Auto-Calibration with Stop Signs

Improvement Methods

Yunhai Han
Department of Mechanical and Aerospace Engineering
y8han@eng.ucsd.edu
Yuhan Liu
Department of Computer Science and Engineering
yul139@eng.ucsd.edu

August 1, 2020

1 Introduction

The chapters in this report can be split into two parts:

1. Experiment results on AVL dataset
2. Two improvement methods

As discussed before, there are two possible improvement methods.

1. For each frame:1D correction along the normal line.
2. For video frames:combine separate information.

Currently, we have finished the coding and debugging for method 1 with one unsolved problem and built the model for method 2.

2 Experiment Results

The following two figures show the calibration results for each camera before the improvement.

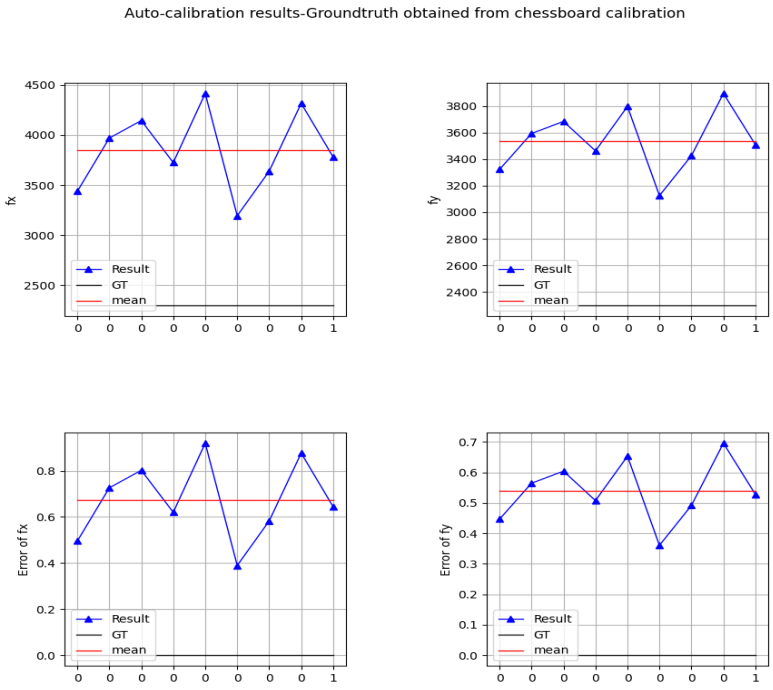


Figure 1: Calibration result for camera1

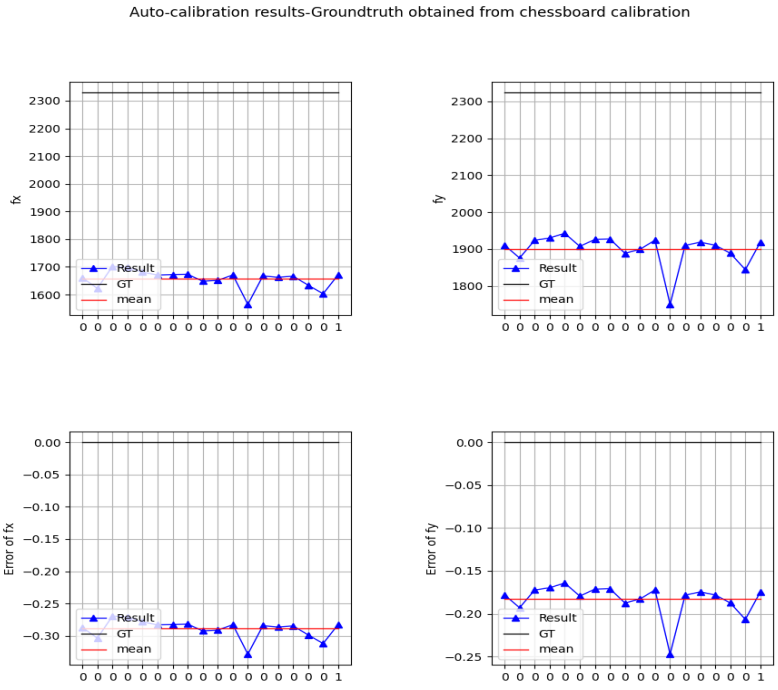


Figure 2: Calibration result for camera6

Duo to the image qualities, the average relative errors of each camera are different.

Next, we will show that after we implement method 1(Line Function Fine-tuning), for each camera, the calibration results are better than previous ones.

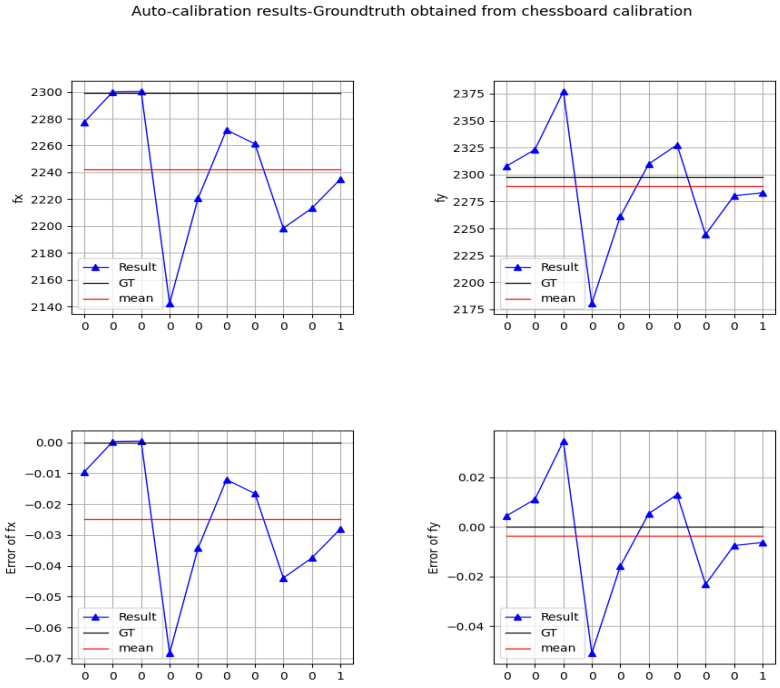


Figure 3: Calibration result for camera1

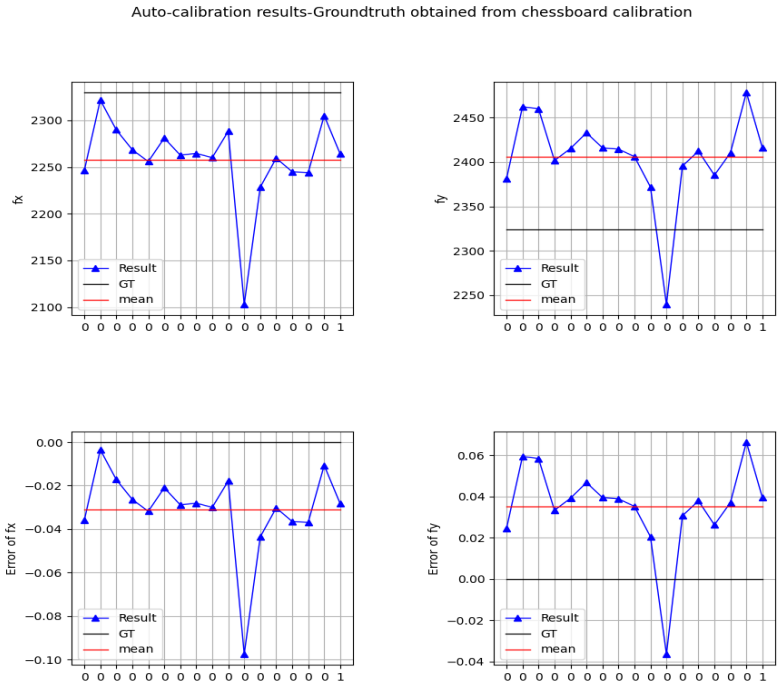


Figure 4: Calibration result for camera6

Here, we give a brief explanation of x-axis.

Assume we have in total N images, every time we use $N-1$ images to calibrate our cameras and these N sets of calibration results are marked by ”0” on the x-axis. At last, we use all of them for calibration and they are marked by ”1” on the x-axis.

Table 1: Camera1:Last set(use all images)

Relative error	Before improvement	After improvement
f_x	64.37%	-2.81%
f_y	52.76%	-0.63%

Table 2: Camera6:Last set(use all images)

Relative error	Before improvement	After improvement
f_x	-28.25%	-2.82%
f_y	-17.44%	3.94%

The effectiveness of the improvement method is apparent.

3 Method 1 - Line Function Fine-tuning

3.1 Problem Formulation

Given one image containing stop signs

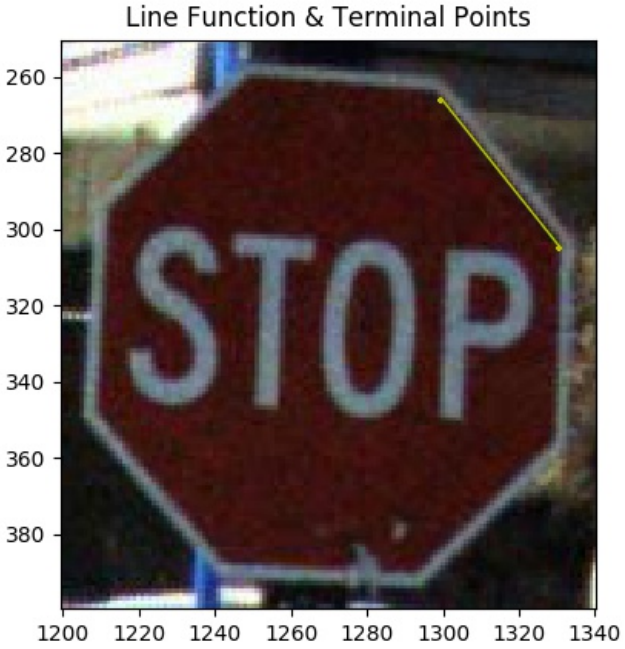
$$\{I(u, v)\},$$

as well as estimated line functions

$$\{(a_i, b_i, c_i)|a_i x + b_i y + c_i = 0, (x, y) \in E_i\}$$

and terminal points

$$\{(x_i^{start}, y_i^{start}), (x_i^{end}, y_i^{end})\}$$



of stop sign edges $\{E_i\}$ (e.g. inner polygon or character "T"), the goal is to find a new line function

$$\{(a_i', b_i', c_i')\}$$

and a new pair of terminal points

$$\{(x_i^{start'}, y_i^{start'}), (x_i^{end'}, y_i^{end'})\}$$

such that the number of edge points in the new line is more than that in the given line:

$$|\{(x,y)|a_i'x + b_i'y + c_i' = 0, x_i^{start'} \leq x \leq x_i^{end'}, y_i^{start'} \leq y \leq y_i^{end'}, (x,y) \in E_i\}|$$

$$>$$

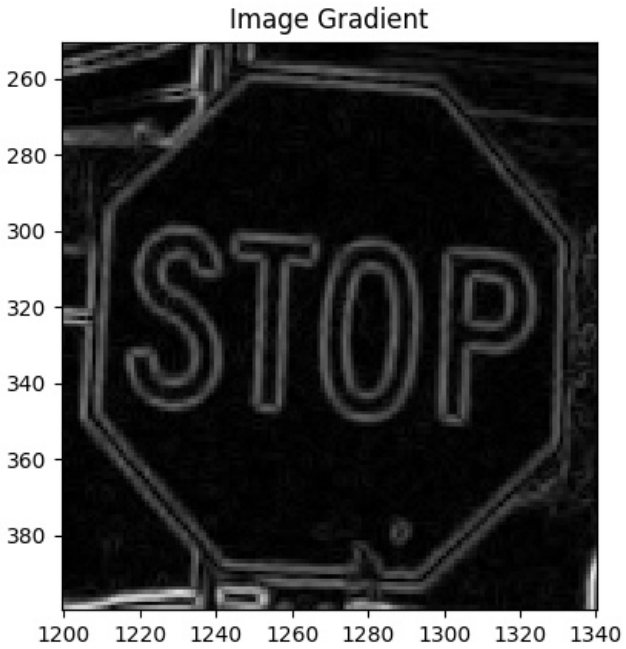
$$|\{(x,y)|a_ix + b_iy + c_i = 0, x_i^{start} \leq x \leq x_i^{end}, y_i^{start} \leq y \leq y_i^{end}, (x,y) \in E_i\}|.$$

3.2 Algorithm

The idea is to take the image gradient around the line (more precisely, in the normal direction) as the guidance of generating a new line.

0. Calculate image gradient magnitude $G(u,v)$ of the image $I(u,v)$:

$$G(u,v) = \sqrt{\left(\frac{I(u+\Delta u, v) - I(u,v)}{\Delta u}\right)^2 + \left(\frac{I(u, v+\Delta v) - I(u,v)}{\Delta v}\right)^2};$$



Apply bilinear interpolation to make it continuous:

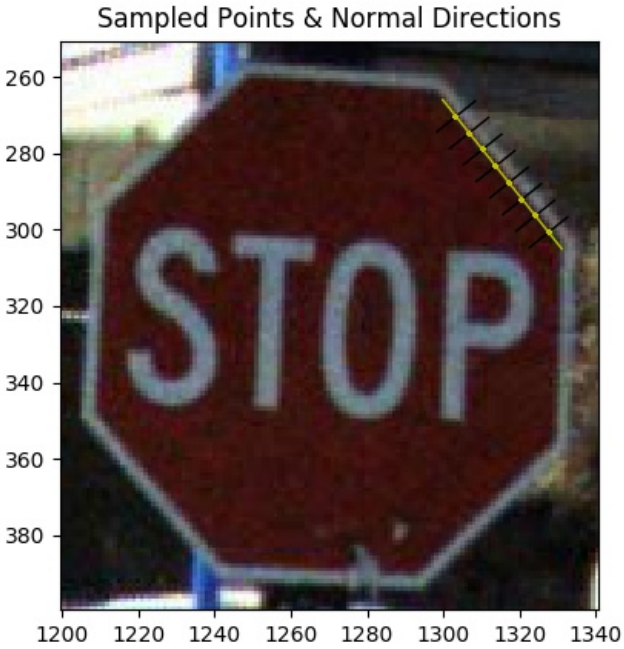
$$G(x,y), \forall (x,y) \in \mathbb{R}^2;$$

1. Sample K equally separated points from line segment $\{(a,b,c), (x^{start}, y^{start}), (x^{end}, y^{end})\}$:

$$\{(x_k, y_k)|x_k = \min(x^{start}, x^{end}) + k\frac{|x^{start} - x^{end}|}{K+1}, y_k = \frac{-ax_k - c}{b}, k \in \{1, ..., K\}\};$$

2. Calculate normal lines passing through each sampled points:

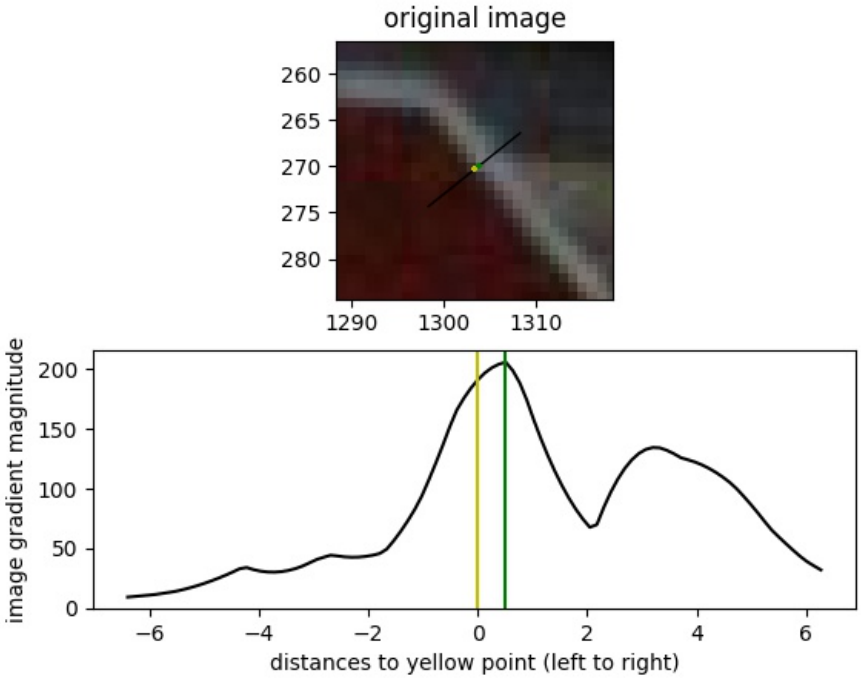
$$\{(a_k, b_k, c_k)|a_k = -b, b_k = a, c_k = ay_k - bx_k\};$$



3. Calculate image gradients along the normal lines around the sampled points:

$$\{G_k(x_i, y_i)|x_k - i \times r_x \leq x_i \leq x_k + i \times r_x, y_i = \frac{-a_k x_i - c_k}{b_k}\},$$

where r_x is the sampling resolution for x-coordinate;



4. Find the locally maximal gradient $G(x^k, y^k), (x^k, y^k) \in \{(x_i, y_i)\}$ along each normal line that is closest to the sampled point (x_k, y_k); Now, each sampled point (x_k, y_k) has a corresponding "improved" new point in the normal direction (a_k, b_k, c_k):

$$\{(x^k, y^k)|(x_k, y_k)\};$$

5. Apply SVD (and RANSAC) to above K points $\{(x^k, y^k)\}$ for a "improved" line function (a', b', c') and terminal pairs $\{(x^{start'}, y^{start'}), (x^{end'}, y^{end'})\}$.

3.3 Results

The results show that lines are adjusted to align with the edges tightly, leading to more accurate estimation of corner points.

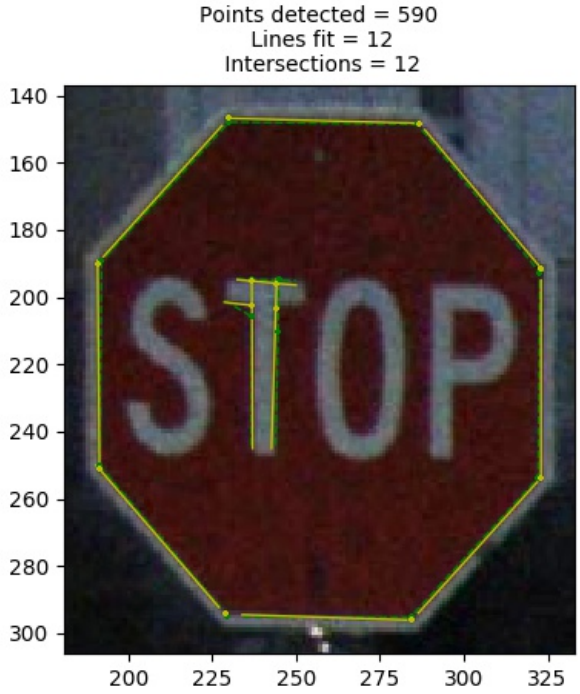


Figure 5: Example from AVL Dataset (green dotted lines & green crosses: before improvement; yellow solid lines & yellow crosses: after improvement)

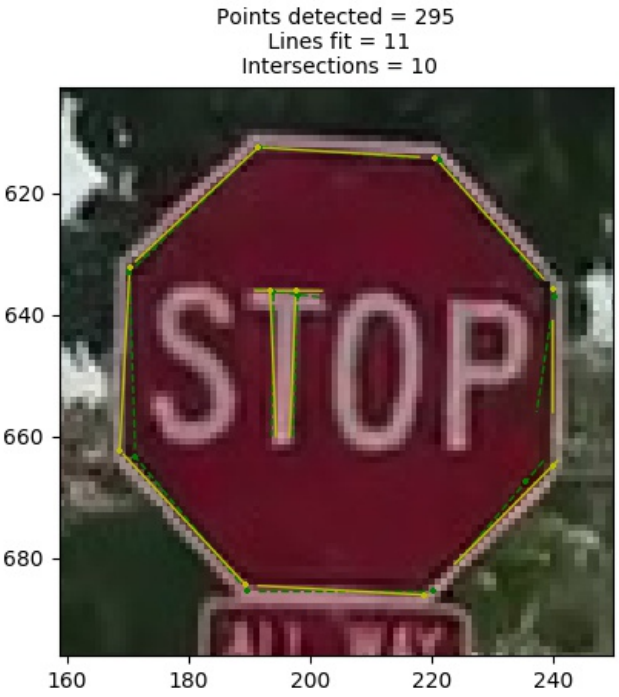


Figure 6: Example from iPhone Dataset (green dotted lines & green crosses: before improvement; yellow solid lines & yellow crosses: after improvement)

3.4 Issues

In most cases, the above algorithm fine-tunes the lines closer to the edges it corresponds to. Yet, it can be mis-guided by the abnormal image gradients, leading to, for example, a line estimation of an edge from the inner polygon mis-aligned with the outer polygon, introducing greater errors to the corner point estimation.

Looking into the gradients in the normal direction of the original estimated line, it is patent that the over-exposure of the background blurs the boundary of the red and white area in the stop sign, eliminating the local gradient peek of the edge in the inner polygon.

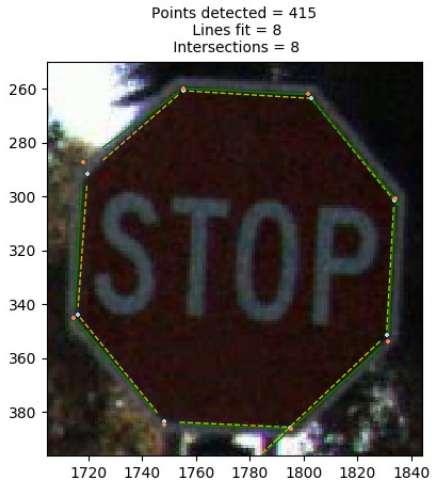


Figure 7: Upper-Left Edge is Over-adjusted (yellow dotted lines & light-blue crosses: before improvement; green solid lines & orange crosses: after improvement)

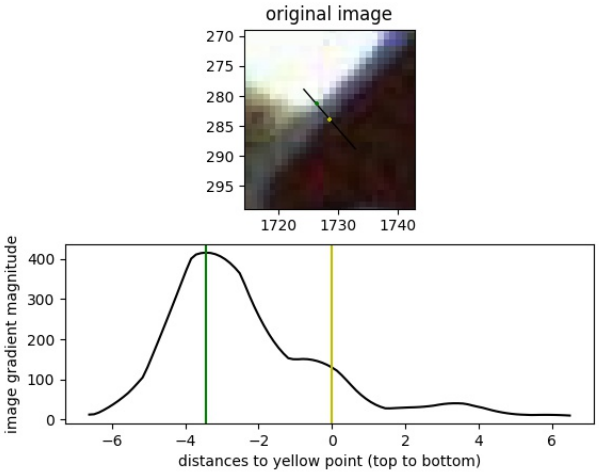


Figure 8: Gradient is Abnormal (black line: normal direction of the original line estimation; yellow point: a sampled point from the original line estimation; green point: the adjusted point of the yellow point)

This problem can be detected by checking if lines on opposite sides of the polygon are "almost" parallel (Due to the perspective effect, they won't be precisely parallel.), with a angle threshold of, say, no more than five degrees:

However, the following outlier is out of reach:

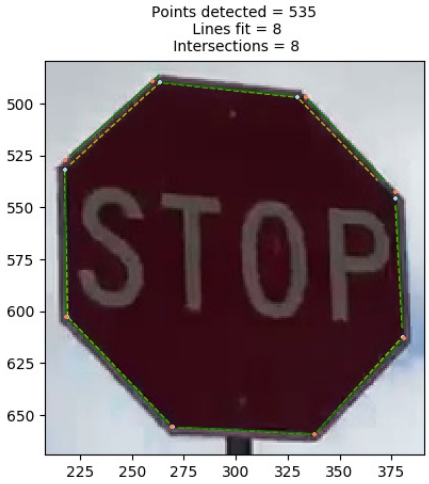


Figure 9: Upper-Left Edge & Upper-Right Edge is Over-adjusted (yellow dotted lines & light-blue crosses: before improvement; green solid lines & orange crosses: after improvement)

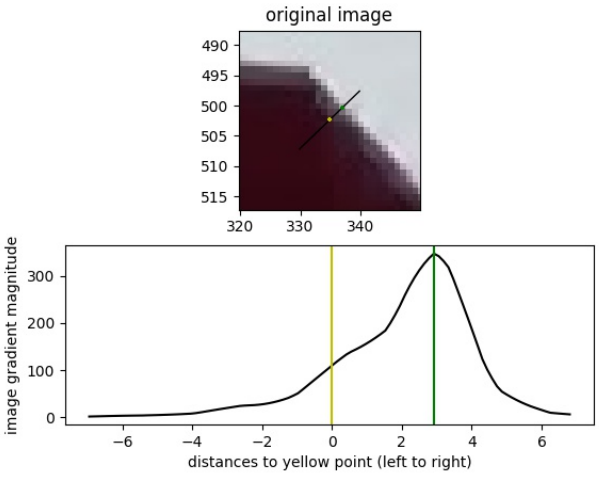


Figure 10: Gradient is Abnormal (black line: normal direction of the original line estimation; yellow point: a sampled point from the original line estimation; green point: the adjusted point of the yellow point)

4 Method 2

Here we focus on the second method: **Combine the information from video frames**.

In the application, the single stop sign is probably detected in several sequential frames. Using traditional planer calibration algorithm[1], each frame is treated separately, which means the authors ignore the correlation between each sequential frame.

In this report, we propose two possible techniques to combine these extra information based on the fact that the same stop sign is detected.

- Bundle Adjustment
- Filter Method

4.1 Background Knowledge

4.1.1 Notation Definition

Notation	Notation Description
w	World coordinate(fixed on the stop sign)
c_1	Camera coordinate for the first frame
c_2	Camera coordinate for the second frame
K	Camera intrinsic matrix
${}^{c_1}_{w}T$	Transform matrix of frame w to frame c_1
${}^{c_2}_{w}T$	Transform matrix of frame w to frame c_2
${}^{c_1}_{c_2}T$	Transform matrix of frame c_1 to frame c_2
${}^{c_2}_{c_1}T$	Transform matrix of frame c_2 to frame c_1
${}^{c_1}_{w}R$	Rotation matrix of frame w to frame c_1
${}^{c_2}_{w}R$	Rotation matrix of frame w to frame c_2
${}^{c_1}_{c_2}R$	Rotation matrix of frame c_1 to frame c_2
${}^{c_2}_{c_1}R$	Rotation matrix of frame c_2 to frame c_1
${}^{c_1}P_w$	The location of the origin of frame w in frame c_1
${}^{c_2}P_w$	The location of the origin of frame w in frame c_2
${}^{c_1}P_{c_2}$	The location of the origin of frame c_2 in frame c_1
X, Y, Z	Object points in frame w(Z=0)
u, v	Image points in the image plane

4.1.2 Transform Equations

Here we list some basic transform that are used in the derivatives:

$$ {}^{c_2}_{w}T = {}^{c_2}_{c_1}T \, {}^{c_1}_{w}T \tag{1} $$

Where, ${}^{c_2}_{w}T = \begin{bmatrix} {}^{c_2}_{w}R & {}^{c_2}P_w \\ 0 & 1 \end{bmatrix}$, ${}^{c_2}_{c_1}T = \begin{bmatrix} {}^{c_2}_{c_1}R & {}^{c_2}P_{c_1} \\ 0 & 1 \end{bmatrix}$, ${}^{c_1}_{w}T = \begin{bmatrix} {}^{c_1}_{w}R & {}^{c_1}P_w \\ 0 & 1 \end{bmatrix}$.

In most cases, the sensors can record the relative pose from the second frame c_2 to the first frame c_1. Hence, instead of using ${}^{c_2}_{c_1}T$, ${}^{c_1}_{c_2}T$ is the more practical information we can directly obtain from sensors. The relation between the two transform is simple:

$$ {}^{c_2}_{c_1}T = {}^{c_1}_{c_2}T^{-1} = \begin{bmatrix} {}^{c_1}_{c_2}R^T & -{}^{c_1}_{c_2}R^T {}^{c_1}P_{c_2} \\ 0 & 1 \end{bmatrix} \tag{2} $$

Hence, the equation (1) can be denoted as:

$$\underbrace{\begin{bmatrix} {}^{c_2}_{w}R & {}^{c_2}P_w \\ 0 & 1 \end{bmatrix}}_{\text{unknown}} = \underbrace{\begin{bmatrix} {}^{c_1}_{c_2}R^T & -{}^{c_1}_{c_2}R^{Tc_1}P_{c_2} \\ 0 & 1 \end{bmatrix}}_{\text{known}} \underbrace{\begin{bmatrix} {}^{c_1}_{w}R & {}^{c_1}P_w \\ 0 & 1 \end{bmatrix}}_{\text{unknown}} \tag{3}$$

Finally, we can obtain the following equations that are useful.

$$\begin{cases} {}^{c_2}_{w}R = & {}^{c_1}_{c_2}R^{Tc_1}_{w}R \\ {}^{c_2}P_w = & {}^{c_1}_{c_2}R^{Tc_1}P_w - {}^{c_1}_{c_2}R^{Tc_1}P_{c_2} \end{cases} \tag{4}$$

4.1.3 Camera Model

The camera project models at each frame can be represented as:

1. Camera frame c_1

$$q_1 = s_1 K W_1 Q \tag{5}$$

where, s_1 is an unknown scalar,$q_1 = \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}$, $W_1 = \begin{bmatrix} {}^{c_1}_{w}R & {}^{c_1}P_w \end{bmatrix}$ and $Q = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$

2. Camera frame c_2

$$q_2 = s_2 K W_2 Q \tag{6}$$

where, s_2 is an unknown scalar,$q_2 = \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix}$, $W_2 = \begin{bmatrix} {}^{c_2}_{w}R & {}^{c_2}P_w \end{bmatrix}$ and $Q = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$

For each frame, q_1 and q_2 can be computed by our previous algorithms and Q remains unchanged.

4.2 Bundle Adjustment

In the original paper, only the pose information with respect to each frame has bee used for building constraints. And then, we can use DLT method to solve the equations of system.

Before we study how to build new constraints, we first introduce some basic equations:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = sK \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = sK \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{7}$$

By abuse of notation, we still use $\mathbf{Q}$ to denote a point on the model plane, but $\mathbf{Q} = [X, Y]^T$ since Z is always equal to zero. In turn, $\overline{\mathbf{M}} = [X, Y, 1]^T$. Therefore, a model point $\mathbf{M}$ and its image $\mathbf{m}$ is related by a homography $\mathbf{H}$:

$$\widetilde{\mathbf{m}} = \mathbf{H}\overline{\mathbf{M}} \quad \text{with} \quad \mathbf{H} = sK \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \tag{8}$$

Given an image of the model plane, an homography can be estimated by pairs of matched points. Let's denote it by $\mathbf{H} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix}$. From (8), we have

$$\begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix} = sK \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix}, \tag{9}$$

$$
\begin{aligned}
h_1 &= sKr_1 \quad \text{or} \quad r_1 = \lambda K^{-1} h_1 \\
h_2 &= sKr_2 \quad \text{or} \quad r_2 = \lambda K^{-1} h_2 \\
h_3 &= sKt \quad \text{or} \quad t = \lambda K^{-1} h_3
\end{aligned} \tag{10}
$$

where s, λ is an arbitrary scalar. Using the knowledge that $\mathbf{r}_1$ and $\mathbf{r}_2$ are orthonormal($r_1^{\mathrm{T}} r_2 = 0$ and $\|r_1\| = \|r_2\|$, that is $r_1^{\mathrm{T}} r_1 = r_2^{\mathrm{T}} r_2$), we have

$$
\begin{aligned}
&\mathbf{h}_1^T K^{-T} K^{-1} \mathbf{h}_2 = 0 \\
&\mathbf{h}_1^T K^{-T} K^{-1} \mathbf{h}_1 = \mathbf{h}_2^T K^{-T} K^{-1} \mathbf{h}_2
\end{aligned} \tag{11}
$$

The traditional method can build two constraints for each frame. It dose not require that the relative poses between two sequential frames are known. In this special application, we study how to add new constraints by considering relative poses.

4.2.1 New Constraints

$$
\begin{aligned}
&{}^{c_1}_{w}R = \begin{bmatrix} \mathbf{r}_{11} & \mathbf{r}_{12} & \mathbf{r}_{13} \end{bmatrix} \\
&{}^{c_2}_{w}R = {}^{c_1}_{c_2}R^T \, {}^{c_1}_{w}R = {}^{c_1}_{c_2}R^T \begin{bmatrix} \mathbf{r}_{11} & \mathbf{r}_{12} & \mathbf{r}_{13} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{21} & \mathbf{r}_{22} & \mathbf{r}_{23} \end{bmatrix}
\end{aligned} \tag{12}
$$

Hence, for the first frame:

$$
\begin{bmatrix} \mathbf{h}_{11} & \mathbf{h}_{12} & \mathbf{h}_{13} \end{bmatrix} = s_1 K \begin{bmatrix} \mathbf{r}_{11} & \mathbf{r}_{12} & {}^{c_1}P_w \end{bmatrix} \tag{13}
$$

and for second frame:

$$
\begin{aligned}
\begin{bmatrix} \mathbf{h}_{21} & \mathbf{h}_{22} & \mathbf{h}_{23} \end{bmatrix} &= s_2 K \begin{bmatrix} \mathbf{r}_{21} & \mathbf{r}_{22} & {}^{c_2}P_w \end{bmatrix} \\
&= s_2 K \begin{bmatrix} {}^{c_1}_{c_2}R^T \mathbf{r}_{11} & {}^{c_1}_{c_2}R^T \mathbf{r}_{12} & {}^{c_1}_{c_2}R^T {}^{c_1}P_w - {}^{c_1}_{c_2}R^T {}^{c_1}P_{c_2} \end{bmatrix}
\end{aligned} \tag{14}
$$

Similarly,

$$
\begin{aligned}
h_{11} &= s_1 K r_{11} \quad \text{or} \quad r_{11} = \lambda_1 K^{-1} h_{11} \\
h_{12} &= s_1 K r_{12} \quad \text{or} \quad r_{12} = \lambda_1 K^{-1} h_{12} \\
h_{13} &= s_1 K {}^{c_1}P_w \quad \text{or} \quad {}^{c_1}P_w = \lambda_1 K^{-1} h_{13}
\end{aligned} \tag{15}
$$

$$
\begin{aligned}
h_{21} &= s_2 K r_{21} \quad \text{or} \quad r_{21} = \lambda_2 K^{-1} h_{21} \\
h_{22} &= s_2 K r_{22} \quad \text{or} \quad r_{22} = \lambda_2 K^{-1} h_{22} \\
h_{23} &= s_2 K {}^{2_1}P_w \quad \text{or} \quad {}^{c_2}P_w = \lambda_2 K^{-1} h_{23}
\end{aligned} \tag{16}
$$

Plug (14) into (16), we obtain:

$$
\begin{aligned}
{}^{c_1}_{c_2}R^T \mathbf{r}_{11} = \lambda_2 K^{-1} h_{21} \quad \text{or} \quad \mathbf{r}_{11} = \lambda_2 {}^{c_1}_{c_2}R K^{-1} h_{21} \\
{}^{c_1}_{c_2}R^T \mathbf{r}_{12} = \lambda_2 K^{-1} h_{22} \quad \text{or} \quad \mathbf{r}_{12} = \lambda_2 {}^{c_1}_{c_2}R K^{-1} h_{22}
\end{aligned} \tag{17}
$$

Right now, we have:

$$
\begin{aligned}
\mathbf{r}_{11} &= \lambda_1 K^{-1} h_{11} \\
\mathbf{r}_{12} &= \lambda_1 K^{-1} h_{12} \\
\mathbf{r}_{11}^{\text{new}} &= \lambda_2 {}^{c_1}_{c_2}R K^{-1} h_{21} \\
\mathbf{r}_{12}^{\text{new}} &= \lambda_2 {}^{c_1}_{c_2}R K^{-1} h_{22}
\end{aligned} \tag{18}
$$

The possible new constraints are:$\mathbf{r}_{11}^{\text{new}T} \mathbf{r}_{12} = 0$ or $\mathbf{r}_{11}^{\text{new}T} \mathbf{r}_{11}^{\text{new}} = \mathbf{r}_{11}{}^T \mathbf{r}_{11}$(the same for $\mathbf{r}_{12}^{\text{new}}$ and $\mathbf{r}_{12}$).

The question is: **How to use them for computation?**

1. $\mathbf{r}_{11}^{\text{new}T}\mathbf{r}_{12} = 0$

   The equation can be written as:

$$h_{21}^T K^{-T} \begin{matrix} c_1 \\ c_2 \end{matrix} R^T K^{-1} h_{12} = 0 \tag{19}$$

   However, it is a little different from (11):

   $\begin{matrix} c_1 \\ c_2 \end{matrix} R^T$ is in the middle of $K^{-T}K^{-1}$ and that would make it difficult to solve this linear equations(it is not a symmetric matrix any more).

   The symmetric matrix $K^{-T}K^{-1}$ actually describes the image of the absolute conic and since it is symmetric, after a few steps of derivatives, we can obtain:

$$\begin{aligned} \mathbf{B} = \mathbf{K}^{-T}\mathbf{K}^{-1} &\equiv \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{f_x^2} & 0 & \frac{-c_x}{f_x^2} \\ 0 & \frac{1}{f_y^2} & -\frac{c_y}{f_y^2} \\ \frac{-c_x}{f_x^2} & -\frac{c_y}{f_y^2} & \frac{c_x^2}{f_x^2} + \frac{c_y^2}{f_y^2} + 1 \end{bmatrix} \end{aligned} \tag{20}$$

   Note that B is symmetric, defined by a 6D vector

$$\mathbf{b} = [B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}]^T \tag{21}$$

   Let the i th column vector of $\mathbf{H}$ be $\mathbf{h}_i = [h_{i1}, h_{i2}, h_{i3}]^T$. Then, we have

$$\mathbf{h}_i^T \mathbf{B} \mathbf{h}_j = \mathbf{v}_{ij}^T \mathbf{b} \tag{22}$$

   with

$$\begin{aligned} &\mathbf{v}_{ij} = \\ &[h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j1}, h_{i2}h_{j2}, h_{i3}h_{j1} + h_{i1}h_{j3}, h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3}]^T \end{aligned} \tag{23}$$

   Therefore, the two equations in (11) can be rewritten as two homogeneous equations in b:

$$\begin{bmatrix} \mathbf{v}_{12}^T \\ (\mathbf{v}_{11} - \mathbf{v}_{22})^T \end{bmatrix} \mathbf{b} = \mathbf{0} \tag{24}$$

   Here, you can see that if $\begin{matrix} c_1 \\ c_2 \end{matrix} R^T$ is in the middle of $K^{-T}K^{-1}$, all the desired properties will be lost and all the variables are interconnected by nature.

2. $\mathbf{r}_{11}^{\text{new}T}\mathbf{r}_{11}^{\text{new}} = {\mathbf{r}_{11}}^T\mathbf{r}_{11}$

   The equation can be written as:

$$\begin{aligned} \lambda_2^2 h_{21}^T K^{-T} \begin{matrix} c_1 \\ c_2 \end{matrix} R^T \begin{matrix} c_1 \\ c_2 \end{matrix} R K^{-1} h_{21} &= \lambda_1^2 h_{11}^T K^{-T} K^{-1} h_{11} \\ \lambda_2^2 h_{21}^T K^{-T} K^{-1} h_{21} &= \lambda_1^2 h_{11}^T K^{-T} K^{-1} h_{11} \end{aligned} \tag{25}$$

   Similarly($\mathbf{r}_{12}^{\text{new}T}\mathbf{r}_{12}^{\text{new}} = {\mathbf{r}_{12}}^T\mathbf{r}_{12}$),

$$\lambda_2^2 h_{22}^T K^{-T} K^{-1} h_{22} = \lambda_1^2 h_{12}^T K^{-T} K^{-1} h_{12} \tag{26}$$

   If so, the quotient of λ_1 by λ_2 can be considered as one unknown variable, which results in the equations system that only involves one extra unknown variable.

   Besides, the most interesting result is that it does not need to know the relative pose between the two camera frames.

To do one step further, we can obtain:

$$\begin{aligned}\frac{\lambda_2^2}{\lambda_1^2}h_{22}^T K^{-T}K^{-1}h_{22} - h_{12}^T K^{-T}K^{-1}h_{12} = 0\\ \frac{\lambda_2^2}{\lambda_1^2}h_{21}^T K^{-T}K^{-1}h_{21} - h_{11}^T K^{-T}K^{-1}h_{11} = 0\end{aligned} \tag{27}$$

The only different part between 11 and 27 is that the quotient is unknown(for equations in 11, since it only involves single image, the scalars on both sides of the equations are the same, thus can be safely cancelled out). Here comes the tricky part:both scalars(λ_1 and λ_2) and intrinsic matrix K are unknown.

Our idea to deal with this problem is that:

1. Compute intrinsic matrix using previous algorithm
2. Plug the estimated intrinsic matrix into equation 27. Ideally, the quotient of λ_2^2 by λ_1^2 should be the same each equation. However, due to the noise, it can not be this case.
3. Hence, it would be interesting if we could modify the value of f_x and f_y to make the error as small as possible($^{upper}\frac{\lambda_2^2}{\lambda_1^2} - {}^{bottom}\frac{\lambda_2^2}{\lambda_1^2}$).
4. We could simply extend to the scenario that there are more than two sequential frames and the total error could be minimized.

We will first do the experiments to compare the error ($\sum(^{upper}\frac{\lambda_2^2}{\lambda_1^2} - {}^{bottom}\frac{\lambda_2^2}{\lambda_1^2})$) when we use groundtruch and the estimated results.

If such property holds true, We also have to figure out how to modify f_x and f_y to reduce the error.

4.3 Filter Method

From the other perspective, we can build the model as follow:

$$\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} = s_1 K \begin{bmatrix} \mathbf{r}_{11} & \mathbf{r}_{12} & {}^{c_1}P_w \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{28}$$

$$\begin{aligned}\begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} &= s_2 K \begin{bmatrix} \mathbf{r}_{21} & \mathbf{r}_{22} & {}^{c_2}P_w \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \\ &= s_2 K_{c_2}^{c_1} R^T \begin{bmatrix} \mathbf{r}_{11} & \mathbf{r}_{12} & {}^{c_1}P_w - {}^{c_1}P_{c_2} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \\ &= s_2 K_{c_2}^{c_1} R^T (\begin{bmatrix} \mathbf{r}_{11} & \mathbf{r}_{12} & {}^{c_1}P_w \end{bmatrix} - \begin{bmatrix} 0 & 0 & {}^{c_1}P_{c_2} \end{bmatrix}) \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \\ &= s_2 K_{c_2}^{c_1} R^T \begin{bmatrix} \mathbf{r}_{11} & \mathbf{r}_{12} & {}^{c_1}P_w \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} - s_2 K_{c_2}^{c_1} R^T \begin{bmatrix} 0 & 0 & {}^{c_1}P_{c_2} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \\ &= \frac{s_2}{s_1} K_{c_2}^{c_1} R^T K^{-1} \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} - s_2 \begin{bmatrix} 0 & 0 & K_{c_2}^{c_1} R^{T\,c_1} P_{c_2} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \\ &= \frac{s_2}{s_1} K_{c_2}^{c_1} R^T K^{-1} \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} - s_2 K_{c_2}^{c_1} R^{T\,c_1} P_{c_2}\end{aligned} \tag{29}$$

For the final expression, ${}^{c_1}_{c_2}R^T, {}^{c_1}P_{c_2}$ are known; $\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}$ and $\begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix}$ can be estimated by previous algorithms;K is the camera intrinsic matrix and s_1, s_2 are two different unknown scalars.

The question is that: **How can we modify the estimation of u_2 and v_2 by this iterative expression.**

4.4 Problem

You can see the second improvement method(Filter method) require accurate relative poses between sequential frames. If the sensor data themselves are noisy or they are unsynchronized, there is no guarantee that it could actually make benefits to the point estimations. Moreover, the calibration results would not be improved. The first improvement method(Bundle adjustment) is of more interest for us since it dose not involve the relative poses.

References

[1] Z. Zhang, "A flexible new technique for camera calibration," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 11, pp. 1330-1334, Nov. 2000, doi: 10.1109/34.888718.