PS #7 Report

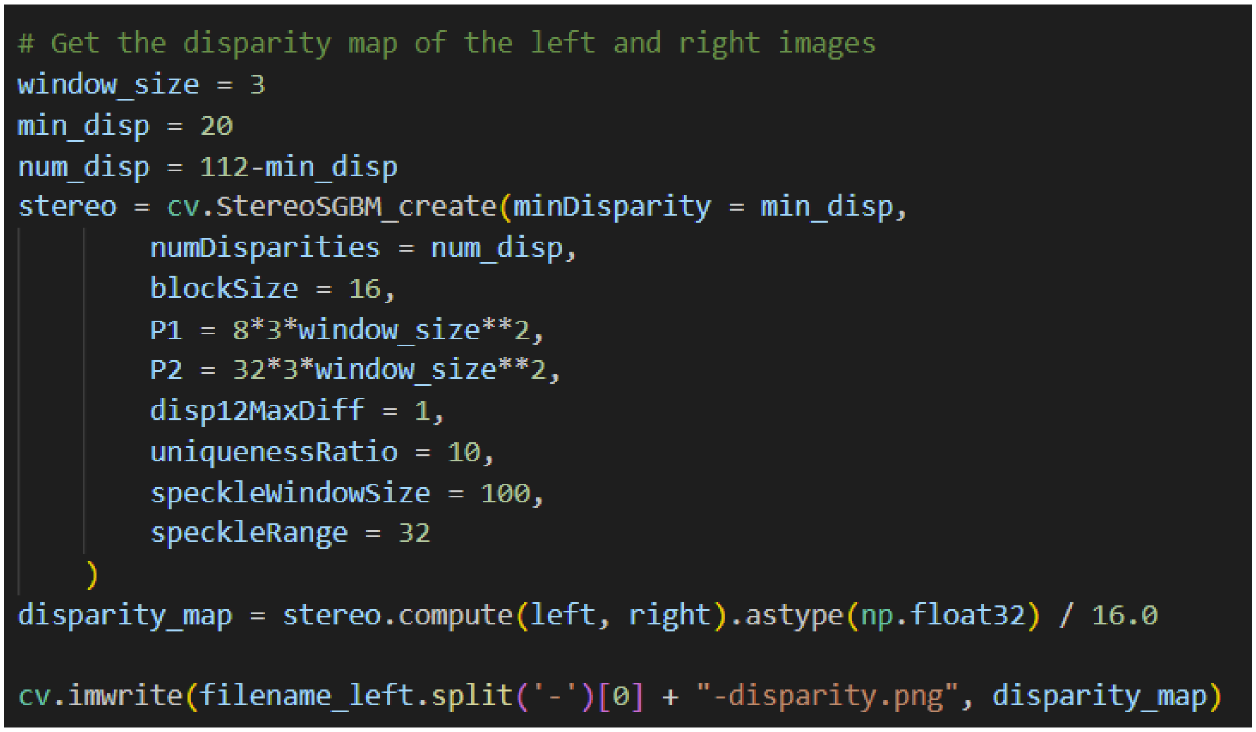
Problem 1: Depth Mapping and 3D Reconstruction

Methods

Parallel stereo images allow depth information to be extracted from images. The first step in this process is to find the disparity map between the left and right images. To do this, semi-global block matching was used. The cv.SGBM.create() method and stereo.compute() method generate a grayscale disparity map between the left and right images. The parameters used were as follows:

Parameter	Value
minDisparities	20
numDisparities	92
blockSize	16
speckleWindowSize	100
speckleRange	32

Table 1: Parameters used in the semi-global block matching method to generate the disparity map.



```python
# Get the disparity map of the left and right images
window_size = 3
min_disp = 20
num_disp = 112-min_disp
stereo = cv.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
    )
disparity_map = stereo.compute(left, right).astype(np.float32) / 16.0

cv.imwrite(filename_left.split('-')[0] + "-disparity.png", disparity_map)
```

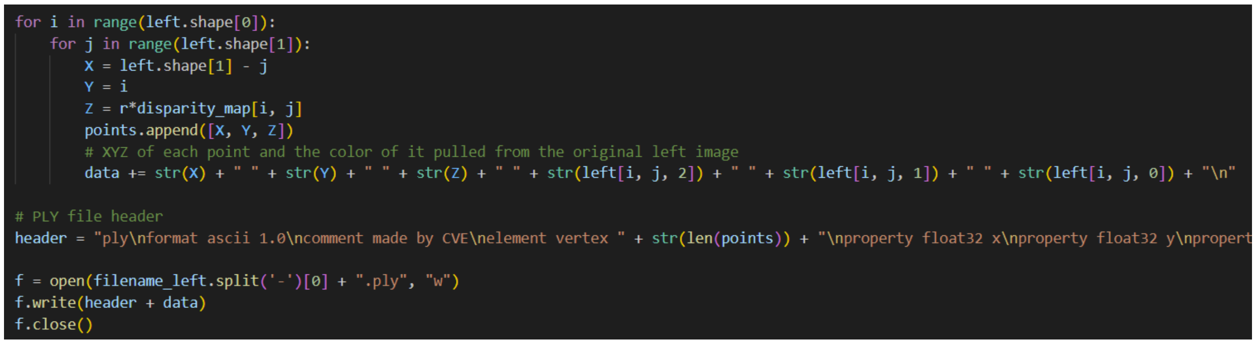
Figure 1: Implementation of the generation of the disparity map.

Once the disparity map was generated, the values in it were used to find the depth of each point in the images. The depth of the point is inversely proportional to the disparity value. The constant of proportionality depends on the camera baseline distance and focal length, but since those were not known for these images, the constant of proportionality was found experimentally. Using the bowling ball image and making the ball roughly spherical, the constant was found to be 5.

To generate the .ply file, a string was generated for each point formatted as such:

```
x y z r g b
```

Concatenating each of these strings with the .ply header information yielded the final 3D reconstruction of each scene from its respective pair of images.



```python
for i in range(left.shape[0]):
    for j in range(left.shape[1]):
        X = left.shape[1] - j
        Y = i
        Z = r*disparity_map[i, j]
        points.append([X, Y, Z])
        # XYZ of each point and the color of it pulled from the original left image
        data += str(X) + " " + str(Y) + " " + str(Z) + " " + str(left[i, j, 2]) + " " + str(left[i, j, 1]) + " " + str(left[i, j, 0]) + "\n"

# PLY file header
header = "ply\nformat ascii 1.0\ncomment made by CVE\nelement vertex " + str(len(points)) + "\nproperty float32 x\nproperty float32 y\npropert

f = open(filename_left.split('-')[0] + ".ply", "w")
f.write(header + data)
f.close()
```

Figure 2: Generation of the .ply 3D reconstruction of the scene.

Results

The following disparity maps were generated from the 3 given pairs of images:

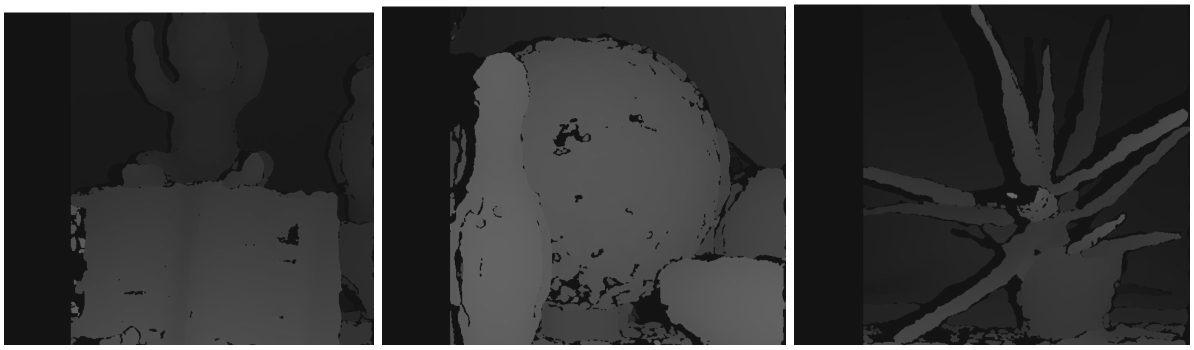


Figure 3: Disparity maps of the given pairs of images.

These disparity maps came out pretty nicely. The bowling ball had some artifacts on the face of the ball, likely due to lighting and reflection. This could probably be smoothed out with some image preprocessing or a larger window size in the SGBM.

The disparity map for the images I took did not turn out well, no matter what parameters I tried to use.

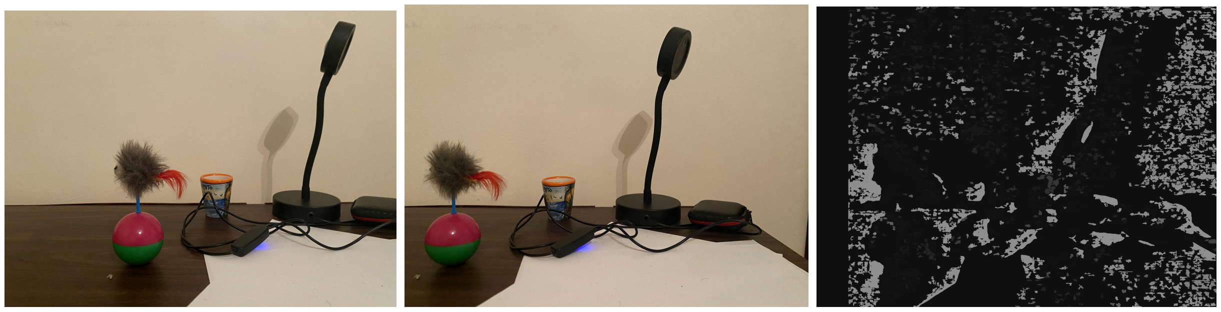


Figure 4: Left, right, and disparity images taken by me.

For the original 3 pairs of images, the following 3D reconstructed images were made.

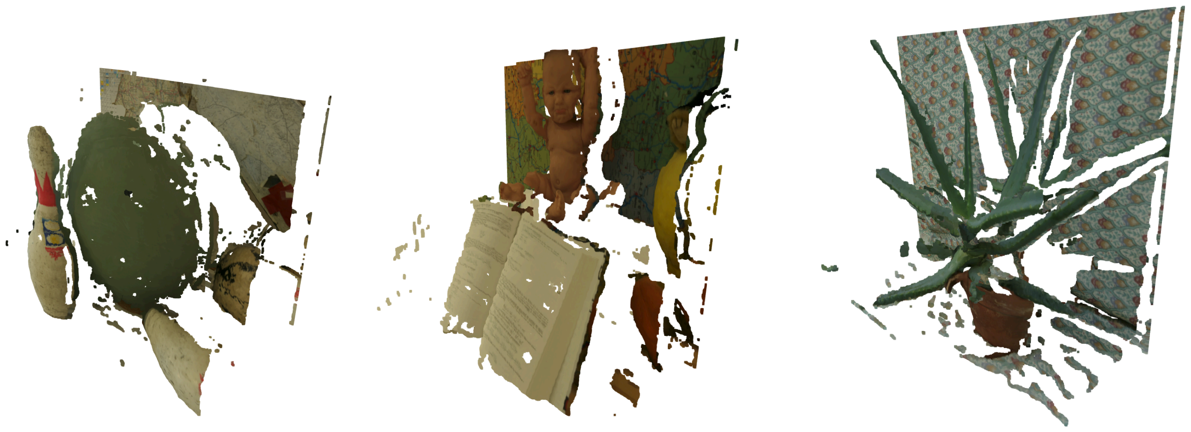


Figure 5: The .ply 3D reconstructions of each of the original 3 pairs of images.

Due to the poor disparity map of the image I took, the 3D reconstruction is completely incoherent.