Depth Perception using Binocular Stereo

EENG 5640 Computer Vision Final Project

University of North TexasCollege of Engineering

Denton, TX

December 2nd,2024

Kavya Sree Maddikara

Joshua Sannareddy

Sai Sandesh Reddy Manda

Instructor: Guturu Parthasarathy

CONTENTS

I Introduction

II Functional Specifications

III Requirement Specifications

IV Design

V Implementation

VI Testing

VII Conclusion

I. INTRODUCTION

The two-dimensional array of pixels used to make a picture lacks the essential information of depth because every object or scene that is photographed is three-dimensional. Because knowing depth would allow for a wide range of robotics and artificial vision applications, a myriad of approaches and strategies have been developed to acquire or recreate depth.

The basic geometric characteristic that depth can be accurately ascertained from several representations is the foundation of all these techniques. In the case of many images taken from different perspectives of the same object or scene, we can establish the depth by cross-referencing the locations of each point or group of points in each image. Thus, object and edge recognition is an important initial step since points must be recognized as identical before their locations in each sample can be ascertained. It takes a lot of computation to solve the correspondence problem.

Luckily, our approach only calls for two sites to be used for image acquisition. The technique known as binocular stereo vision was created after it was discovered that animals with two eyes can perceive depth and distance quickly, precisely, and effortlessly. We'll discuss this in more detail later, but for now, let's assume that our algorithm uses two picture inputs to create a third image. A map of the depths throughout the entire image will be produced by combining the two inputs to create this third image.

II. FUNCTIONAL SPECIFICATIONS

To build a stereo vision system, we designed a Python script that can compute depth maps from any type of image, whether it was taken from nature or man-made sources. We expect this approach to work best with images taken from two locations with the same horizontal height of epipolar lines and with sufficient illumination. We also presume that there is no obscuration of the item. Naturally, if there are pixels missing from one image because of occlusion, the correspondence will not be fully realized, and we may expect that the software will not function as effectively in low light. Pixel blocks are usually used to create a comparable item or form.

III. REQUIREMENT SPECIFICATIONS

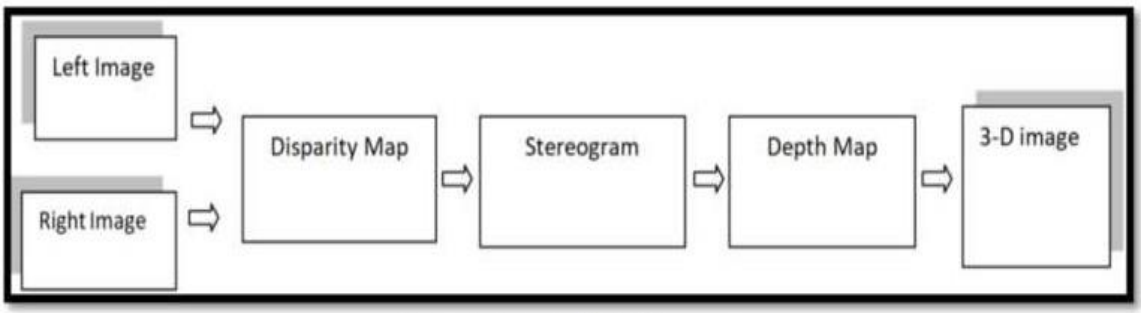
The list of requirements that our system must fulfill is displayed in the table below.

Requirement	Description
Disparity Map	Develop a function that calculates the difference between two input pictures.
Stereogram	Develop a function that uses the two inputs to produce a new image.
Depth Map	Make a function that uses the disparity table to create a depth map.

IV. DESIGN

To gain a better understanding of the fundamental flow of our software, the stereogram, depth map, and disparity map can be separated into three pieces. This is the sole mathematical calculation our program does. To develop the code and enable it to process inputs and display outputs correctly, additional data manipulation is required. Importing the required libraries is the first step in our procedure. Although NumPy is a given, the cv2 library will handle the most of the work because we can create these maps using its APIs. We can then construct the disparity map function using the left and right images, the x-axis variations between key places, and the size number of pixels we want the blocks to consist of.

The creation of an object that corresponds with blocks for the algorithm is the following stage. The stereo relationship between the left and right images can be determined by utilizing this item. Next, the function that creates a stereogram from an image will be constructed. We create a grayscale intensity matrix with the same dimensions as the input image after determining its form. The depth map function is then constructed using the disparity map as input and the distance between the two capture points as input. This function reads the pictures and creates the depth map.



Binocular Stereo Block Diagram

V. IMPLEMENTATION

We chose Python as the coding language to accomplish our plan. Below is the Python code for Binocular Stereo and Disparity.

```
from __future__ import print_function
import numpy as np
import cv2
ply_header = '''ply    '''
def write_ply(fn, verts, colors):
    verts = verts.reshape(-1,3)
    colors = colors.reshape(-1,3)
    verts =np.hstack([verts,colors])
    with open(fn,'wb') as f:
        f.write((ply_header % dict(vert_num=len(verts))).encode('utf-8'))
        np.savetxt(f,verts,fmt='%f %f %f %d %d %d')

if __name__=='__main__':
    print('loading images...')
    imgL = cv2.pyrDown(cv2.imread('im0.png',0))
    imgR = cv2.pyrDown(cv2.imread('im1.png',0))

    window_size=3
    min_disp =16
    num_disp =112-min_disp
    stereo = cv2.StereoSGBM_create(minDisparity = min_disp,
                                   numDisparities =num_disp,
                                   blockSize =16,
                                   P1= 8*3*window_size**2,
                                   P2 = 32*3*window_size**2,
                                   disp12MaxDiff =1,
                                   uniquenessRatio =10,
                                   speckleWindowSize =100,
                                   speckleRange =32)
    print('computing disparity....')
    disp = stereo.compute(imgL,imgR).astype(np.float32)/16.0
    print('generating 3d point cloud.....',)
    h,w =imgL.shape[:2]
    f=0.8*w
```

```
    f=0.8*w
    Q=np.float32([[1, 0, 0, -0.5*w],
                  [0, -1, 0, 0.5*h],
                  [0, 0, 0,   -f],
                  [0, 0, 0,   0]])
    points = cv2.reprojectImageTo3D(disp, Q)
    colors = cv2.cvtColor(imgL, cv2.COLOR_BGR2RGB)
    mask = disp>disp.min()
    out_points = points[mask]
    out_colors = colors[mask]
    out_fn ='out,ply'
    write_ply('out.ply', out_points, out_colors)
    print('%s saved' % 'out.ply')
```

VI. TESTING

To ensure that the code would generate an accurate Binocular Stereo Map, we had to test it on a range of cases. We decided to both take our own photos and find images online in order to fully test our application. Many examples of the code under test are provided below. The left and right images as well as the differences between the two pictures are included in these tests.

As noted earlier, two of the images were from the online Middlebury collection. Both the pipe and the computer examples fall under this. In order to test the code, more test photos were taken, including the art room, ladder, and chess table. Even though the results from the Middlebury dataset were more accurate than those from these tailored shots, the photographs' data nevertheless shows binocular stereo discrepancy. It is possible to attribute this lack of accuracy to the lack of a stereoscopic camera. Accordingly, there was a greater chance of inaccuracy depending on minor movements throughout the photo shoo

Example 2: Ladder Images

Ladder Left Table Image

Ladder Right Table Image

Ladder Image Disparity

VII. Conclusion

Every one of our test situations shows that our algorithm was unable to produce precise depth maps if the two images were taken too close to one another. Naturally, our depth map would be useless if we used the same image as the input for left and right since no depth could be seen. Since the disparity map would not identify any disparity at all, the depth function would not get any input, and the software would most likely generate error codes. The file format may also have a substantial impact on the outcomes because all inputs must first be compressed to meet the size specifications of our approaches. A good depth estimation algorithm would therefore need to be adjusted with all of these factors (and hardware) in mind, accounting for different aspect ratios, image resolutions, and cameras. Although our program was designed to operate with online stock photos, we spent much of the testing using mobile phone photos, which resulted in damage to the depth maps.

All things considered, we have demonstrated that 2-dimensional images can still retain some depth information. As robots advances, it will be crucial to accomplish this in a much more efficient manner so that systems may handle other tasks besides depth sensing with less time, energy, and computer resources.

REFERENCES: -

"Middlebury Stereo Datasets," Middlebury Vision,

https://vision.middlebury.edu/stereo/data/

Brandt, R., Strisciuglio, N., Petkov, N., amp; Wilkinson, M. H. F. (2020, February 20). Efficient binocular stereo correspondence matching with 1-D Max-Trees. Pattern Recognition Letters.

https://www.sciencedirect.com/science/article/pii/S0167865520300581.