Tran Ngoc Anh - 200154
Nguyen Thu Huyen - 190033
Phan Nguyen Tuong Minh - 190048

Computer Vision, Spring 2023
Prof. Phung Manh Duong

ASSIGNMENT 2 REPORT

Disclaimer: *We know that this is an individual assignment, however, we could not produce a good disparity map at the beginning and needed each other's support to have the results shown in this report. Our individual results were not too different from each other, so we think that it's best that we can merge all our ideas into one report, so that you don't have to read 3 repetitive reports for grading. Thank you so much for your understanding.*

😳Theory

Let the essential matrix $E = R[t]_x$ where R is the rotation matrix and $[t]_x$ is the cross-product of the translation matrix t with any other vector.

We got,

$$E = R[t]_x = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}$$

The two cameras are put in parallel and at the same height, so:

- The rotation matrix R can be presented as the identity matrix I (no rotation)
- $t_2 = 0$

Moreover, the 2 cameras have the same baseline B, so $t_3 = 0$.

We got,

$$E = R[t]_x = I \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -t_1 \\ 0 & t_1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -t_1 \\ 0 & t_1 & 0 \end{bmatrix}$$

In the real world, the projection of point P on the left image and right image can be presented as: $p_l = [u, v, f] \; and \; p_r = [m, n, f]$ (same focal length and baseline)

We got, $p_l^T(E)p_r = 0$

As such,

$$[u \quad v \quad f] \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -t_1 \\ 0 & t_1 & 0 \end{bmatrix} \begin{bmatrix} m \\ n \\ f \end{bmatrix} = [0 \quad ft_1 \quad -vt_1] \begin{bmatrix} m \\ n \\ f \end{bmatrix} = n(ft_1) - v(ft_1) = 0$$

As $f > 0$ and $t_1 > 0$, $n = v$

$\rightarrow p_l$ and p_r are on the same vertical coordinate.

🥱Programming

🥲Part 1: Calibrated (not simple at all) Stereo

Camera Setup

During implementation, we used 2 cameras with the baseline of distance d = 15 cm. The object that we aim to detect is our water bottle. The distance between the water bottle and the camera is around 30 - 40 cm.

For the demo, the distance between the cameras is ~ 10 cm. We run the demo to detect a human with distance from the camera of 50 cm.

Methodology

Our solution for creating the disparity map and retrieving the distance of the object from the camera is as follows:

1. Camera Calibration

We perform camera calibration on both cameras using a set of 93 chessboard images for each one. For each camera, we use OpenCV's calibrateCamera() to get the camera matrix, distortion coefficients, rotation and translation matrix. We also used getOptimalCalibration() to ensure that the errors in undistorting the pixels are minimal, thus the camera parameters are closer to the correct values.

After the calibration, we decided to keep the intrinsic camera parameters the same for two cameras so that to maintain the similarity between two cameras.

The focal length we used for our code is 4.01 cm.

2. Preprocessing images

Before processing the frames to find the necessary parameters for creating disparity, we apply some filters to reduce noise using Gaussian Blur with kernel size = 5 and histogram equalization.

3. Perform transformation on two cameras and calculate the essential and fundamental matrices

The fundamental matrix provides the relationship between two images of the same scene taken by 2 different positions of 2 stereo cameras. It will provide the relationship between the feature correspondence of these stereo images based on the projection of the points from

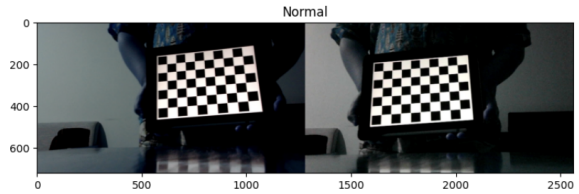
the scene that are captured on both images. In our implementation, we use OpenCV's stereoCalibrate() to perform the transformation and retrieve the fundamental matrix.

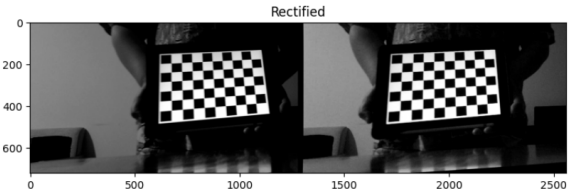
Please find the values for our fundamental matrix and rotation matrix here.

4. Stereo rectification

After getting the fundamental matrices between two images from the stereo camera, we use the camera's intrinsic parameters to undistort the images, then conduct stereo rectification on the images using the rotation matrices.

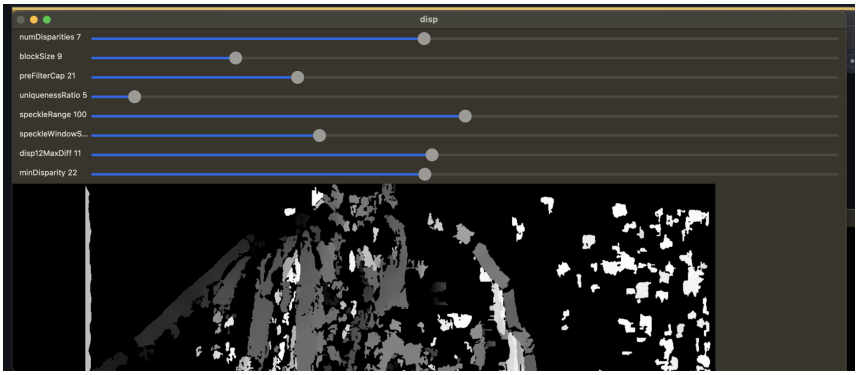
After that, we computed the rectification map for the left and right frames, and applied the mapping on the left and right frames. This is the mapping between the original image pixels and their transformed values after undistorting and rectifying the frames.





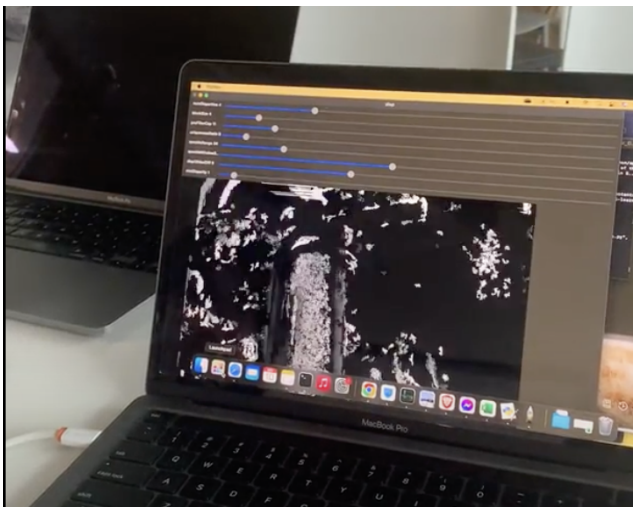
5. Create the disparity map

We use OpenCV's StereoSGBM() to create the disparity map for our detecting object. The parameters we used for tuning the disparity map is shown in the below screenshot:



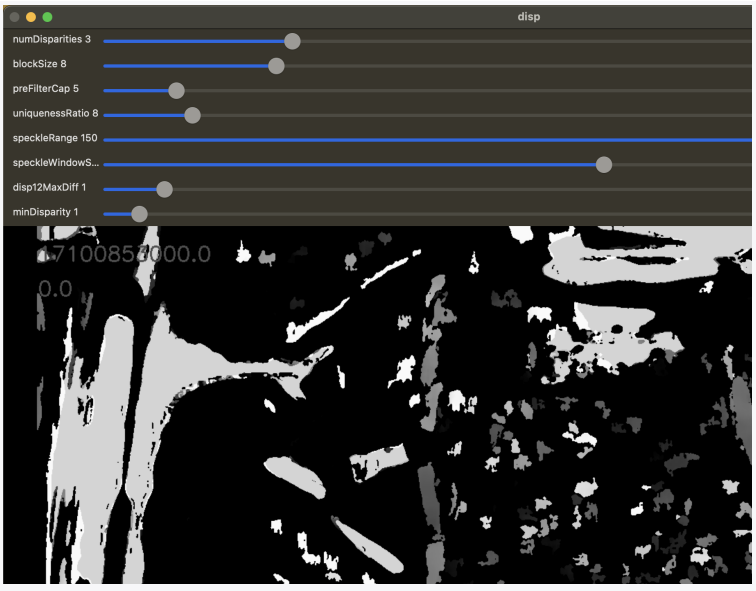
We used a slider GUI referenced here for easier parameter tuning, since the conditions of the testing environment can affect the results of disparity maps significantly.
Our final results for the disparity map with normalized disparity values.

Our testing results can be found here. The test was conducted to show the disparity map of the water bottle. The disparity map seems to work well under normal light conditions (no classroom lighting), and we were able to see the bottle through our output. That said, the results still contain a lot of fuzziness. Please find our results in this Live video.



6. Calculate distance of the object from the camera using the depth map

By choosing one specific pixel in our frame, we generate the distance of the object from the camera using this formula: `depth = (12*focal_length)/(disparity+1e-6).`
Initially, we followed this formula: `Z = B*f / disparity` with B as baseline, and f as focal length. We defined an image point from the output frame to track disparity value for distance estimation. However, both the former and latter version of the equations give fuzzy results, which we could not use to estimate the actual distance.
For example, in our live demo version, even though the parameters were modified on the spot to detect the human/object, disparity changes appear indefinite.

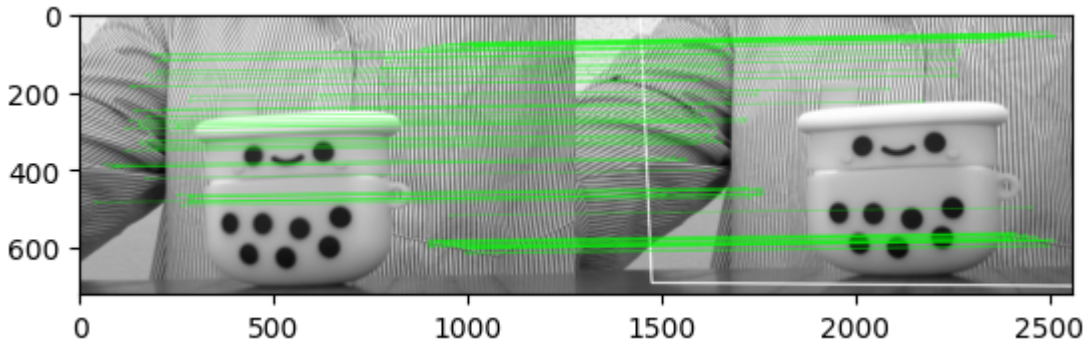


🥹Part 2. Uncalibrated stereo

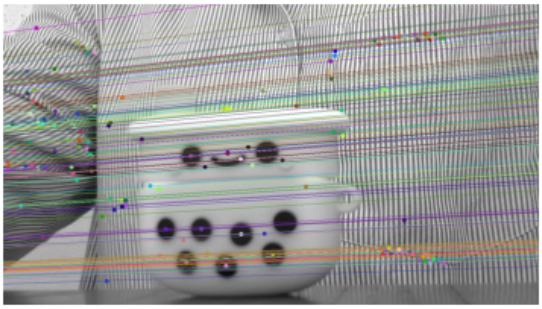
In this part, we took 5 pictures of an object using the same Logitech C270 webcam from different angles. The process to generate a depth map for an uncalibrated stereo system was a replication of guidance steps mentioned in Lecture 9 - Uncalibrated Stereo.

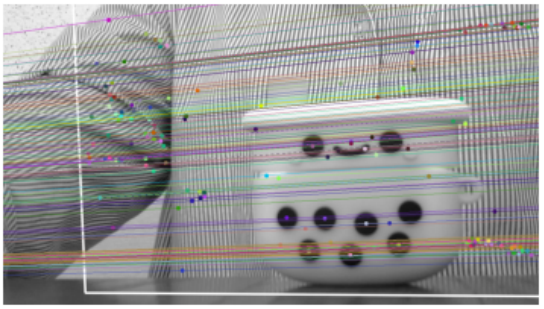
Detailed steps and results are elaborated below:

Step 1: On assumption that the camera matrix is known, we did not execute camera calibration at this step.
Step 2: We used SIFT detector and FLANN matcher to generate correspondence points for the images. Matched keypoints are connected using green lines.



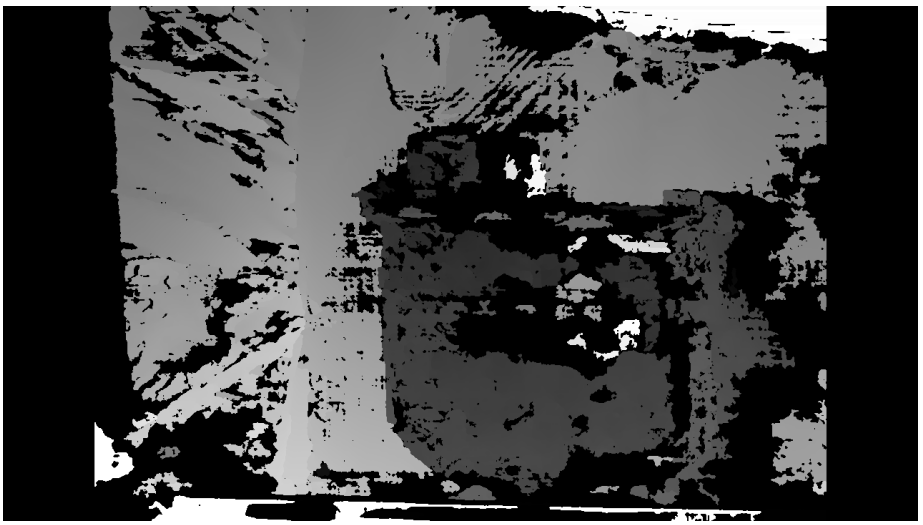
Step 3: We then used the derived key points to calculate the fundamental matrix using openCV's `findFundamentalMat()` function. The corresponding points and fundamental matrix served as inputs to find epilines corresponding to points in both images using `cv2.computeCorrespondEpilines()`.





Step 4: Undistort / rectify images using warpPerspective transformation. Inputs are derived from `stereoRectifyUncalibrated()` function.

Step 4: Rectified images are then used to calculate depth map from a StereoSGBM system and produces output image as below:



In general, we can visually approximate the object shape from the output image, thus temporarily satisfied with the chosen parameters.

Appendix

Source code and image files are to be found in this Drive folder as well as this repository for programming part.