UNIVERSITÀ DI BOLOGNA

School of Engineering

Master Degree in Computer Engineering

Computer Vision and Image Processing M

Stereo Robot Navigation

Professor: **Luigi Di Stefano**

Students:

Lisa Innocenti Uccini

Academic year 2023/2024

Contents

Objective

Given a video sequence taken by a stereo camera mounted on a moving vehicle, projectâs objective is to sense information concerning the space in front of the vehicle which may be deployed by the vehicle navigation system to automatically avoid obstacles.

The input data consist of a pair of synchronized videos taken by a stereo camera (*robotL.avi*, *robotR.avi*), with one video concerning the left view (*robotL.avi*), the other the right view (*robotR.avi*). Moreover, the parameters required to estimate distances from stereo images are provided below:

- focale f = 567.2 pixel
- baseline b = 92.226 mm

The main task of the project requires the following steps:

1. Computing the disparity map in a central area of the reference frame (e.g. a squared area of size 60x60, 80x80 o 100x100 pixels), so to sense distances in the portion of the environment which would be travelled by the vehicle should it keep a straight trajectory.
2. Estimate a main disparity (dmain) for the frontal (wrt the camera) portion of the environment based on the disparity map of the central area of the reference frame computed in the previous step, e.g. by choosing the average disparity or the most frequent disparity within the map.
3. Determine the distance (z, in mm) of the obstacle wrt to the moving vehicle based on the main disparities (in pixel) estimated from each pair of frames:
$$z(mm) = \frac{b(mm) \times f(pixel)}{d_{main}(pixel)}$$
4. Generate a suitable output to convey to the user, in each pair of frame, the information related to the distance (converted in meters) from the camera to the obstacle. Moreover, an alarm should be generated whenever the distance turns out below 0.8 meters.
5. Compute the real dimensions in mm (W,H) of the chessboard pattern present in the scene. Purposely, the OpenCV functions cvFindChessboardCorners and cvDrawChessboardCorners may be deployed to, respectively, find and display the pixel coordinates of the internal corners of the chessboard. Then, assuming the chessboard pattern to be parallel to the image plane of the stereo sensor, the real dimensions of the pattern can be obtained from their pixel dimensions (w,h) by the following formulas:
$$W(mm) = \frac{z(mm) \times w(pixel)}{f(pixel)}$$
$$H(mm) = \frac{z(mm) \times h(pixel)}{f(pixel)}$$

6. Moreover, students should compare the estimated real dimensions to the known ones (125 mm x 178 mm) during the first approach manoeuvre of the vehicle to the pattern, so to verify that accuracy becomes higher as the vehicle gets closer to the pattern. Students should also comment on why accuracy turns out worse during the second approach manoeuvre.

In the following chapters, we will delve into the execution mode of the most important steps among those just listed, particularly focusing on describing the libraries used to achieve objectives.

Introduction

The project can be seen as consisting mainly of two parts:

- the first concerns the calculation of the disparity map in order to determine the z-distance from the obstacle for each video frame. The functions that implement this part are found in the file 'disparityMap.py;
- the second concerns the recognition of the chessboard in each frame and the calculation of its relative dimensions, which will then be compared with the real ones. The functions that implement this part are found in the file 'chessboardRecognition.py'

In the following two chapters, both parts of the project will be described in more detail, with particular attention to the description of the work carried out and the most critical steps.
The results section aims to present the obtained results with the help of graphs generated by executing the Python code.

Disparity map

For each frame of the videos, it is required to compute the disparity map in a central area of the reference system. Using the OpenCV library in Python, we can efficiently and accurately implement this process.

OpenCV (Open Source Computer Vision) is an open-source library for computer vision and image processing. It provides a range of pre-trained models and algorithms that can be employed for common computer vision tasks.

To begin, it is necessary to initialize the stereo disparity calculator using the StereoBM_create function of OpenCV. During this initialization, two fundamental parameters are specified:

- numDisparities, which indicates the maximum number of disparities expected to be detected. This parameter is determined by the project specifications;
- blockSize, which determines the size of the block used to compute disparity. This value was chosen considering the constraint imposed by the library, which requires it to be an odd number less than or equal to 99. The chosen value aims to produce a disparity map that is as smooth as possible, free of noise and spurious peaks.

Once the stereo disparity calculator is initialized, we can proceed with the actual computation of the disparity map using the compute function, passing stereo images as arguments, where these images represent respectively the left and right frames.

```
stereoMatcher = cv2.StereoBM_create(numDisparities=128, blockSize=83)
disparity = stereoMatcher.compute(imgL, imgR)
```

After obtaining the disparity map, we can select only the central frames of interest for further processing. To do this, we calculate the coordinates of the central point of the image and define an interval around this point. Finally, we extract only the region of interest from the disparity map, thus allowing for a more focused and detailed representation depth of the scene.

This procedure enables us to obtain an accurate and high-quality disparity map, providing a solid basis for detecting distances in the portion of the environment that would be traversed by the vehicle if it were to maintain a straight trajectory.

Before applying the function with the provided constants (focal length and baseline) to calculate the distance (z, in mm) of the obstacle relative to the vehicle, it is necessary to estimate the main disparities (in pixels) for each frame pair.

For the calculation of the main disparity, it was chosen to use the mean instead of, for example, the median or the most frequent disparity, as it allowed for more precise and accurate results.

Dimension chessboard

The process of chessboard recognition using the OpenCV functions cvFindChessboardCorners and cvDrawChessboardCorners is necessary as it allows us to estimate the dimensions of the chessboard assuming it is parallel to the plane of the stereo sensor image.

Firstly, utilizing the cvFindChessboardCorners function, we attempt to locate the internal corners of the chessboard in the current image. If the function successfully returns, meaning if a match has been found, we proceed with drawing the borders of the chessboard using the cvDrawChessboardCorners function. This enables us to visually display the coordinates of the internal points of the chessboard, facilitating the process of dimension calculation.

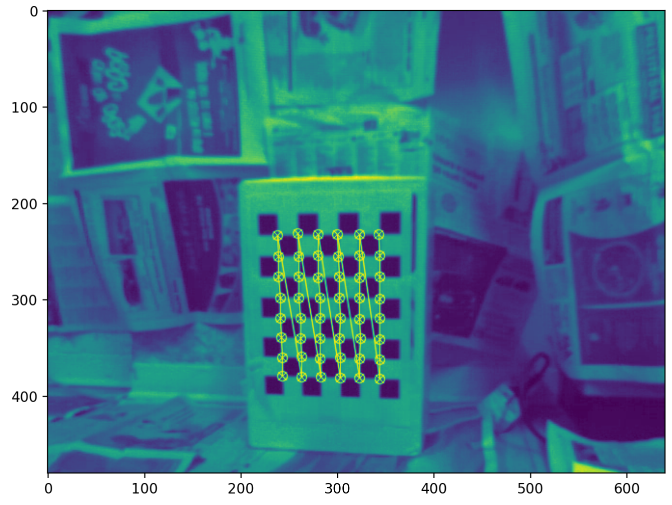

Figure 1: Displaying the internal chessboard points of a frame

Subsequently, we compute the dimensions of the chessboard in terms of pixels using the identified corner coordinates. By comparing the coordinates of the extreme corners of the chessboard, we can determine the width (w) and height (h) of the chessboard in pixel.

```
ret,corners = cv2.findChessboardCorners(img ,c.pattern_size)
if ret == True:
    cv2.drawChessboardCorners(img, c.pattern_size, corners, ret)

    #compute pixel dimention of chessboard
    w = abs(corners[c.NUM_H*(c.NUM_W-1)][0][0]-corners[0][0][0])
    h = abs(corners[c.NUM_H-1][0][1]-corners[0][0][1])
```

Starting from the width and height dimensions in pixels of the chessboard recognized within the video frame under consideration, the corresponding dimensions in millimeters are calculated. This is done by taking into account the previously calculated z-distance for the given image and the focal length of the camera.

This process, implemented through the provided Python code, yields an accurate estimation of the chessboard dimensions, which can be compared with the actual dimensions to verify its precision.

There is a possibility that the findChessboardCorners function may return a value ret equal to False. In this case, it means that the chessboard pattern was not recognized in the frame. Consequently, for such frames, the estimation of w and h will not be performed.

Results

Once the entire algorithm described in the preceding paragraphs has been executed, the results obtained are presented through a series of illustrative graphs. These graphs provide a visual representation of the main metrics calculated during the processing of each frame of the input videos. They share the same x-axis, which represents all the frames of the video.

The first graph, labeled "dMain," depicts the average disparity for each frame of the video. This value gives an indication of the relative distance of the object from the stereo sensor, as the disparity is an indicator of depth.

Subsequently, a graph illustrates the distance z in millimeters of the obstacle for each frame of the video. This data is obtained by applying the provided constants, such as focal length and baseline, to the disparity calculation.

Another graph represents the presence or absence of an alarm for each frame. This alarm is triggered if the obstacle is detected at a distance closer than 0.8 meters from the robot, identified by a flag of 0 or 1.

Finally, the last graph illustrates the variance between the estimated size (base multiplied by height, W*H) of the chessboard and the actual size (provided in the specifications) in millimeters for each frame of the video. This graph allows for an evaluation of the algorithm's accuracy in recognizing and estimating the dimensions of the chessboard compared to the real ones.

In this way, through a visual representation of the obtained results, it is possible to assess the effectiveness and accuracy of the algorithm implemented in stereo image processing.

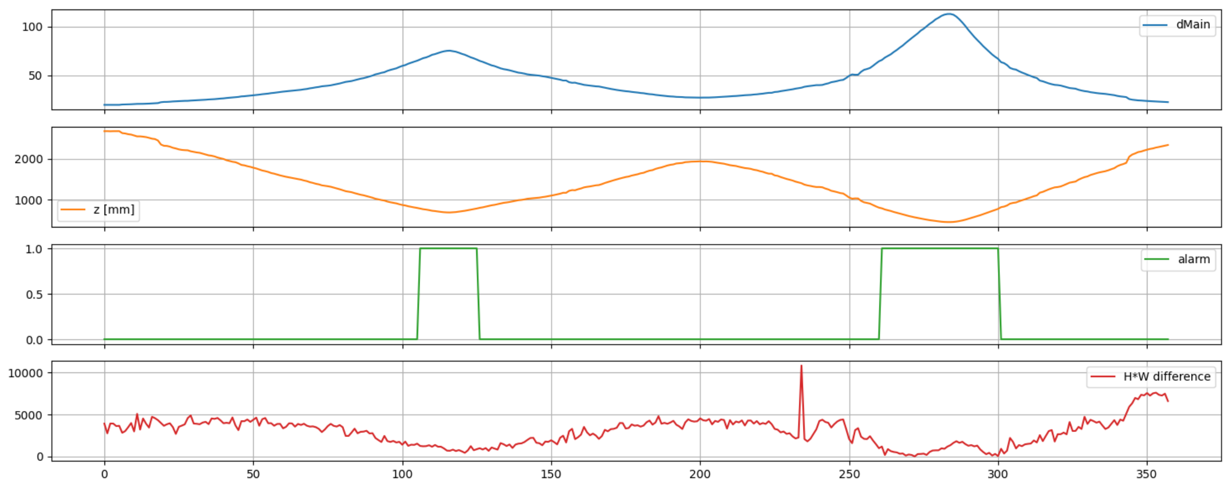

Figure 2: Results Plot

Conclusions

The analysis of the results obtained highlights the success of both parts of the project.

Initially, the calculation of the disparity map and the subsequent determination of the z-distance from obstacles yielded consistent and reliable results. The graph includes frames in which the robot approaches the obstacle beyond a certain threshold, highlighting two time intervals during which the alarm is activated.

Regarding the recognition of the chessboard and the calculation of its dimensions, positive results were also obtained. In particular, it was observed that as the robot approaches the chessboard, the discrepancy between the real and calculated dimensions decreases significantly.

It's interesting to note that the precision of the estimation is higher during the first approach phase compared to the second, as highlighted in the last graph provided. This phenomenon can be attributed to several factors. During the second approach phase, as the camera further approaches the chessboard, there may be greater perspective distortion negatively affecting the accuracy of the dimension estimation. Additionally, camera movement can play a significant role: the more uniform and stable the approach movement, the more accurate the dimension estimation.
For example, during the first approach phase, the movement may be smoother and less affected by vibrations or oscillations, allowing the camera to capture sharper and more detailed images of the chessboard. Conversely, during the second phase, the additional approach may cause more camera shaking and increased perspective distortion, thereby compromising the accuracy of the dimension estimation.

We can conclude by saying that the results obtained and presented are more than satisfactory and meet all the requirements of the project. Furthermore, they provide a solid foundation for further developments and future improvements.