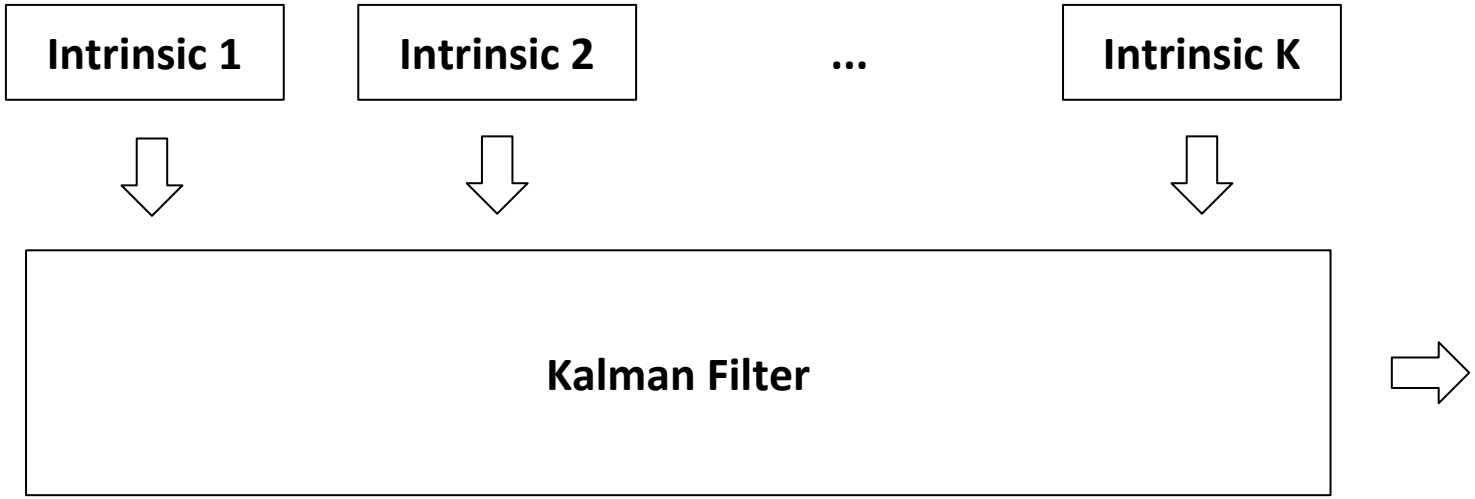
Suppose we have in total N images.

- Mode1: Each time we select $N-1$ images and calibration the camera. Run Kalman filter on N sets of results.
- Mode2: Set a moving window and its length is $K(K < N)$. We gradually select K images in the temporal order. Run Kalman filter on $N - K + 1$ sets of results.
- Mode3: Gradually increse the number of images used for calibration. at the first iteration, only 2 images are used(minimum number of imgaes that are required) and at the last iteration, all N images are used.

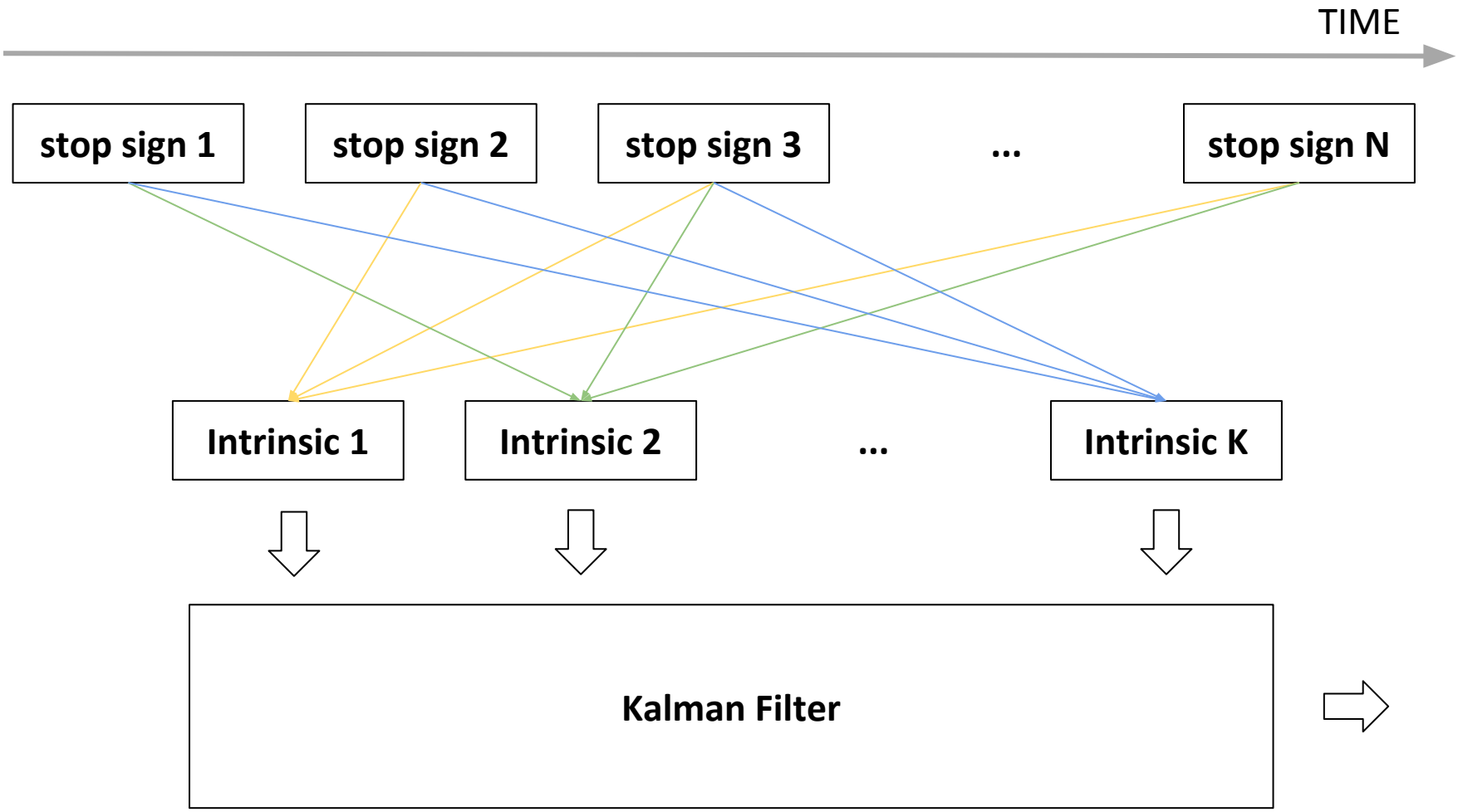
We do experiments for camera1 and camera6 and this are the results for the three modes(the noise parameters are the same):

Kalman Filter

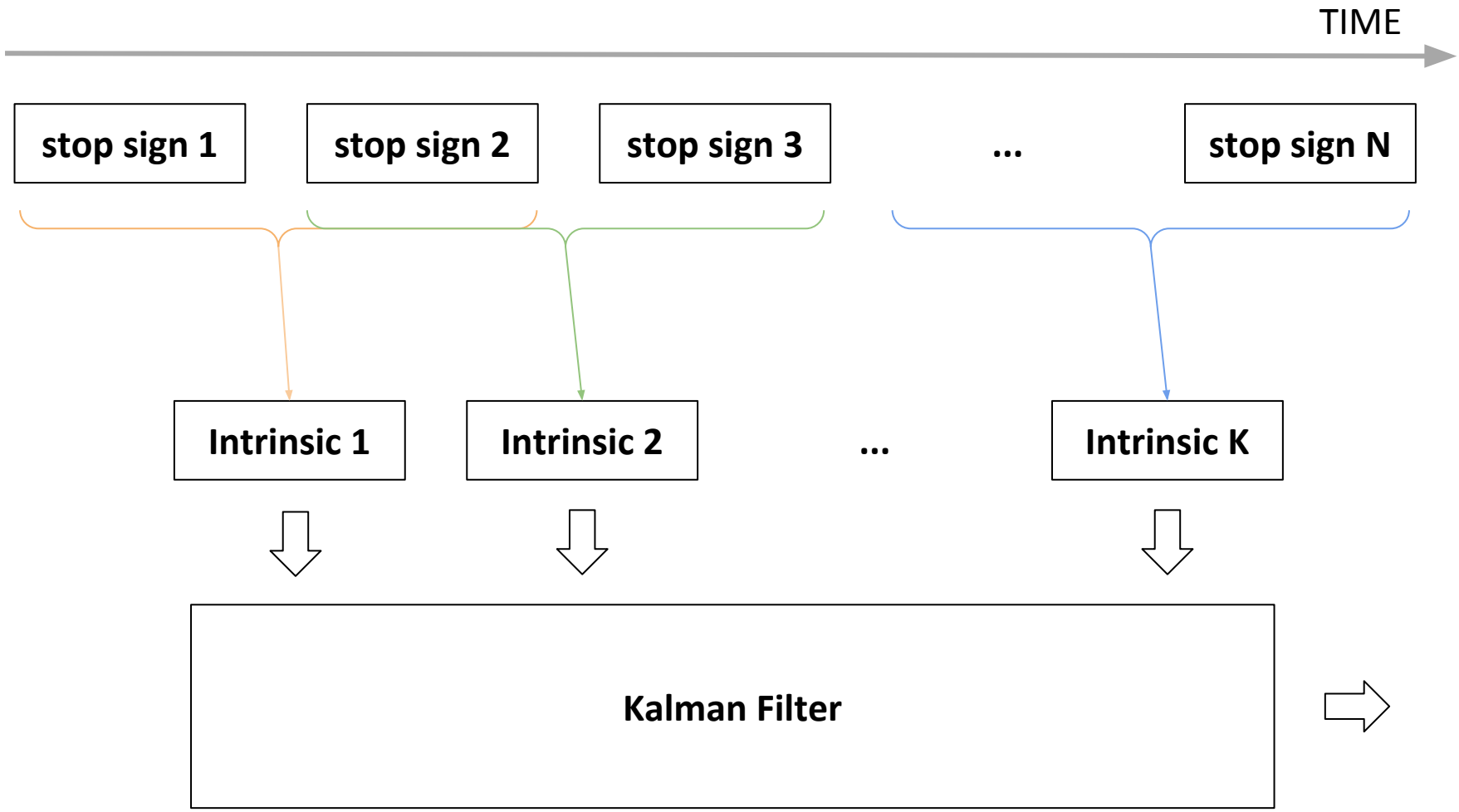
How to calculate these intrinsic matrices?



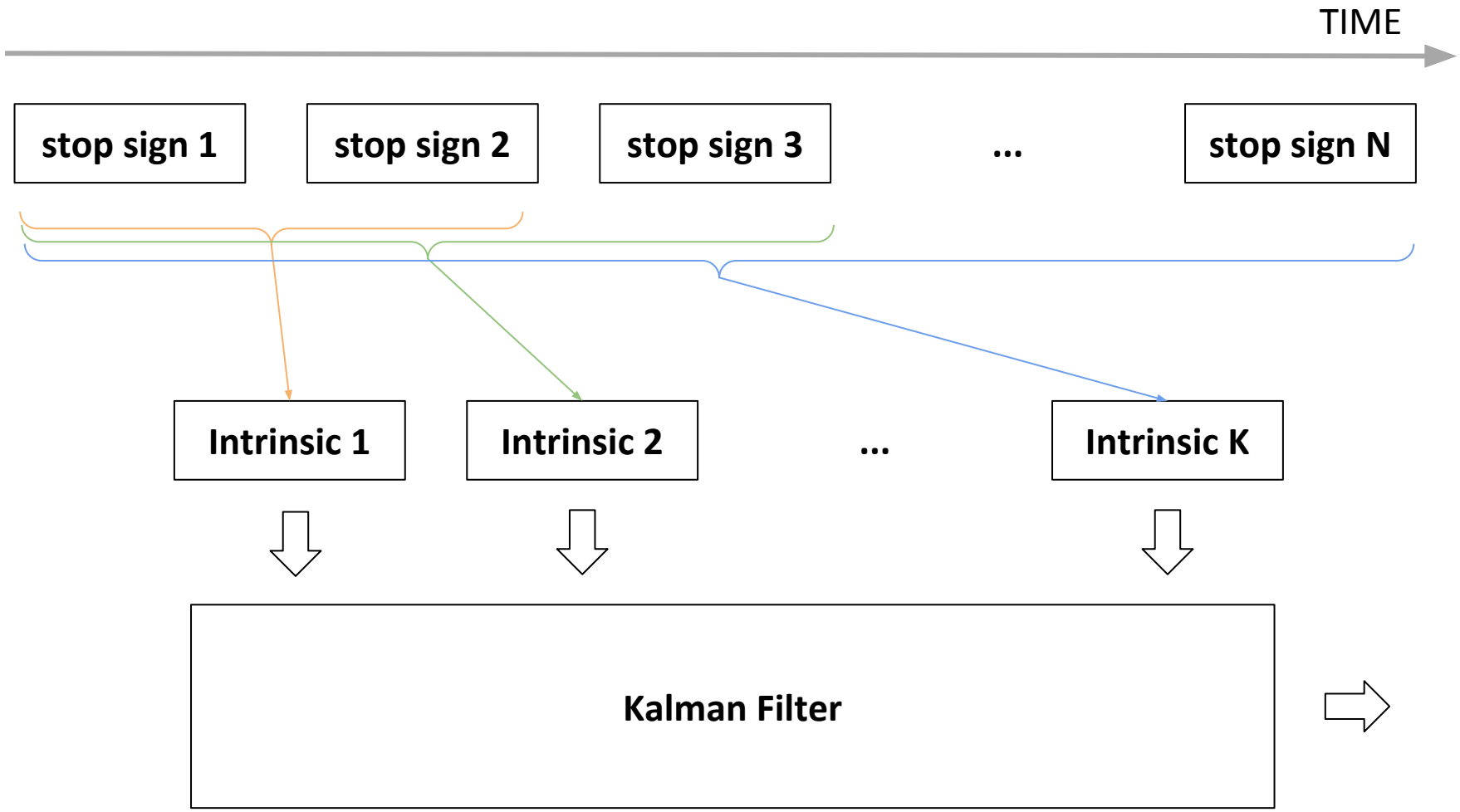
Kalman Filter - Smoothing Mode



Kalman Filter - Sliding Window Mode



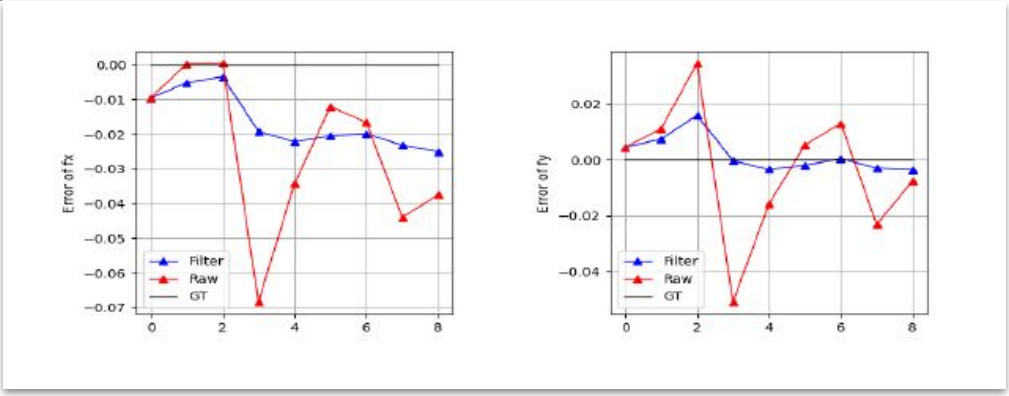
Kalman Filter - Accumulation Mode



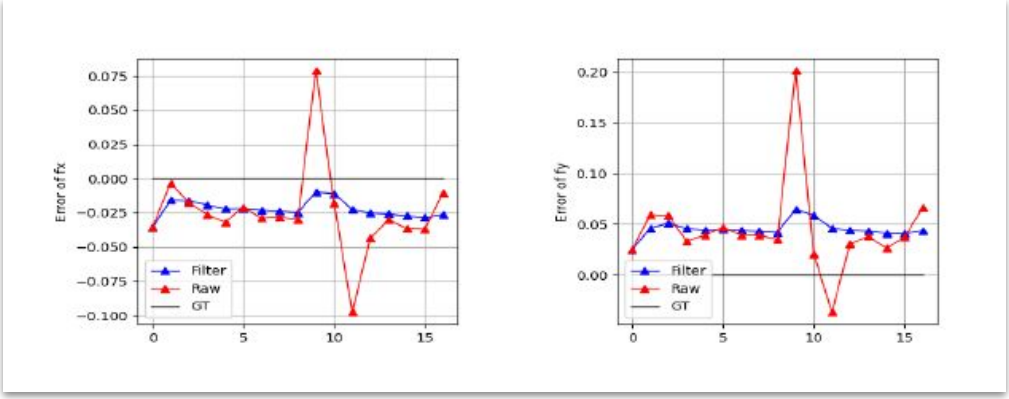
Kalman Filter - Results

Smoothing Mode

camera1:



camera6:

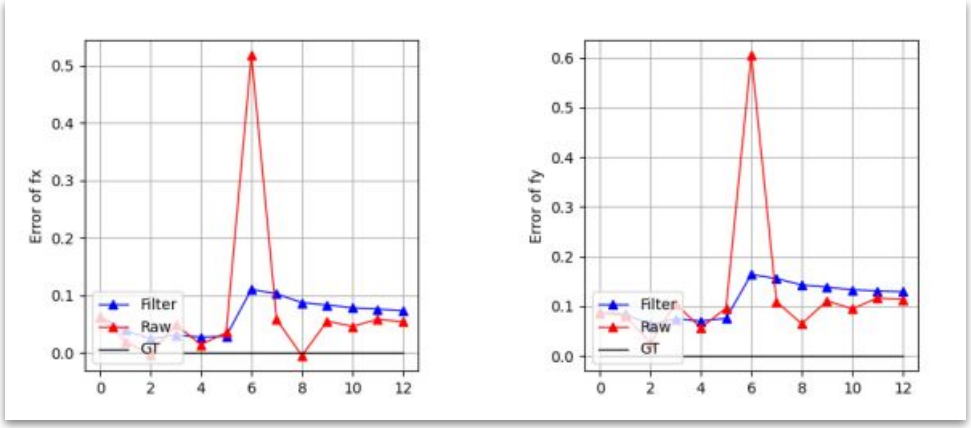


Because each time we use $N-1$ images, the calibration results themselves are the most accurate, there is no big difference after implementing Kalman Filter.

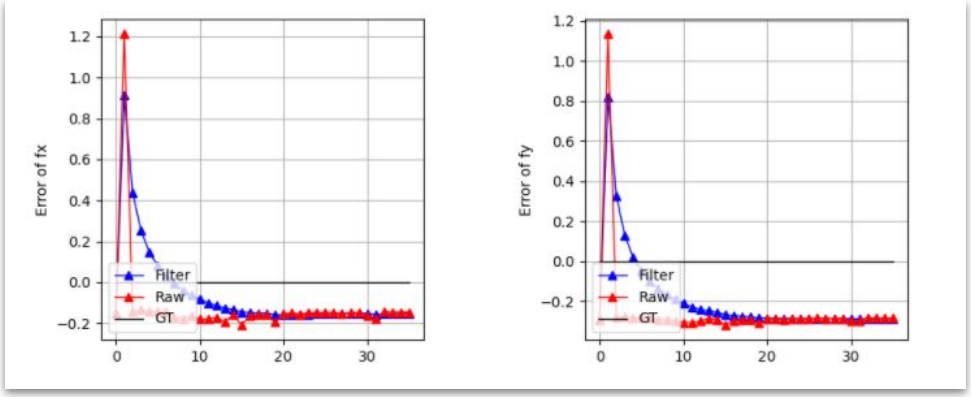
Kalman Filter - Results *Automatic*

Smoothing Mode

camera1:



camera6:

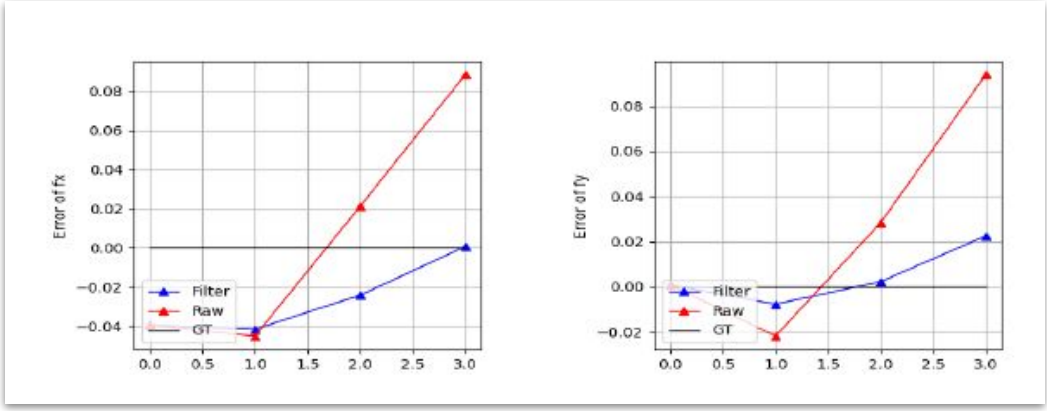


These two figures show the results that are obtained automatically.
The average relative errors are larger than before.

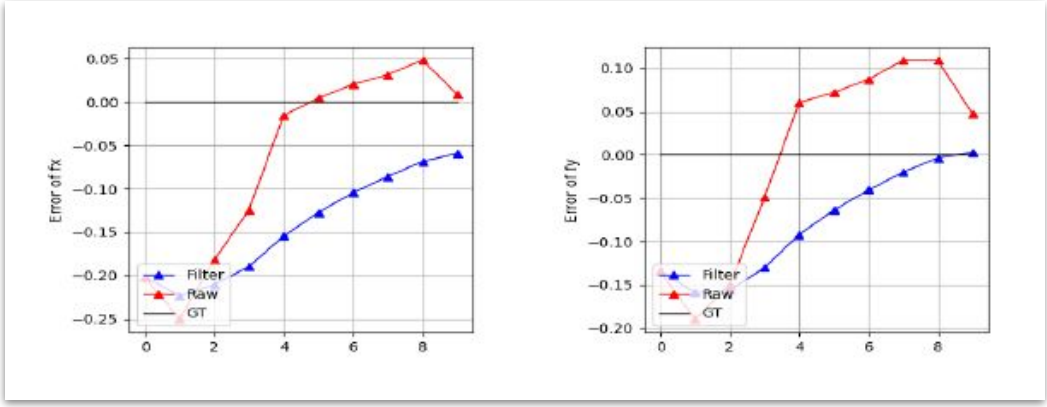
Kalman Filter - Results

Sliding Window Mode

camera1($K = 6$):



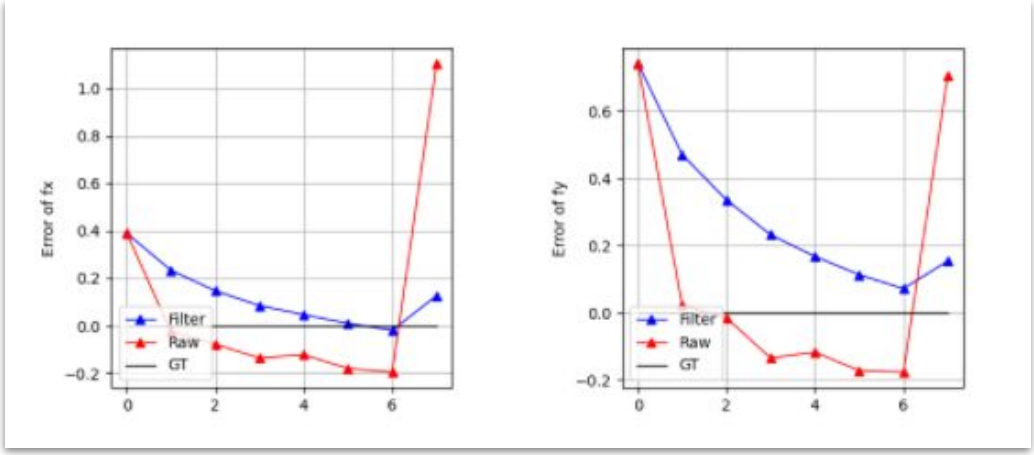
camera6($K = 8$):



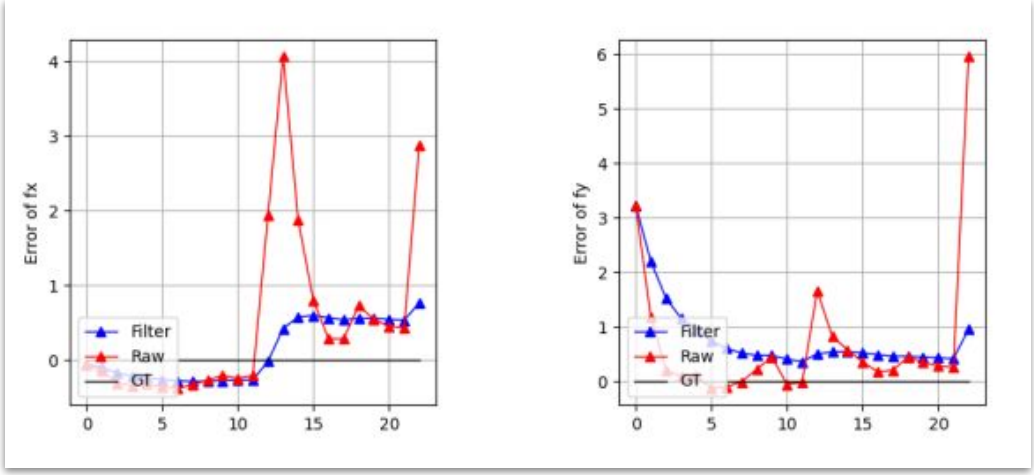
Kalman Filter - Results *Automatic*

Sliding Window Mode

$\text{camera1}(K = 6)$:



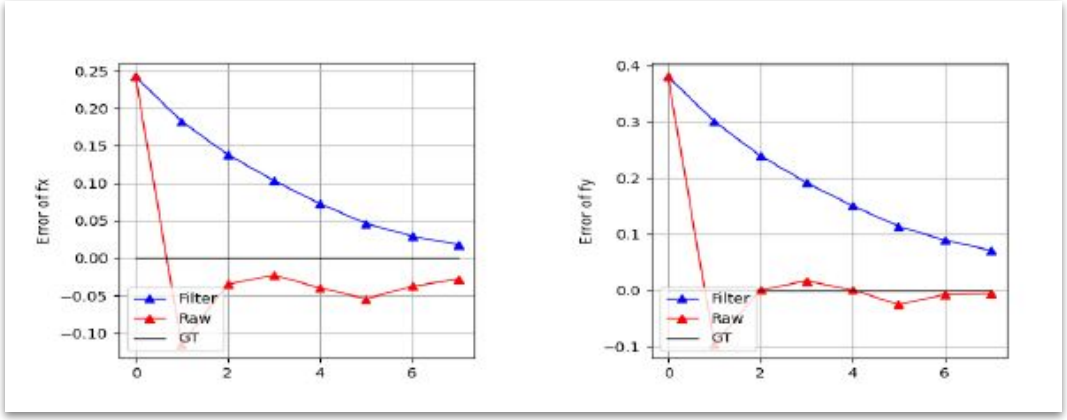
$\text{camera 6}(K = 10)$:



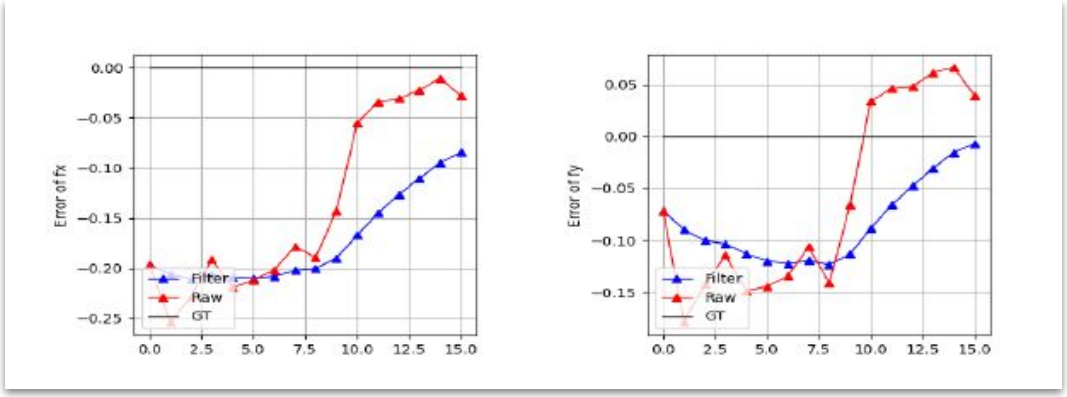
Kalman Filter - Results

Accumulation Mode

camera1:



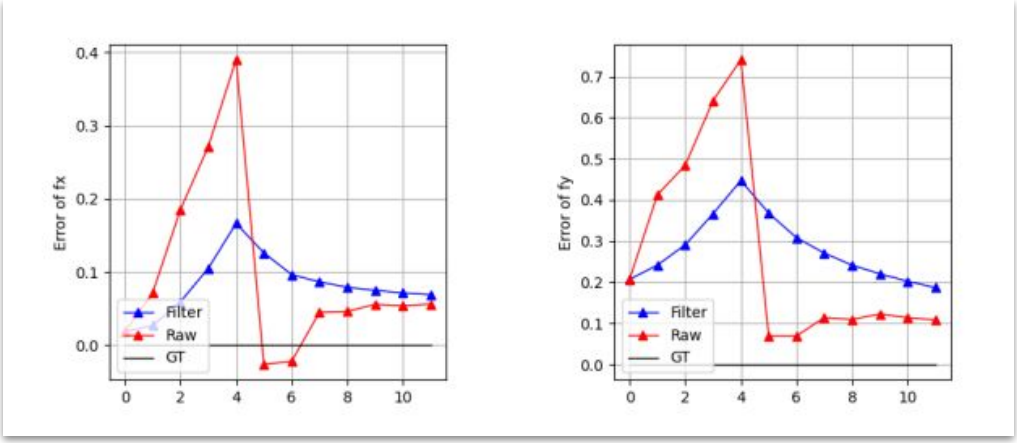
camera6:



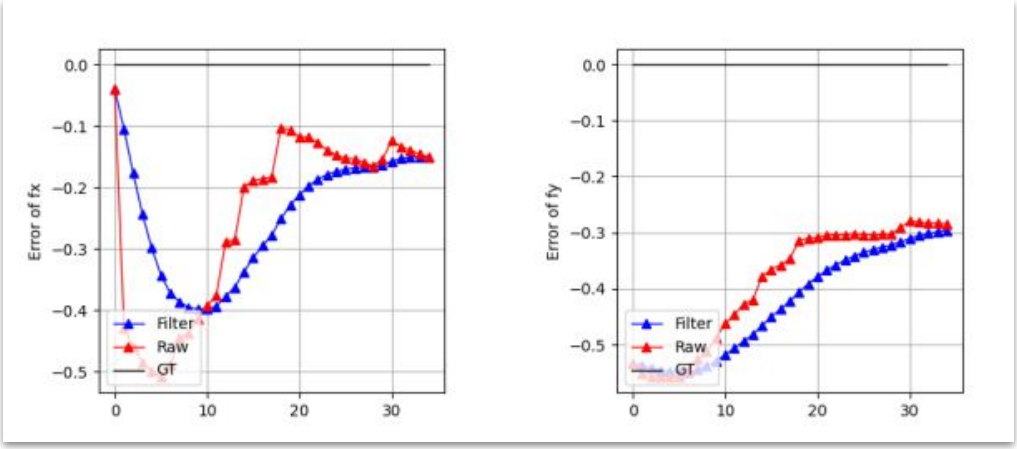
Kalman Filter - Results *Automatic*

Accumulation Mode

camera1:



camera6:



Thank you!