Camera Intrinsic Calibration Using Stop Signs

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

July 31, 2020

1 Overview

The aim of this project is to calibrate the cameras without any available objects like chessboards when the vehicle is driving on the road. In the last winter break, we found some papers about self-calibration and compared the advantages and disadvantages of each method proposed by them. Finally, we decided to use license plates as the reference known objects for auto-calibration because at that moment, we thought they are the most common objects in the scenes.

The importance of auto-calibration for autonomous vehicles can be shown as follow:

In some papers, the researchers emphasize that the cameras' intrinsic and extrinsic parameters could change especially for these installed on the vehicles. As a result, some methods don't employ the hypothesis that the camera's parameters are constant and instead take them as varying variables. In *Geometric Stability of Low-cost Digital Consumer Cameras*, they consider the cameras' geometric stability when using the camera's features such as zoom or auto focus. In *The Effects of Temperature Variation on Single-Lens-Reflex Digital Camera Calibration Parameters*, they investigate the effect of variations in temperature on modern single-lens-reflex(SLR) digital cameras from a series of trials. The results show that temperature could affect camera calibration parameters.

Considering that the working conditions for autonomous vehicles could change a lot in two aspects:

- From the short-term perspective, the temperature is totally different during daytime or in the night. Besides, after a long journey on the bumpy road, the lens in the camera may move.
- From the long-term perspective, the temperature is totally different during different seasons. And it is harder to guarantee that every parts in the cameras are intact.

2 Related work

In *Iterative Calibration of a Vehicle Camera using Traffic Signs Detected by a Convolutional Neural Network*, they use detected traffic signs to calibrate a vehicle camera. The main principles behind it are the same as those used in chessboard calibration(with known objects). However, the results are not comparable with the ones computed from chessboard calibration. In *AutoCalib:*

Automatic Traffic Camera Calibration at Scale, researchers from Microsoft Research use vehicle models to calibrate the cameras. They extract the feature points on the detected vehicle from the image and associate them with the points on the real vehicle model. However, due to the fact that various vehicles made by different car companies have different vehicle models, the results are not good. In *A novel method for camera external parameters online calibration using dotted road line*, they use vanishing points obtained from road lines to calibrate the cameras. In *Camera Calibration from Video of a Walking Human*, they propose a method to obtain the vanishing points from a walking human.

3 Experiments and findings

In this case, we can only expect to extract four noisy corners points from each license plate on the images, which is not enough to obtain accurate results. To be more specific, we could not obtain any reasonable results only using license plate. Then, we did a series of simulations and found the positive relations between the number of points and the accuracy of results. Thus, we thought stop sign may be a better object for calibration since in ideal conditions, 24 points are available.

4 Project timeline

Here we briefly describe the work that have been done:

1. Winter vacation

   Try to extract license plate from videos.

2. Winter quarter

   a Fail to obtain accurate calibration results using only four corners

   b Study the kernel of the planer calibration algorithms and the factors that affect accuracy.

   c Decide to use stop sign for calibration

3. Spring quarter

   Build the experiment simulator and finish the whole system

5 Stop sign calibration

5.1 Simulator

Before doing experiments on the real stop signs, we first build a simulator in which we manually project a stop sign(shown in Fig1) onto an image using specified intrinsic and extrinsic matrix. The reason is that it is easier for us to compare the calibration results with the groundtruth(set manually).

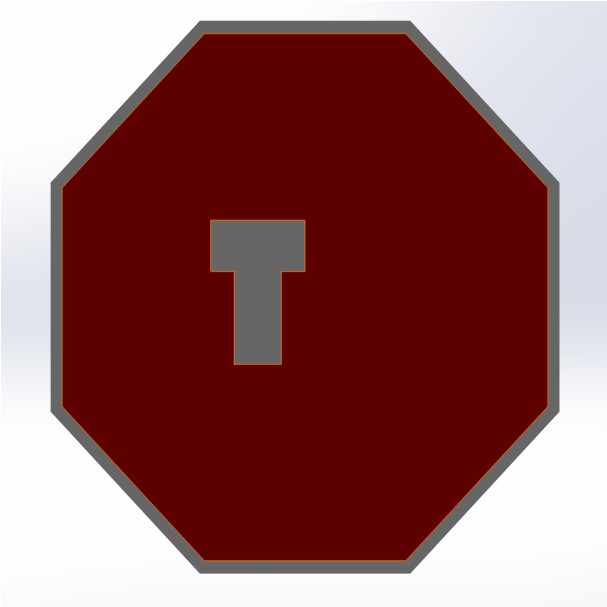


Figure 1: Reference object(stop sign)

This simulator is considered as a testbed for our whole system:

- Extraction of stop sign
- Detection of line points
- Line extraction from points
- Calibration

Here are two examples:

Figure 2: Simulated image

5.2 Real stop sign

We also took videos of real stop signs and input the images into the same system. Here are some example:

Figure 3: Real images

The image size of each is [720,1280] because these images were took in portrait mode.

The results shown below are all obtained from real images instead of them from simulator.

5.3 Extraction of Stop Signs

5.3.1 Method One - Gaussian Naive Bayes

The current system adopts the primitive algorithm of Gaussian Naive Bayes (GNB), to classify each pixel as belonging to a stop sign or not.

Given a pixel of three channels $x_* \in \mathbb{R}^3$, the GNB classifier generates a label $y_* \in \{\texttt{YES}, \texttt{NO}\}$ as follows:

$$y_* = \underset{y \in \{\texttt{YES},\texttt{NO}\}}{\arg\max} \log \theta_y^{MLE} + \sum_{l=1}^{3} \log \phi \left(x_{*l}; \mu_{yl}^{MLE}, \left(\sigma_{yl}^{MLE}\right)^2 \right)$$

where the three parameters ($k \in \{\texttt{YES}, \texttt{NO}\}, l \in \{1, 2, 3\}$)

$$\theta_k^{MLE} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1} \{y_i = k\}$$

$$\mu_{kl}^{MLE} = \frac{\sum_{i=1}^{n} x_{il} \mathbf{1} \{y_i = k\}}{\sum_{i=1}^{n} \mathbf{1} \{y_i = k\}}$$

$$\sigma_{kl}^{MLE} = \sqrt{\frac{\sum_{i=1}^{n} \left(x_{il} - \mu_{kl}^{MLE}\right)^2 \mathbf{1} \{y_i = k\}}{\sum_{i=1}^{n} \mathbf{1} \{y_i = k\}}}$$

are trained with n labeled pixels: $(x_i, y_i) \in \mathbb{R}^3 \times \{\texttt{YES}, \texttt{NO}\}$.

Once the image with pixel-wise labels is obtained, candidate stop sign areas can be proposed by grouping connected pixels together. By enforcing prior knowledge of stop signs (like the area range and the aspect ratio), false positives can be filtered. In this way, the pixel-wise mask and bounding box for each stop sign are extracted.

5.3.2 Results - GNB

Figure 4: GNB Mask 1 (Left: original image; Middle: Bounding Box; Right: Pixelwise Mask)

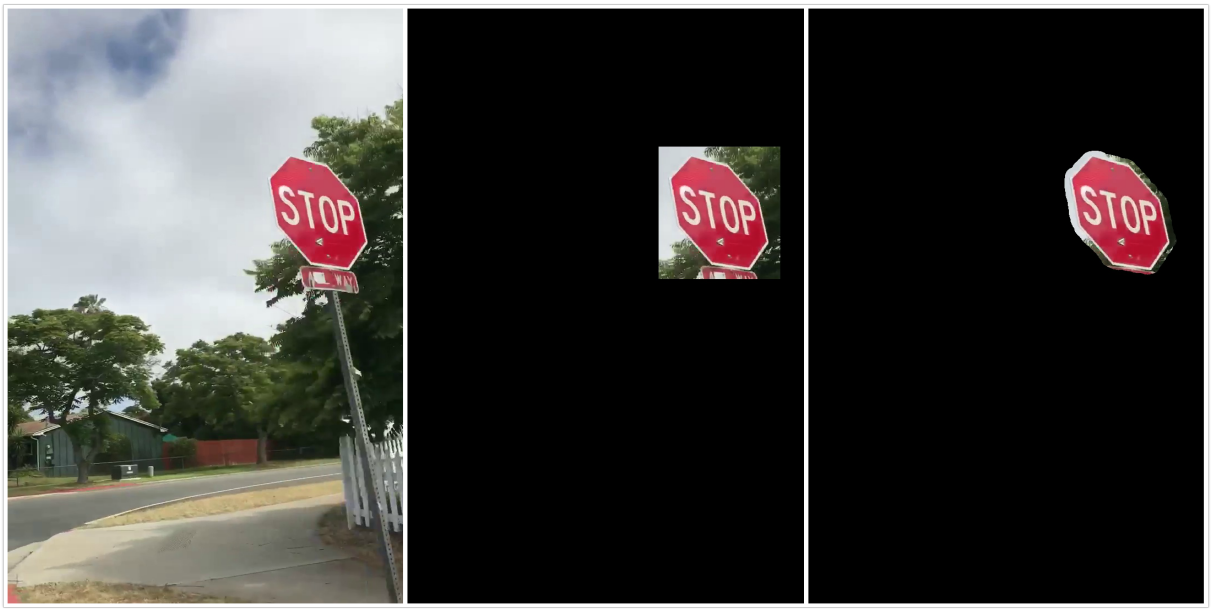


Figure 5: GNB Mask 2 (Left: original image; Middle: Bounding Box; Right: Pixelwise Mask)

5.3.3 Issues - GNB

1. Training The classifier was trained to classify only the redness of the stop signs - the white borders and the characters are not labelled as stop signs; The training dataset contains only images with forth-faced, clear-captured stop signs under good weathers - the classifier has trouble with worse-conditioned stop sign images.

Figure 6: Unconnected Stop Sign Mask due to Undetected White Areas

Figure 7: Undetected Ill-faced Stop Sign

2. Model The GNB is a statistical model based only on the pixel colors. It does not learn any semantic meaning of stop signs. Objects like a traffic light with similar colors as the stop sign's red can be classified as stop signs (false positives), while stop signs in shadows, under- or over-exposure (false negatives) may be dismissed.

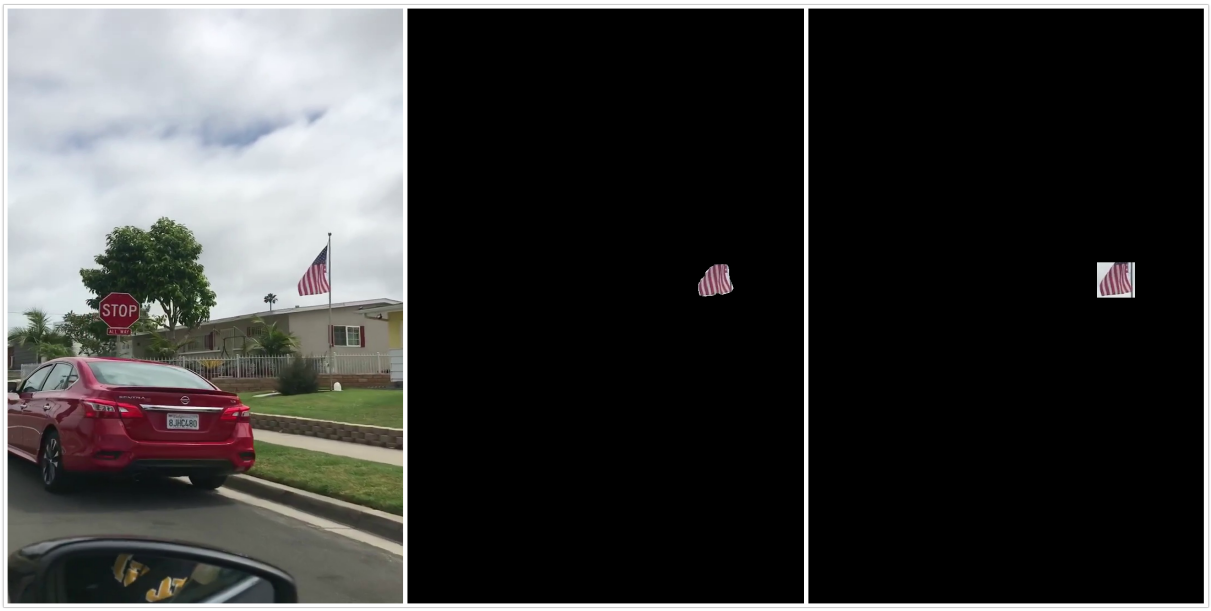


Figure 8: False Positive with Similar Color as Stop Signs

Figure 9: Undetected Stop Sign due to Under-exposure

In the future, deep learning methods (e.g. object detection, semantic segmentation, instance segmentation) should be introduced to get more precise extraction.

5.3.4 Method Two - HSV Color Extraction

This is a simpler method, but outperforms the GNB method in terms of robustness to various color conditions (e.g. illumination, saturation).

To obtain the pixel-wise segmentation, two steps are needed:

1. Conversion Convert the original image to the HSV color space;
2. Thresholding Label pixels within a range of (Hue, Saturation, Value) triple as the foreground, while the rest background.

In this way, (almost) all red pixels are extracted as candidates. And similar grouping and filtering methods (as in Section 5.3.1) can be applied to attain the pixel-wise masks and bounding boxes.

5.3.5 Results - HSV

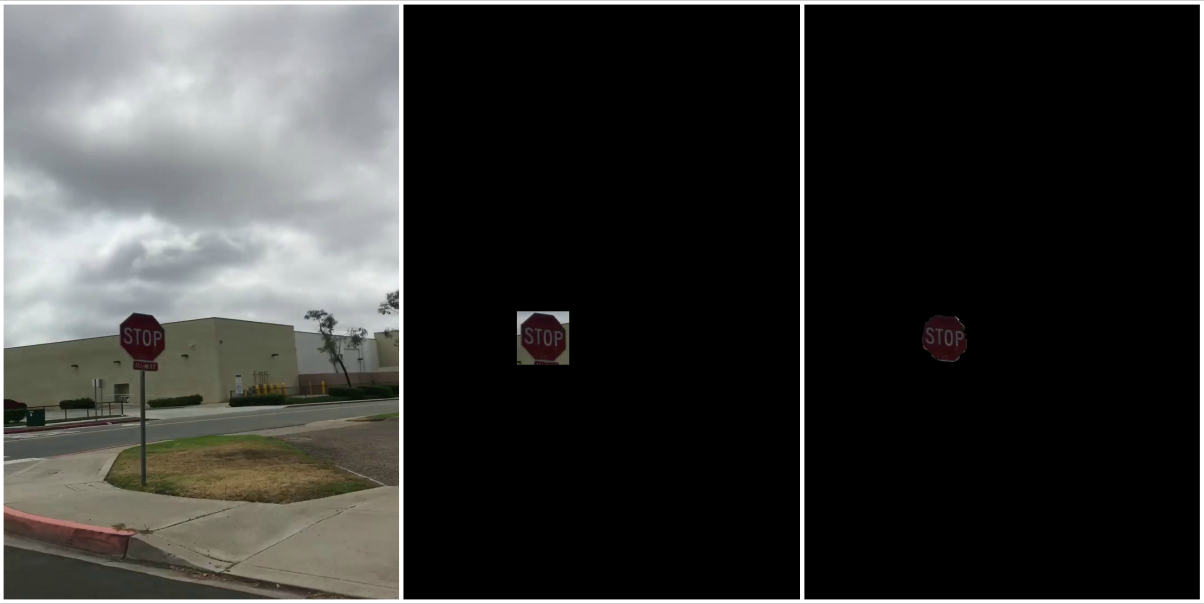


Figure 10: HSV Mask 1 (Left: original image; Middle: Bounding Box; Right: Pixelwise Mask)

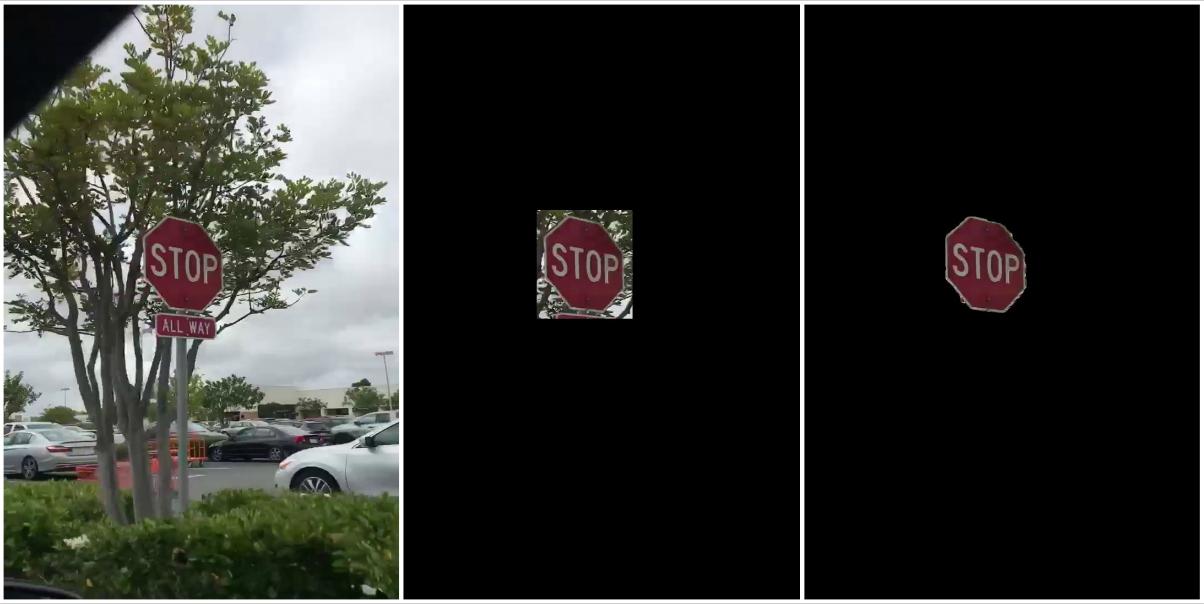


Figure 11: HSV Mask 2 (Left: original image; Middle: Bounding Box; Right: Pixelwise Mask)

The under-exposure and unconnected issues in Section 5.3.3 are solved as shown above.

5.3.6 Issues - HSV

1. False Positives HSV method extracts all red pixels in the image, rather than the pixels in stop-sign red as in the GNB method. Thus, better post-processing algorithm is desired to filter out (more) false positives.

Figure 12: Pixel-wise Segmentation 1 (Left: original image; Middle: GNB; Right: HSV)

Figure 13: Pixel-wise Segmentation 2 (Left: original image; Middle: GNB; Right: HSV)

5.4 Detection of line points

We employed the method proposed in *A Sub-Pixel Edge Detector: an Implementation of the Canny/Devernay Algorithm(method of extracting line points with sub-pixel accuracy)* and detected all the points(with sub-pixel accuracy) whose pixel gradients are larger than a specified threshold.

5.4.1 Algorithm

a. Canny Method

Canny proposed to use the first-derivative Gaussian filter for two-dimensional signals (images) by analyzing the one-dlimensional signal along the direction n, normal to the

boundary. Thus, edge points are defined to the local maximum in the direction n of

$$\frac{\partial}{\partial n} G \star I$$

The direction n orthogonal to the boundary can be estimated by the image gradient

$$n \approx \frac{\nabla(G \star I)}{|\nabla(G \star I)|}$$

Which corresponds to the norm of the gradients along x and y axis.

This method of keeping only local maximum is called *non-maximum suppression* and it is very vulnerable to random noise. Thus, Canny proposed to discriminate true edges using two thresholds, H and L. Local maximum with strength above the high threshold H are immediately validated and added to the output; local maximum connected to validated points are validated if their strength is above the low threshold L.

In all, Canny's method depends on three parameters: S, the standard deviation of the Gaussian kernel G, and the two thresholds H and L.

b. The Devernay Sub-Pixel Correction

Canny's method finds edge pixels. In other words, edge points are extracted up to the pixel grid precision. In some applications one may require to have the position of the edge points with finer accuracy. For that aim, Devernay proposed a very elegant addition to the original Canny algorithm to produce sub-pixel accuracy edge points.

Suppose the pixel point B is a local maximum of $||g||$ in the direction of the image gradient and the gradient at point A and point C can be approximated by:

$$\|g(A)\| \approx \frac{g_x(B)-g_y(B)}{g_x(B)}\|g(D)\| + \frac{g_y(B)}{g_x(B)}\|g(E)\|$$
$$\|g(C)\| \approx \frac{g_x(B)-g_y(B)}{g_x(B)}\|g(G)\| + \frac{g_y(B)}{g_x(B)}\|g(F)\|$$

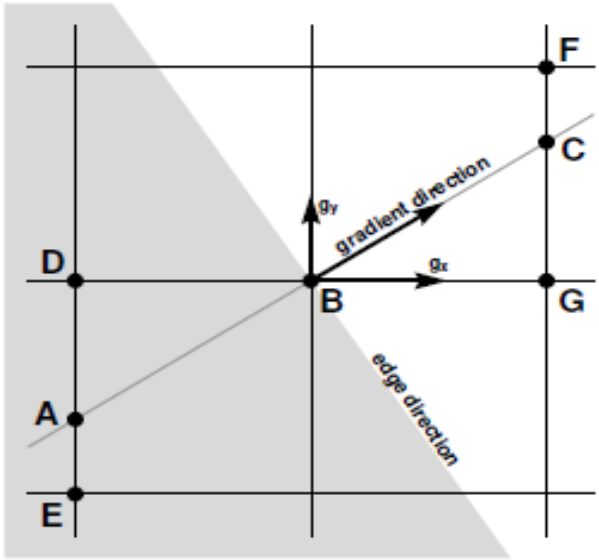


Figure 14: Canny operator

Devernay proposed that the position of an edge point is refined as the maximum of an interpolation of the gradient norm; the correction is as simple as computing a quadratic interpolation of the gradient norm between three neighboring positions along the gradient direction.

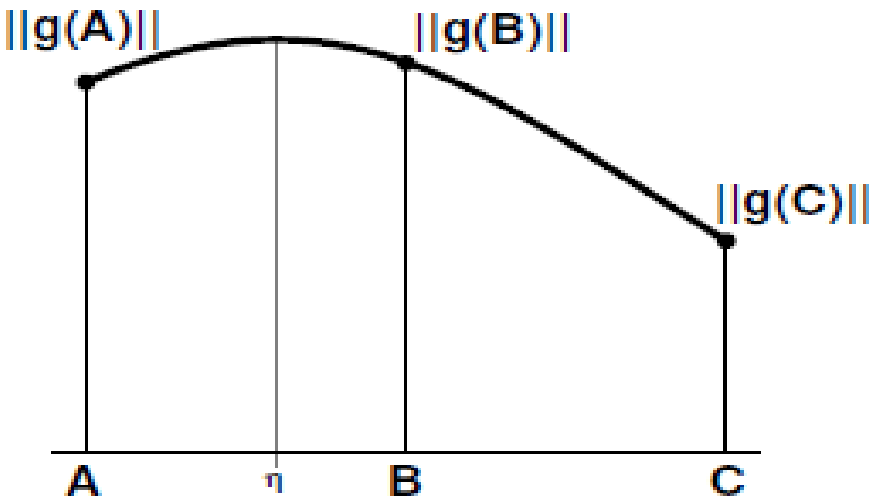


Figure 15: Profile of the norm of the image gradient along the direction of the image gradient

Fig14 and Fig15 shows an example: Canny's method keeps pixel B as an edge pixel because $\|g(B)\| > \|g(A)\|$ and $\|g(B)\| > \|g(C)\|$. However, there may be an intermediate position η between A and C where the norm of the image gradient is larger than in those points. Point η would correspond better to the position of the edge. Devernay proposed a method to estimate that position and then compute an offset vector, along the direction n, to give the sub-pixel position of the edge point near the edge pixel B

To limit the computational cost, Devernay proposed to use a simple quadratic interpolation of the gradient norm along the three points used in the Canny operator: $(A, \|g(A)\|)$, $(B, \|g(B)\|)$ and $(C, \|g(C)\|)$. Solving the maximum point leads to an offset:

$$\eta = \frac{1}{2} \frac{\|g(A)\| - \|g(C)\|}{\|g(A)\| + \|g(C)\| - 2\|g(B)\|}$$

relative to the vector $\overrightarrow{BC}$.

c. Edge Point Chaining

Each edge point is computed independently from the others and the ones that belong to the same edge need to be chained to form the curves corresponding to the contours of the image. A simple procedure can be used for the chaining: connect an edge point to the nearest other edge points not farther than a certain tolerance.

The first verification that is performed is that edge points to be chained have a similar gradient orientation. To make a chaining from A to B, it is required that the angle between $g(A)$ and $g(B)$ be less than 90 degree. This is verified by the condition $g(A) \cdot g(B) > 0$, where $\cdot$ is the dot product.

The second verification is that: An image contour separates a light region from a darker one. The chaining needs to be coherent in the sense that consecutive chains must leave the darker region to the same side of the curve. A simple way of imposing this is to verify that the vector from edge point A to edge point B to be chained is roughly orthogonal to the image gradient in A in one of the two possible directions. As a convention, we will call a forward chaining from edge point A to edge point B one in which $\overrightarrow{AB} \cdot g(A)^{\perp} > 0$, where $\vec{v}^{\perp}$ corresponds to vector $\vec{v}$ rotated 90 degrees.

More details are included in the previous paper I mentioned.

5.4.2 Results

a. Extraction of line points in RGB color space

Here, four sets of extracted points, which corresponds to the four images in Fig3 respectively, are shown:

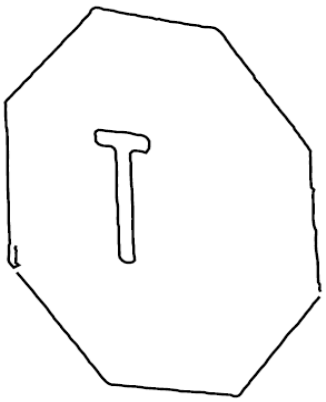


Figure 16: Output1

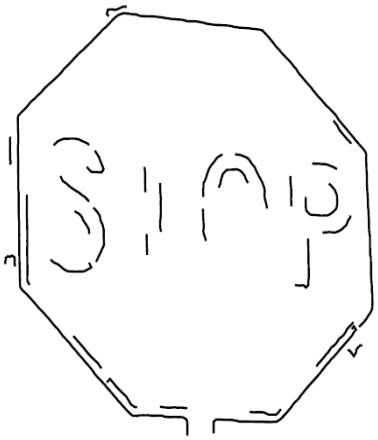


Figure 17: Output2

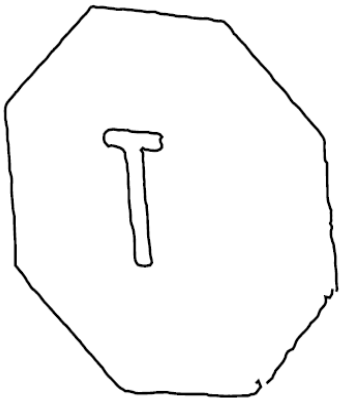


Figure 18: Output3

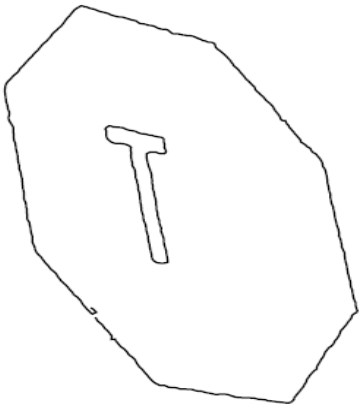
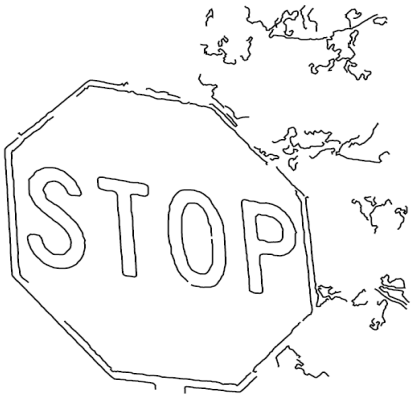


Figure 19: Output4

From the above four figures, output1, output3 and output4 are perfect and can be safely used in the next part. However, output2 is not good:many edges can not be detected. This problem happens due to the bad image quality.

You can see that we decide to only use the inner octagon and the character T because we found that if the zebra crosswalk or other objects(for example, branches) intersects with the stop sign, it is hard to distinguish the outer octagon. Besides, after experiment, only 12 corner points for each image are enough to provided good calibration result.

Besides, in some cases, there exist much more lines points than we want and it would make it really hard to classify these points.

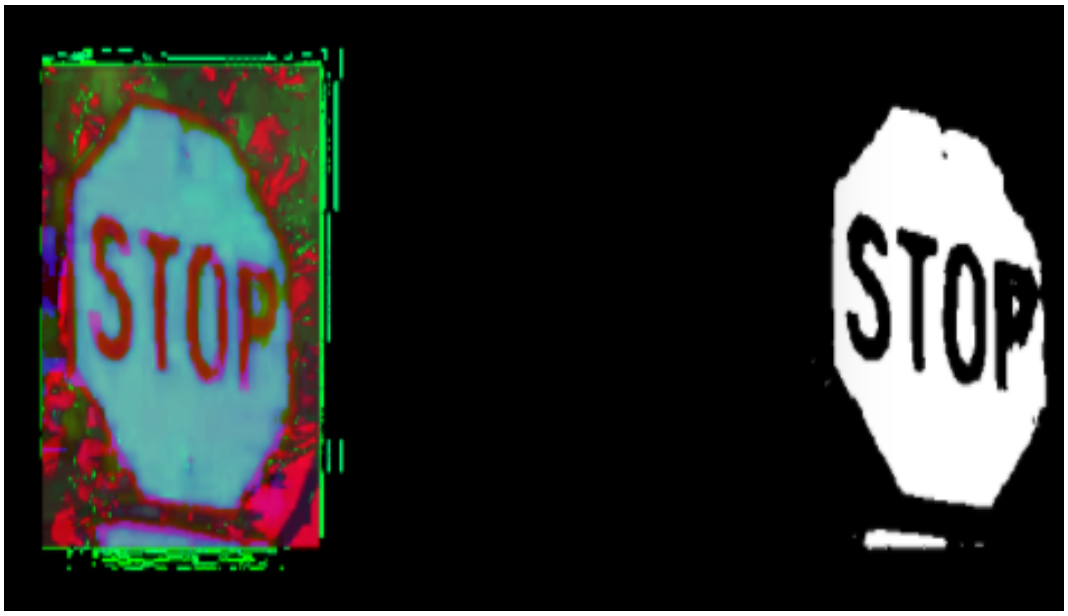


As described before, we also try to extract line points in HSV color space and we find both the advantages and disadvantages.

b. Extraction of line points in HSV color space

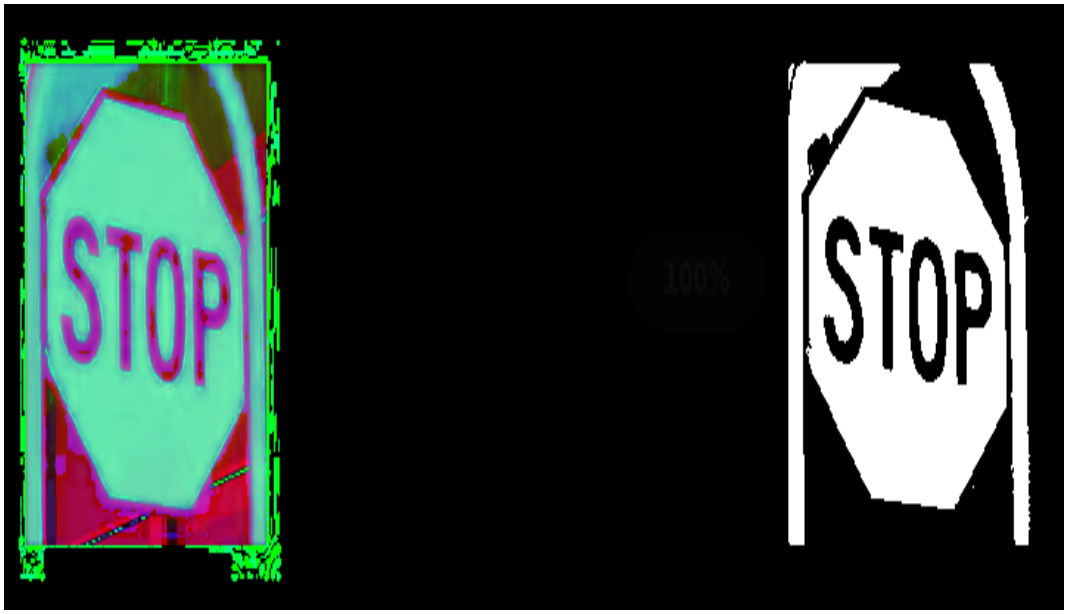
We set a threshold and obtain the mask image.

Advantage: You can see that all the branches or any other objects which are not red can be removed in the mask image.



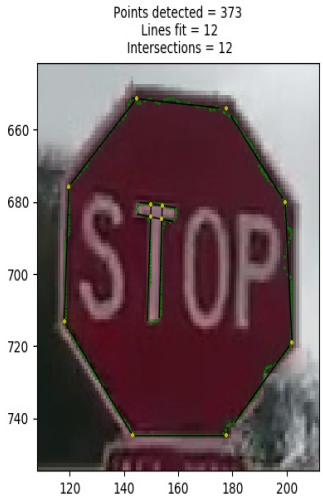
Disadvantage: We may lose some boundary information due to bad lighting conditions(shadows on the stop sign) or some blurs.

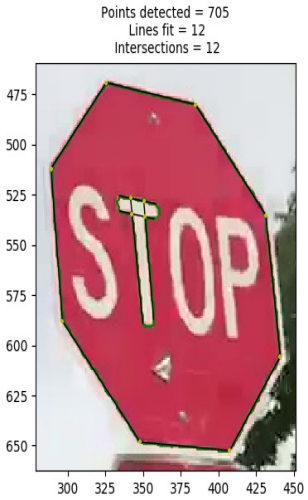
This would cause serious problems for our calibration algorithm since high accuracy is required.



We refer to this image as "high-quality image" in our project. The common features are that they have good lighting conditions and no blurs or damages can be found on the stop sign.

The quality of the images determines the accuracy of estimated corner points.





At the last, when we run calibration algorithm with the points detected from these high-quality images, we can obtain amazing results.

5.4.3 Unsolved problems

1. High-quality image

   Right now, We want to use the HSV image for this part. Here comes the main problem:how can we automatically distinguish these high-quality images from the bad ones.

   On the other side, this would require much more images that be fed into our system. However, we don't think it is a big problem because when we drive on the road, the stop signs are the most common thing. Besides, It is much easier to obtain the high-quality images on a sunny day for its good lighting conditions.

   In our current system, this part involves much manual work. However, if this algorithm is applied to autonomous driving car in the future, we have to figure out a method to distinguish bad images and discard them.

   The data association that will be introduced in the next part has the potential possibility to solve this problem.

5.5 Edge Estimation and Intersection Filtering

5.5.1 Edge Estimation Method

The algorithm in section 5.4 generates sub-pixel accurate edge points of the inner polygon and the character 'T', respectively. By iteratively applying RANSAC over SVD on the points, line segments of edges can be estimated, and their intersections are the candidates for corners of the inner polygon and the character 'T'.

Given N homogeneous coordinates of 2D points

$$X = \begin{pmatrix} x_1 & y_1 & 1 \\ & \vdots & \\ x_N & y_N & 1 \end{pmatrix},$$

three line parameters

$$n = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

are to be found, so that $||X \cdot n||_2^2$ is minimized. This can be solved by getting the Singular Value Decomposition (SVD) of the homogeneous points matrix $X = U \cdot \Sigma \cdot V^T$, where the line vector n is the column vector of matrix V corresponding to the smallest singular value in Σ.

The points of any edge can not be separated from the whole point set, but a line supported by most "close points" (to be defined later) must belong to one of the edges. Here, a RANSAC algorithm is used to vote for this line:

Repeat for R times:

1. Sample Randomly sample k kernel points X_{kernel} from the whole point set X (in practice, $k = 2$ as the minimum required points of a line);
2. Fit Fit a line n to these kernel points X_{kernel} with the SVD explained above;
3. Evaluate Calculate the distance of all points in X to the line n:

   $$D = \frac{|X \cdot n|}{\sqrt{a^2 + b^2}}$$

   and count the number of "close points", which are points with distance no more than a threshold d_{thresh} (in practice, $d_{thresh} = 1$ to get sub-pixel accurate line estimation) as the "correctness".

The line n corresponding to the largest "correctness" is the best line fitting the most points.

Remove the support points, and repeat above procedure for L times, then L edges can be estimated ($L = 8$ for inner polygon, $L = 4$ for character 'T').

One consideration is to choose a proper repeat number R for the RANSAC, so that R is as small as possible to make a efficient algorithm, while large enough to guarantee hitting the edge with probability $p \in [0, 1]$:

$$R = \frac{\log_2(1-p)}{\log_2(1-(1-\epsilon)^k)},$$

where k is the kernel size, and $\epsilon \in [0, 1]$ is the probability of observing an outlier in the points set. To find the ith $i \in \{1, \dots L\}$ edge, $\epsilon = 1 - \frac{1}{L-i+1}$. If noisy points (points that don't belong to any edge) are considered, ϵ should be increased accordingly.

5.5.2 Intersection Filtering Method

Only part of the line intersections from section 5.5.1 belongs to the corners. To filter out the outliers, intersections are required to be close (within a threshold) to at least one of the terminal points of the corresponding line segments.

However, this doesn't work with corners in between the segment's terminals (e.g. 'T's' center corners). To account for this, intersections close (within a threshold) to the points' centroid are also included.

Further, corners of convex shapes (e.g. 8 corners of the inner polygon, or 4 center corners of the character 'T') can be sorted in (counter-)clockwise order, according to the angles of the lines connecting them to the centroid.

5.5.3 Results

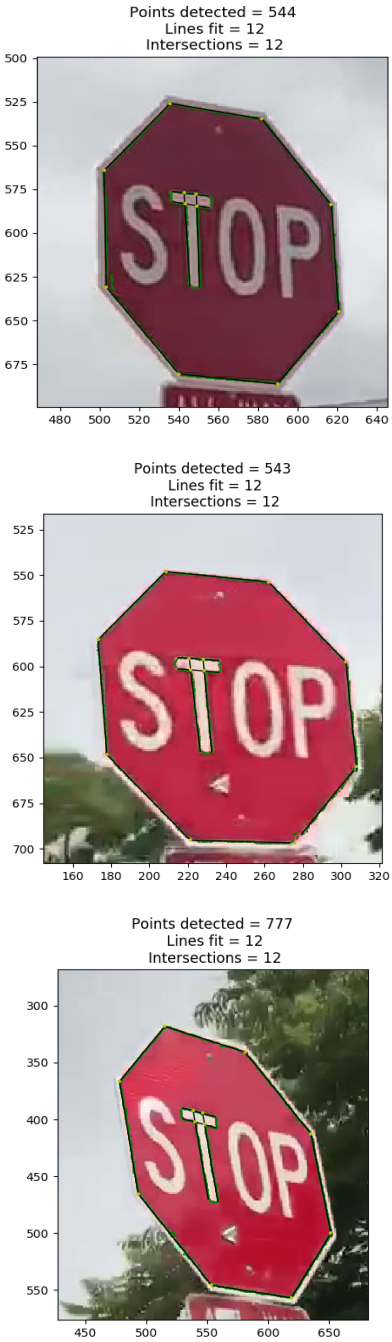


Figure 20: Examples (Green Dots: Edge Points from Section 5.4; Black Lines: Estimated Line Segments of Edges; Yellow Crosses: Filtered Intersection Points)

5.5.4 Issues

1. Edge Estimation RANSAC assumes that the number of outliers are less than that of inliners. If not the case, the algorithm should fail.

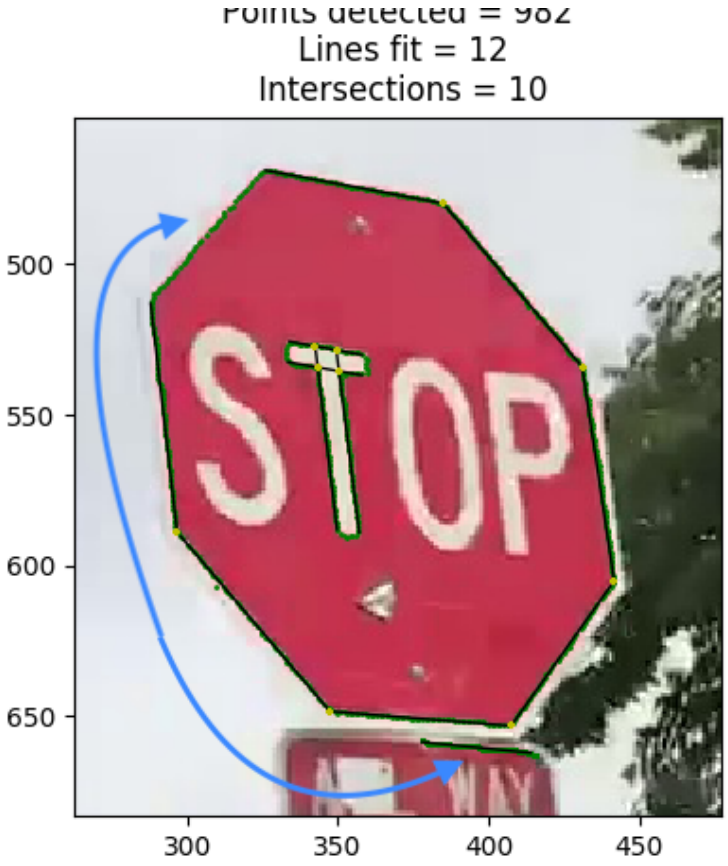


Figure 21: Miss-estimated Line Segment

2. Intersection Filtering The assumptions

   - corners are close to the terminals;
   - corners are close to the centroid

   are strong. It is quite common to have unexpected situations like

   - estimated segments are longer or shorter than the edges, so intersections are dismissed from corners because they are not close to them segments' terminals;

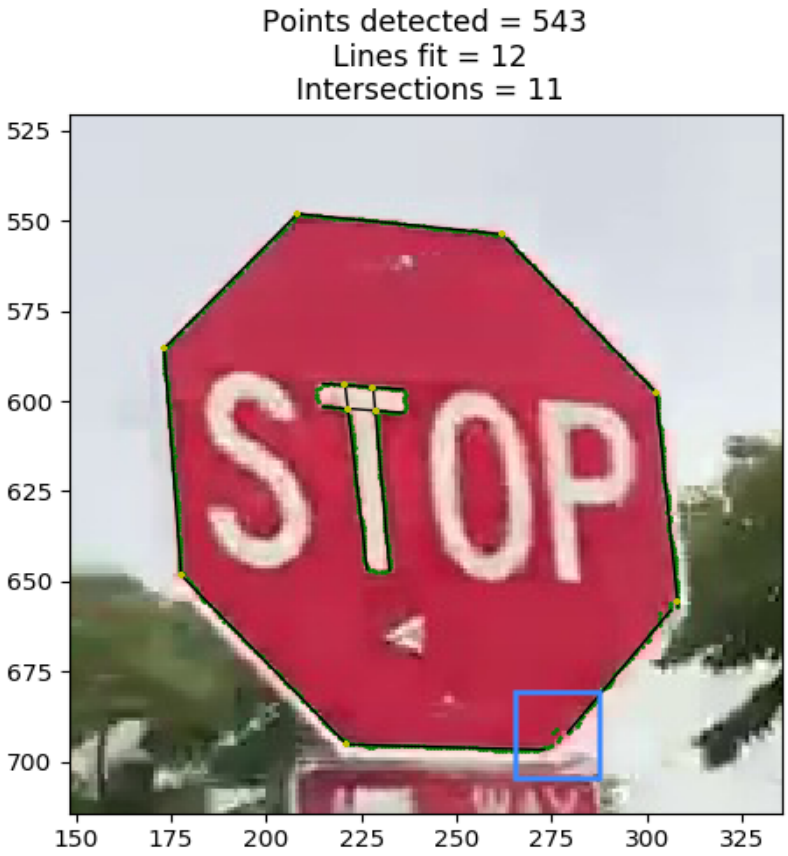


Figure 22: Dismissed Corner due to Short Line Segment

- an intersection is close to a terminal, but is not a corner point.

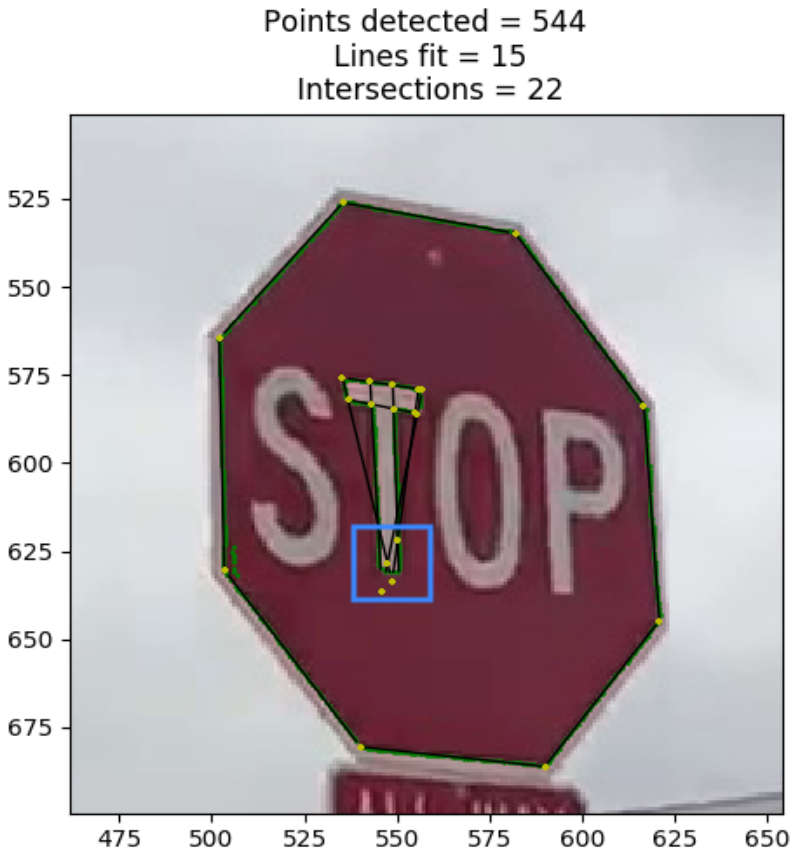


Figure 23: Confusing Intersections

5.6 Calibration

The calibration kernel is based on the famous algorithm:plane calibration. It is also the algorithm behind the chessboard calibration. Here, the algorithm detail are first proposed by Zhang in his famous paper *Flexible Camera Calibration By Viewing a Plane From unknown Orientations*.

5.6.1 Algorithm

1. Notation

   A 2D point is denoted by $\mathrm{m} = [u, v]^T$. A 3D point is denoted by $\mathrm{M} = [X, Y, Z]^T$. We use $\widetilde{\mathrm{x}}$ to denote the augmented vector by adding 1 as the last element: $\tilde{\mathbf{m}} = [u, v, 1]^T$ and $\tilde{\mathrm{M}} = [X, Y, Z, 1]^T$. A camera is modeled by the usual pinhole: the relationship between a 3 D point M and its image projection m is given by

   $$s\tilde{\mathbf{n}} = \mathbf{A}[\mathbf{R} \quad \mathbf{t}]\tilde{\mathbf{M}}$$

   where s is an arbitrary scale factor, $(\mathbf{R}, \mathbf{t})$, called the extrinsic parameters, is the rotation and translation which relates the world coordinate system to the camera coordinate system, and $\mathbf{A}$, called the camera intrinsic matrix, is given by

   $$\mathbf{A} = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

with (u_0, v_0) the coordinates of the principal point, α and β the scale factors in image u and v axes, and γ the parameter describing the skewness of the two image axes We use the abbreviation $\mathbf{A}^{-T}$ for $\left(\mathbf{A}^{-1}\right)^T$ or $\left(\mathbf{A}^T\right)^{-1}$.

2. Homography between the model plane and its image

   Without loss of generality, we assume the model plane is on $Z = 0$ of the world coordinate system. Let's denote the i^{th} column of the rotation matrix $\mathbf{R}$ by $\mathbf{r}_i$. From (1), we have

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

   By abuse of notation, we still use M to denote a point on the model plane, but $M = [X, Y]^T$ since Z is always equal to 0. In turn, $\widetilde{\mathrm{M}} = [X, Y, 1]^T$. Therefore, a model point M and its image m is related by a homography H:

$$s\tilde{\mathbf{m}} = \mathbf{H}\mathbf{M} \quad \text{with} \quad \mathbf{H} = \mathbf{A}\left[\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{t}\right] \tag{2}$$

   As is clear, the 3×3 matrix H is defined up to a scale factor.

3. Constraints on the intrinsic parameters

   Given an image of the model plane, an homography can be estimated (see Appendix A). Let's denote it by $\mathbf{H} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix}$. From (2), we have

$$\begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix} = \lambda \mathbf{A} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix}$$

   where λ is an arbitrary scalar. Using the knowledge that r_1 and r_2 are orthonormal, we have

$$\mathbf{h}_1^T \mathbf{A}^{-T} \mathbf{A}^{-1} \mathbf{h}_2 = 0 \tag{3}$$

$$\mathbf{h}_1^T \mathbf{A}^{-T} \mathbf{A}^{-1} \mathbf{h}_1 = \mathbf{h}_2^T \mathbf{A}^{-T} \mathbf{A}^{-1} \mathbf{h}_2 \tag{4}$$

   These are the two basic constraints on the intrinsic parameters, given one homography. Because a homography has 8 degrees of freedom and there are 6 extrinsic parameters (3 for rotation and 3 for translation), we can only obtain 2 constraints on the intrinsic parameters. Note that $\mathrm{A}^{-T}\mathrm{A}^{-1}$ actually describes the image of the absolute conic [16]. In the next subsection, we will give an geometric interpretation.

4. Closed-form solution using DLT method Let

$$\mathrm{B} = \mathrm{A}^{-T}\mathrm{A}^{-1} \equiv \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{bmatrix} = \begin{bmatrix} \frac{1}{\alpha^2} & -\frac{\gamma}{\alpha^2\beta} & \frac{v_0\gamma - u_0\beta}{\alpha^2\beta} \\ -\frac{\gamma}{\alpha^2\beta} & \frac{\gamma^2}{\alpha^2\beta^2} + \frac{1}{\beta^2} & -\frac{\gamma(v_0\gamma - u_0\beta)}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} \\ \frac{v_0\gamma - u_0\beta}{\alpha^2\beta} & -\frac{\gamma(v_0\gamma - u_0\beta)}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} & \frac{(v_0\gamma - u_0\beta)^2}{\alpha^2\beta^2} + \frac{v_0^2}{\beta^2} + 1 \end{bmatrix} \tag{5}$$

   The above equation can be simplified if we assume $\gamma = 0$.

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

$$\mathbf{b} = [B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}]^T \tag{6}$$

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

$$\mathbf{h}_i^T \mathbf{B} \mathbf{h}_j = \mathbf{v}_{ij}^T \mathbf{b} \tag{7}$$

with

$$\mathbf{v}_{ij} = [h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j1}, h_{i2}h_{j2}$$
$$h_{i3}h_{j1} + h_{i1}h_{j3}, h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3}]^T$$

Therefore, the two fundamental constraints (3) and (4), from a given homography, can be rewritten as 2 homogeneous equations in $\mathbf{b}$

$$\begin{bmatrix} \mathbf{v}_{12}^T \\ (\mathbf{v}_{11} - \mathbf{v}_{22})^T \end{bmatrix} \mathbf{b} = \mathbf{0} \tag{8}$$

If n images of the model plane are observed, by stacking n such equations as (8) we have

$$\mathbf{V}\mathbf{b} = \mathbf{0} \tag{9}$$

The solution of the above equation to is well known as the eigenvector of $\mathrm{V}^T\,\mathrm{V}$ associated with the smallest eigenvalue (equivalently, the right singular vector of V associated with the smallest singular value).

5. Maximum likelihood estimation using optimization method The solution from DLT method can be set as the initial value for optimization. Indeed, only a few iterations are needed.

5.7 Results

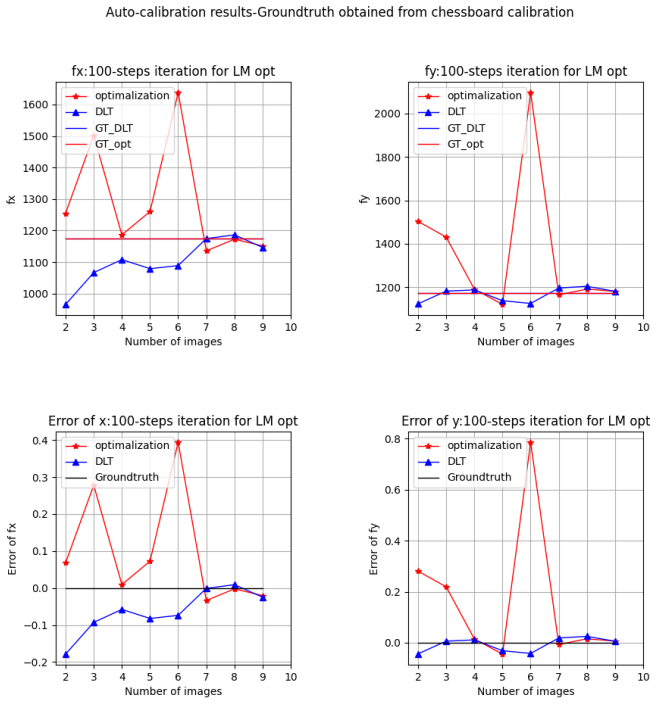


Figure 24: Calibration result

The relative errors when using more than 7 high-quality images are very small for both f_x and f_y.

Table 1: The number of high-quality images:9

Relative error	DLT	Opt(100-steps)
f_x	2.483%	2.037%
f_y	0.562%	0.5618%

Table 2: The number of high-quality images:8

Relative error	DLT	Opt(100-steps)
f_x	0.907%	0.2348%
f_y	2.503%	2.037%

5.8 Issue

Here, we consider the chessboard calibration result as the groundtruth since it is commonly used. If necessary, we can use another way to verify the correctness of out results:

Estimate the distance between the camera and an plane object using the obtained intrinsic matrix. The object is placed at a known distance. Compare the estimated distance and the true distance.

6 Improvements

There are two main issues:

- Automatic corner points matching algorithm
- Collect more data and do the same experiment for the generality

7 Importance of Data Association

7.1 Goal

Given intersections from Section 5.5, the goal is to assign each stop sign corner from polygons or characters (`image point`) its physical coordinates (`physical point`) on the plane, and filter out non-corner intersections.

7.2 Potential Method

The idea is bridging the `image points` and the `physical points` with a reference stop sign image with pre-associated corners.

Now the problem degenerates to matching points between images. A straight-forward solution is:

1. Encode point information (local and global) into a descriptor;
2. Find the closest descriptor (i.e. shortest distance) on the other image and match them;
3. A distance threshold can be used to remove outliers.

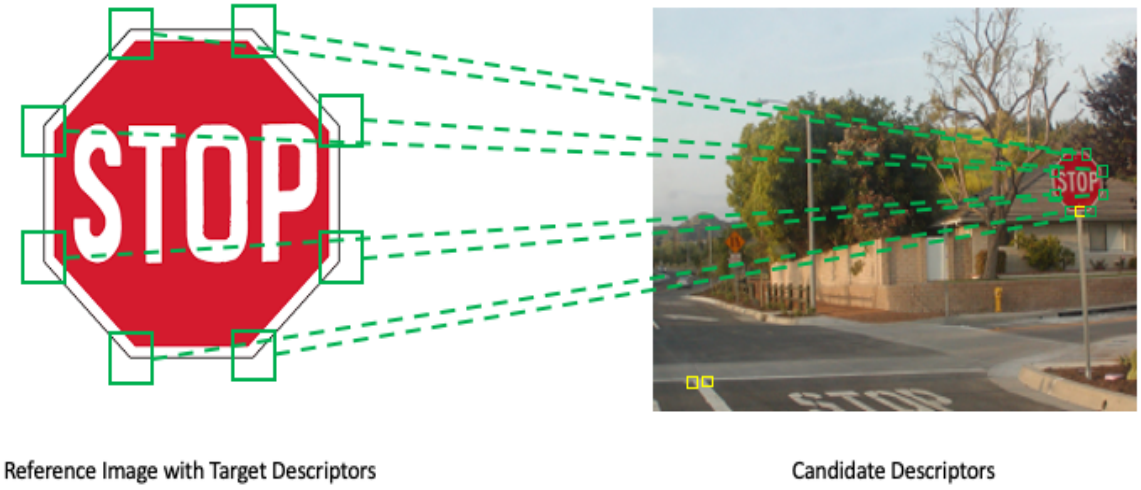


Figure 25: Data Association Method

7.3 Benefits

An effective data association method solves many previous issues:

- Higher tolerance for miss-detected stop signs or noisy points - can be filtered out;
- Relieve the system from manual point matching;
- No need to separate line points in character 'T' or polygons.
- Keep the high-quality images and discard the bad ones.