CV Challenge Documentation

G02

July 2019

Contents

0.0 Group members

Name	Vorname	Email	Immatrik-Nr
Guan	Fuqi	fuqi.guan@tum.de	03665875
Wu	Fan	ge49tet@mytum.de	03714421
Yu	Hang	hang.yu@tum.de	03712417
Zhao	Yidong	yidong.zhao@tum.de	03710074
Zheng	Hanwen	hanwen.zheng@tum.de	03710585

0.1 Source and Reference

Disparity Map:

1. `https://courses.cs.washington.edu/courses/cse455/09wi/Lects/lect16.pdf`

2. `http://vision.deis.unibo.it/~smatt/Seminars/StereoVision.pdf`

T,R, Epipolar:

1. Matlab grader

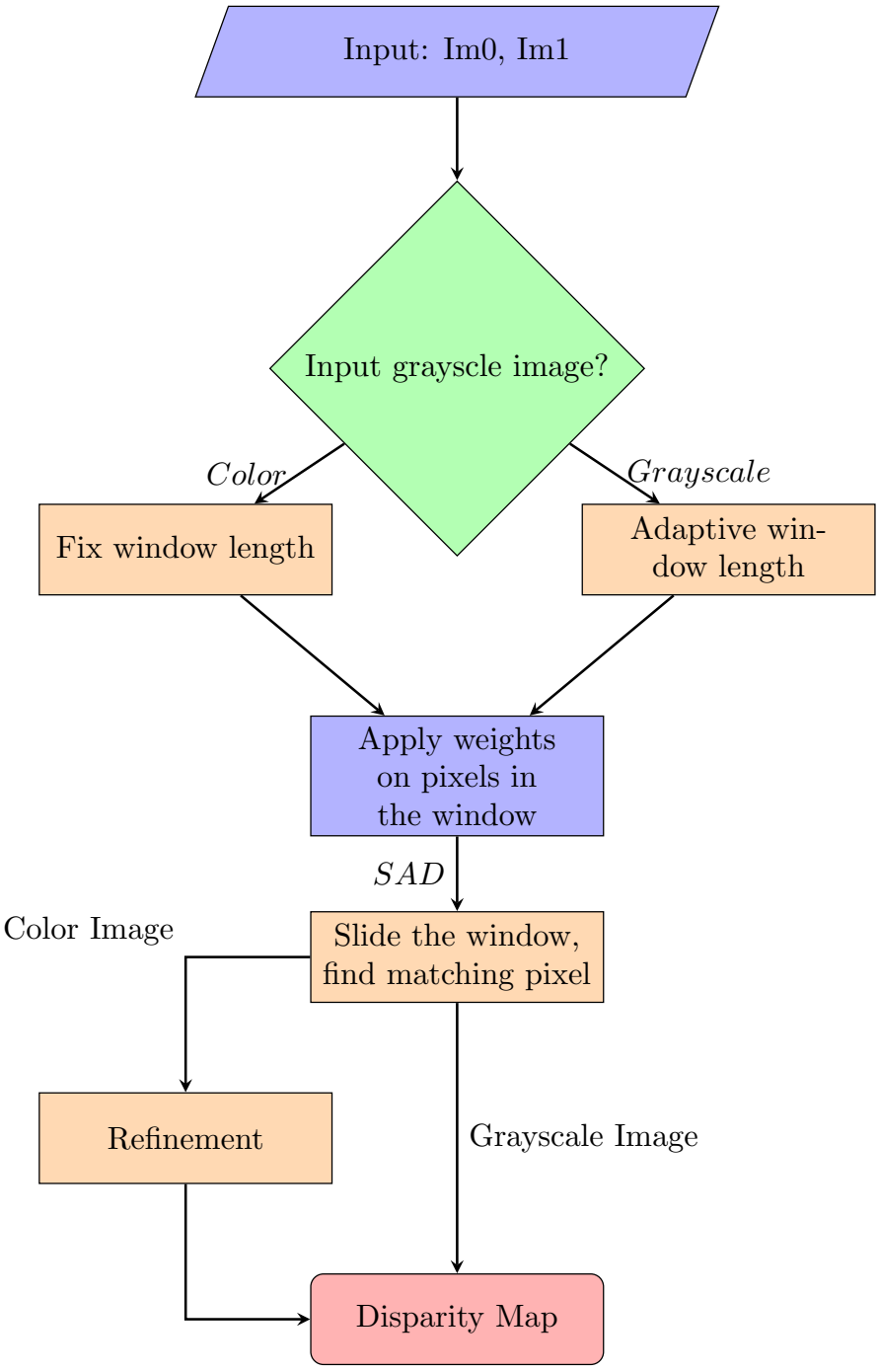
2. For Debug reference: `https://bitbucket.org/computervision/computer-vision/src/master/`

PFM format file parser:
`http://www.cs.virginia.edu/~cab6fh/pfmio/parsePfm.m`

0.2 Overview

In the next page a flowchart of the core algorithms (Disparity Map) is introduced. And later the details about each process in the flowchart is discussed in chapter 1. The bonus function 3D-Plot is also shown in chapter 2.

At the end, the algorithm for Translation and Rotation calculation is also explained, which was also implemented during the CV-homework in Matlab grader.


Input: Im0, Im1
Input grayscle image?
Color
Grayscale
Fix window length
Adaptive win-
dow length
Apply weights
on pixels in
the window
SAD
Color Image
Slide the window,
find matching pixel
Refinement
Grayscale Image
Disparity Map

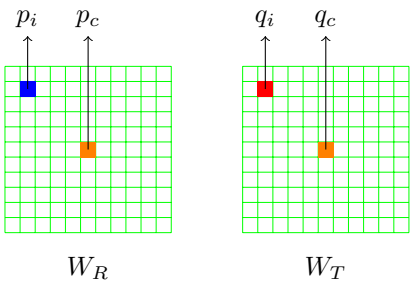
Chapter 1

Disparity Map

1.1 Adaptive Weighting

We apply the the adaptive weighting algorithm, a powerful local block based stereo-matching method, to compute the initial disparity map. The idea is, when calculating the costs in the two windows, namely the total absolute difference between those corresponding pixels, those which are closer to, and have similar color with the central pixel matter more than the marginal ones or those with different color. Thus the weight at each pixel is negatively correlated to the distance and the color difference to the central pixel.

For gray scale images, it is very hard to get a good estimation in the areas of slanted surfaces and repetitive patterns. If the intensity inside a block doesn't change that much, we can then enlarge the block size until it includes more pixels such that the block contains not only the homogeneous areas.



The cost function of two windows is as follows:

$$C(p_c, q_c) = \frac{\sum\limits_{p_i \in W_R, q_i \in W_T} w_r(p_i, p_c) \cdot w_t(q_i, q_c) \cdot AD(p_i, q_i)}{\sum\limits_{p_i \in W_R, q_i \in W_T} w_r(p_i, p_c) \cdot w_t(q_i, q_c)}$$

where p_i and q_i are two corresponding pixels in windows of image_0 and image_1, AD is Absolute Difference function of the two pixels and γ_p and γ_c are self-defined coefficients, in this challenge we set $\gamma_p = (windowsize - 1)/_2$ and $\gamma_c = 2\gamma_p$

$$w_r = e^{-\frac{d_p(p_i, p_c)}{\gamma_p}} \cdot e^{-\frac{d_c(I(p_i), I(p_c)}{\gamma_c}}$$

$$w_t = e^{-\frac{d_p(q_i, q_c)}{\gamma_p}} \cdot e^{-\frac{d_c(I(q_i), I(q_c)}{\gamma_c}}$$

1.2 Post-Refinement

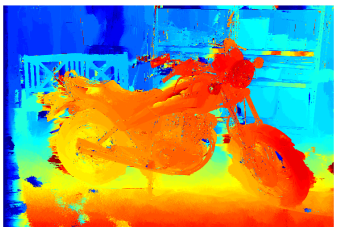
The difference of quality between disparity maps estimated by different matching criteria is not that large as expected. Post refinement of the disparity map is therefore of great significance. The refinement algorithm we applied in the challenge is based on the idea that there shouldn't be large disparity changes along a homogeneous surface with a similar color.

Firstly we scan along each horizontal line, and divide the line into smaller segments, in each segment pixels have a similar color. For those pixels whose disparity value deviates a lot from the median value of disparity on the segment, they will be smoothed to the median value. Then we do the same in a vertical manner. However the smoothing alongside scan-line causes severe scan-line effect. To solve this, we apply a median filter to the disparity image. Algorithm 1 shows the refinement process:

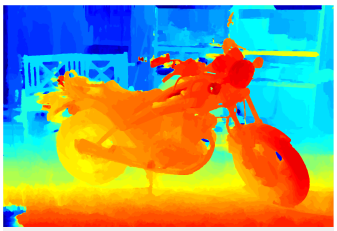
Algorithm 1 Refine disparity map

Input: im0, Dmap
Output: Dmap
width, height $\leftarrow size(im0)$
for *i = 1 to height* **do**
  $start_x \leftarrow 1$
  for *j = 1 to width* **do**
    $d = DistColor(im0(i,j), im0(i, start_x)$
    if $d > thres_1$ **then**
      $med = median(Dmap(i, start_x : j))$
      for $k = start_x$ *to* j **do**
        if $|Dmap(i,k) - med| > thres_2$ **then**
          $Dmap(i,k) = med$
        end
      end
    end
  end
end
for *j = 1 to width* **do**
  $start_y \leftarrow 1$
  for *i = 1 to height* **do**
    $d = DistColor(im0(i,j), im0(start_y, j)$
    if $d > thres_1$ **then**
      $med = median(Dmap(start_y : i, j))$
      for $k = start_y$ *to i)* **do**
        if $|Dmap(k,j) - med| > thres_2$ **then**
          $Dmap(k,j) = med$
        end
      end
    end
  end
end
MedianFilter(Dmap)

Figure 1.1 shows the result of adaptive weighting algorithm and the refined disparity map. As is shown in the figure, after refinement, we got an improvement of more than 2 dB on PSNR.



(a) Before refinement(15.8507 dB)



(b) After refinement(17.9088 dB)

Figure 1.1: Disparity map of motorcycle

1.3 Results

Figure1.2, 1.3, 1.4 and 1.5 show the results of our disparity map estimation algorithm:

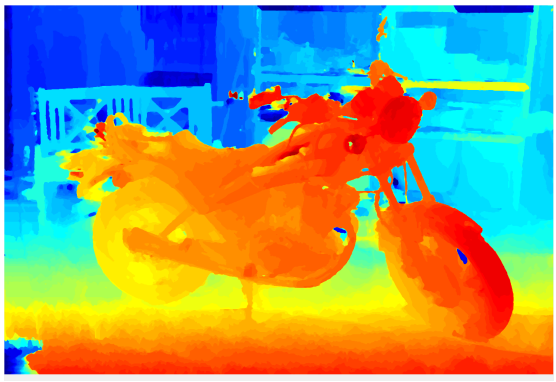


Figure 1.2: Motorcycle disparity map(17.9088 dB)

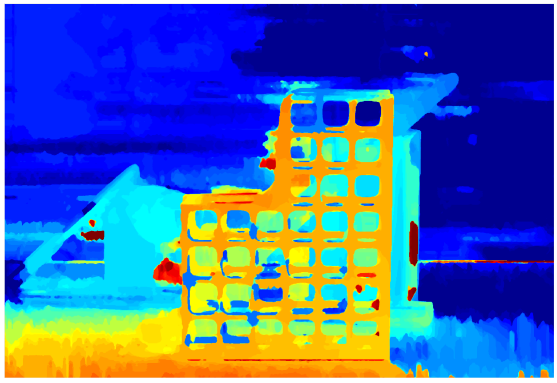


Figure 1.3: Sword disparity map(13.8110 dB)

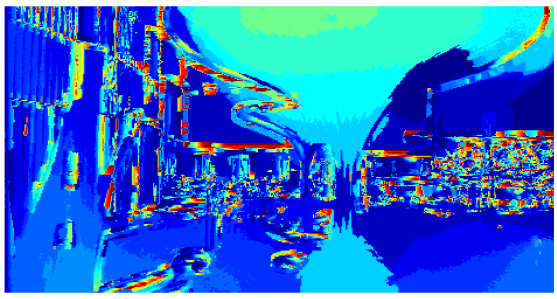


Figure 1.4: Playground disparity map(9.1365 dB)

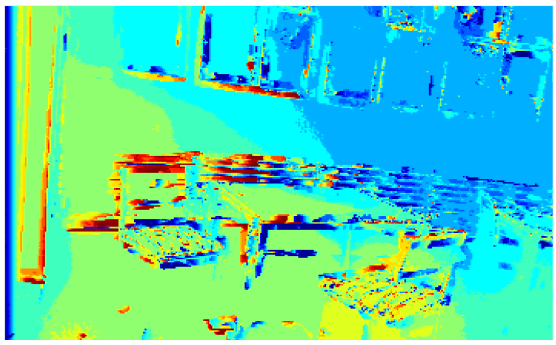


Figure 1.5: Terrace disparity map(9.3818 dB)

As is shown in the figures, our algorithm works good on the two color images, but has a poor performance on the gray scale images, due to the lack of color information, repetitive patterns and slanted surfaces in the scene.

Chapter 2

3D Plot

This chapter introduces 3D Plot function that is realized besides the main task. After calculating disparity map of the two images, we reconstruct the 3-D scene from 2-D images using disparity map and other calibration parameters and plot each point with color of original pixel.

To convert from the floating-point disparity value d(pixels) in the .pfm file to depth Z(mm), the following equation can be used:

$$Z = \frac{baseline \times f}{d + doffs}$$

where *baseline* is camera baseline in mm, f is focal length in pixels, and *doffs* is x-difference of principal points. These parameters have been provided in .pfm file.

The relationship between pixel coordinates and normalized coordinates can be written as

$$\boldsymbol{x}^{(hom)} = \boldsymbol{K^{-1}x'}$$

where $\boldsymbol{K}$ is intrinsic parameter matrix provided in .pfm file. We transform pixel coordinates into normalized coordinates using this formula.

After calculating normalized coordinates, we want to project points in focal plan into world frame. The equivalence between point $\boldsymbol{x}^{(hom)}$ in camera frame and point $\boldsymbol{P}$ in world frame can be described as

$$\boldsymbol{x}^{(hom)} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \boldsymbol{P} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Now that we know Z and $\boldsymbol{x}^{(hom)}$, it is easy to calculate world coordinates of $\boldsymbol{P}$. The calculation is realized in **plot_3D.m**. The results of four scenes are shown as below.

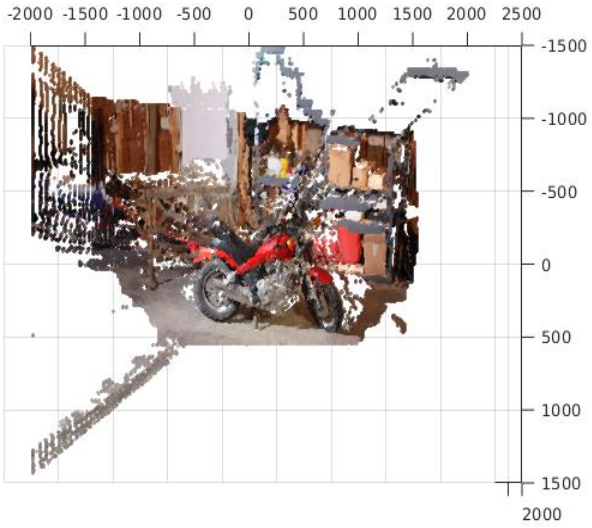


Figure 2.1: 3D plot of Motor

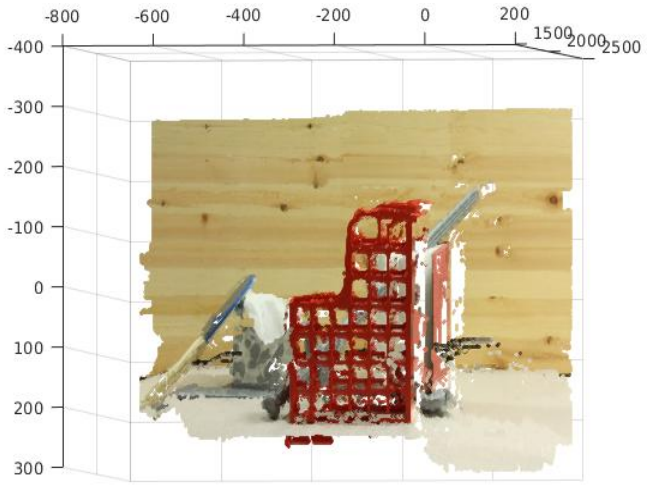


Figure 2.2: 3D plot of Sword

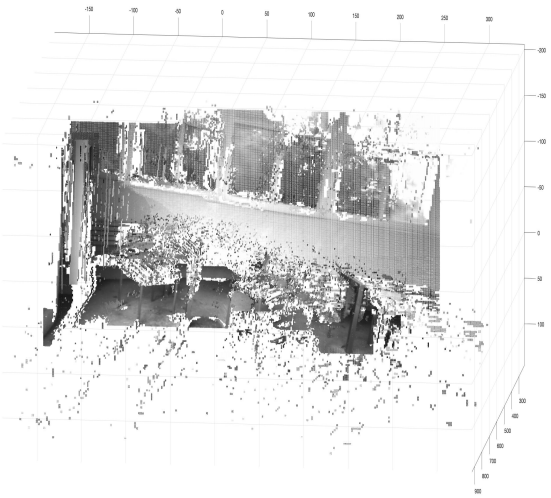
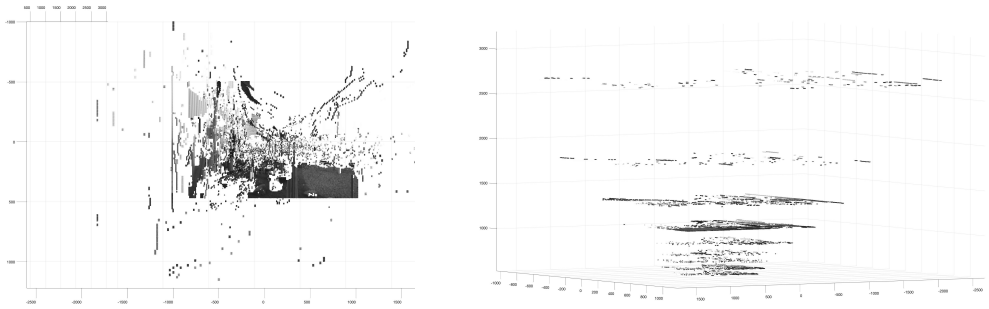


Figure 2.3: 3D plot of Terrace



(a) 3D plot of Playground (b) 3D plot of Playground distance levels

Figure 2.4: 3D Playground

Chapter 3

Euclidean movement

3.1 Extract features with Harris detector

First we use Sobel filter to get image gradients in x- and y- direction separately. Then use Harris detector to determine if a given pixel (x,y) is a corner feature. Harris detector can be calculated as follows.

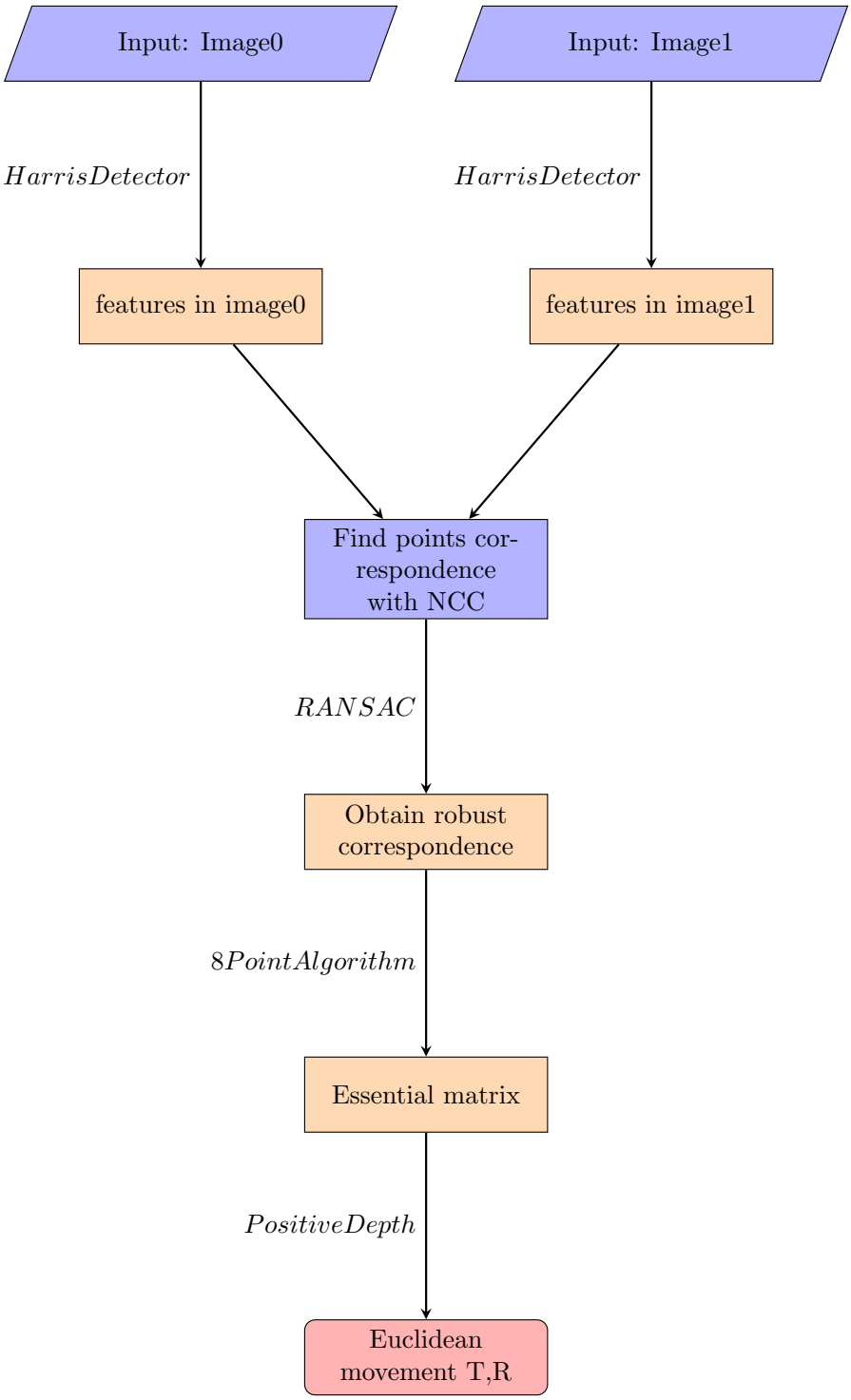
$$H := det(G) - k(tr(G))^2 = (1 - 2k)\lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2$$

where

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

k is the parameter between the corner and edge. For efficient Harris implementation, we set τ as the threshold for corner-decision. If the smallest singular value $\sigma_{min(G)}$ is bigger than the prefixed threshold τ, then mark the pixel as a feature (corner) point.

The detector above only selects the features via a global threshold value τ, but the features over the image can vary greatly. To solve that, we first eliminate weak features below a small threshold to get rid of noise. Then we divide the image into smaller tiles, within a tile only the strongest features are stored. In order to prevent multiple features of one and the same real object point from being extracted, we set a minimal distance (in pixels) between two features. By introducing the distance and the global threshold, and set the maximum number of features within a tile as N, only $M < N$ features per tile are extracted at last.


Input: Image0
Input: Image1
$HarrisDetector$
$HarrisDetector$
features in image0
features in image1
Find points correspondence with NCC
$RANSAC$
Obtain robust correspondence
$8PointAlgorithm$
Essential matrix
$PositiveDepth$
Euclidean movement T,R

3.2 Find point correspondences

For further stereo algorithms, we want to get pairs of correspondence points from the previously obtained feature points.

In general, we can use Sum of Squared Difference(SSD) to find the correspondent points. For given point V_i in the first image, we can find correspondent feature point W_j in the second image with the criteria $j = \text{argmin}_k\, d(V_i, W_k)$, where $d(V, W) = \|V - W\|_F^2$. However, SSD method can be strongly affected with brightness and rotation.

Therefore, we need to process the points first. We create windows to match the regions around the feature points, in each window, we normalize the intensity and location of the points. For normalization, first subtract the mean values, and then divide by standard deviation.

$$W_n := \frac{1}{\sigma(W)}(W - \overline{W})$$

where $\overline{W} = \frac{1}{N}(11^T W 11^T)$, $\sigma(W) = \sqrt{\frac{1}{N-1}\left\|W - \overline{W}\right\|_F^2}$.

We use Normalized Cross Correlation(NCC) to find correspondent points, defined as $\frac{1}{N-1} tr(W_n^T V_n)$. Two normalized pixels are similar when the correlation is high, so we set minimal correlation value as the threshold to match points.

3.3 Get Essential Matrix

To select a set of robust correspondence point pairs from all correspondence points, we use RANdom SAmple Consensus ($RANSAC$) iterative algorithm as follows.

1. Select eight randomly selected correspondence pairs.

2. Estimate essential matrix with eight-point algorithm, then calculate Sampson distances to determine the consensus set.

3. If several Consensus sets of the same size are found, keep this solution; otherwise, back to step 1 if we have iterations left.

After we get the robust correspondent pairs, we can use eight-point algorithm to estimate the essential matrix.

3.4 Extract the Euclidean movement

With eight point algorithm, essential matrix is determined out of scaling. Therefore, there are two possible Euclidean motions (R, T), which can be extracted

from the estimated Essential Matrix E. In total, there are four combinations, but only one is geometric plausible. For all four combinations, we estimate the depth by setting up a linear system and solving it with the singular value decomposition(SVD). The correct movement (R, T) can be determined via the most positive depth condition.