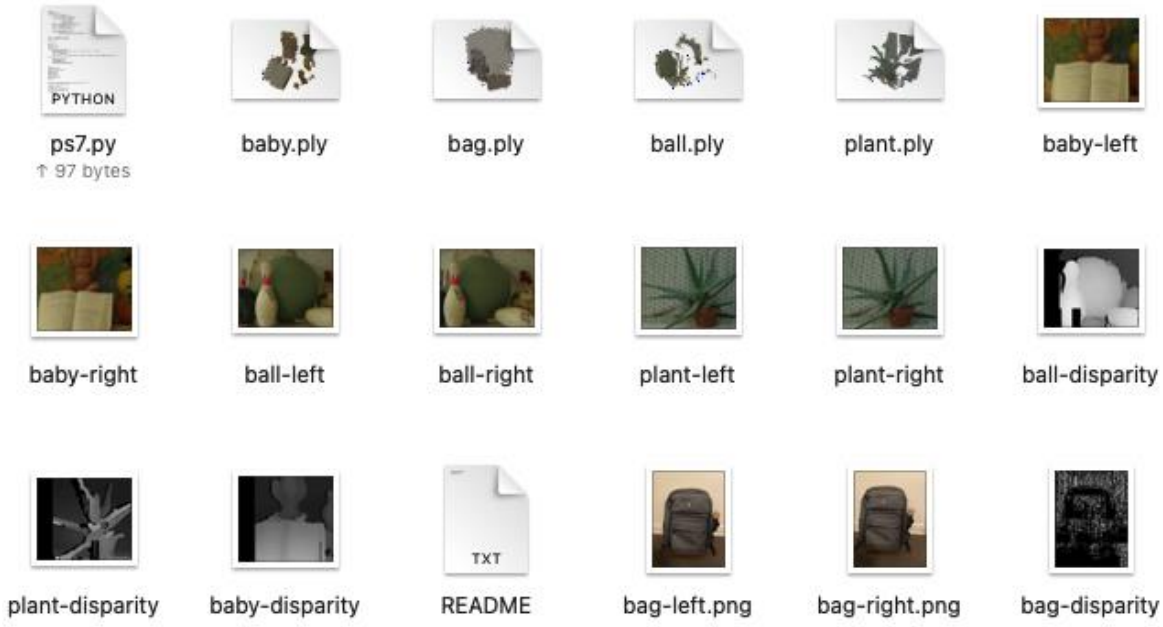
PS7 Report

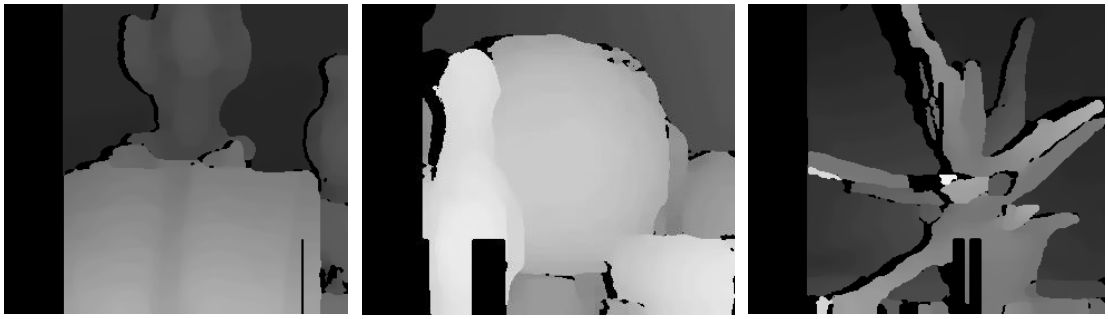



```
img_l = cv2.imread("ball-left.png")
img_r = cv2.imread("ball-right.png")

min_disp = 5
num_disp = 3*16

#min_disp = 8
#num_disp = 4*16

window_size = 8
left_matcher = cv2.StereoSGBM_create(
        minDisparity=min_disp,
        numDisparities=num_disp,  # max_disp has to be dividable by 16 f. E. HH 192, 256
        blockSize=2*window_size,
        P1=8 * 3 * window_size**2,
        # wsize default 3; 5; 7 for SGBM reduced size image; 15 for SGBM full size image (1300px and above); 5 Works nicely
        P2=32 * 3 * window_size**2,
        disp12MaxDiff=1,
        uniquenessRatio=10,
        speckleWindowSize=100,
        speckleRange=1,
        mode=cv2.StereoSGBM_MODE_SGBM_3WAY
    )
```

In this code, we need to use cv2.StereoSGBM_create function to create a left image matcher. By adjusting input arguments such as minimum disparity, number of disparities, block size, P1, P2, etc., I found the best values to construct the disparity image and polygon file. For the image, I found the best windows size is 8, minimum disparity is 5 and number of disparity is 48, which need to be dividable by 16.

```
disp = left_matcher.compute(img_l, img_r).astype(np.float32) / 16.0
output_img = (disp-min_disp)/num_disp
```

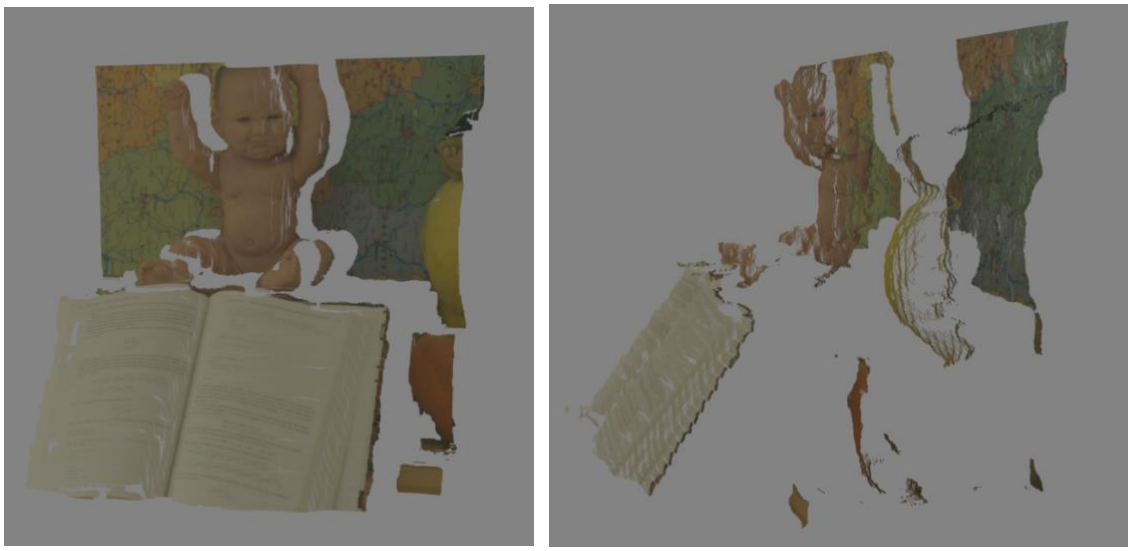
Then the Python will compute the disparity by left_matcher.compute command for you and we need to divide it by 16. Then for the output image, we need to minus minimum disparity and divide it by number of disparities to get our output image.



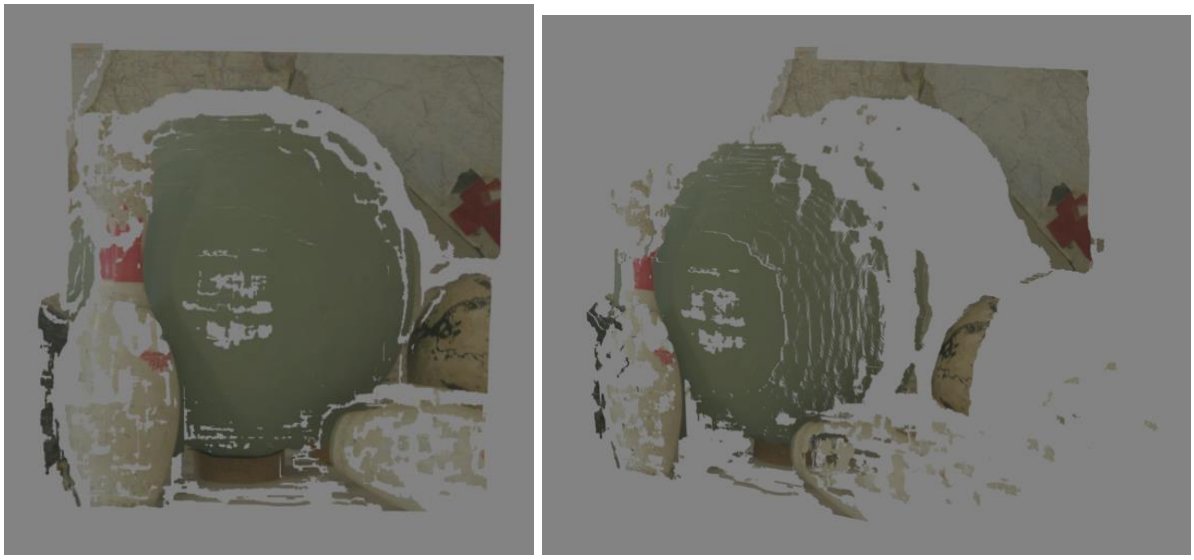
The disparity image shows the disparity in an intuitive way in grayscale

```
def depth_map(disp, frame_name, img_l):
        h, w = img_l.shape[:2]
        f = 0.8 * w  # guess the focal length
        Q = np.float32([[1, 0, 0, -0.5 * w],
                        [0, -1, 0, 0.5 * h],
                        [0, 0, 0, -3 * f],
                        [0, 0, 1, 100]])
        points = cv2.reprojectImageTo3D(disp, Q)
        colors = cv2.cvtColor(img_l, cv2.COLOR_BGR2RGB)
        mask = disp > disp.min()
        points = points[mask].reshape(-1, 3)
        colors = colors[mask].reshape(-1, 3)

        points = np.hstack([points, colors])
        with open(frame_name, 'wb') as f:
                f.write((ply_header % dict(vert_num=len(points))).encode('utf-8'))
                np.savetxt(f, points, fmt='%f %f %f %d %d %d ')
```

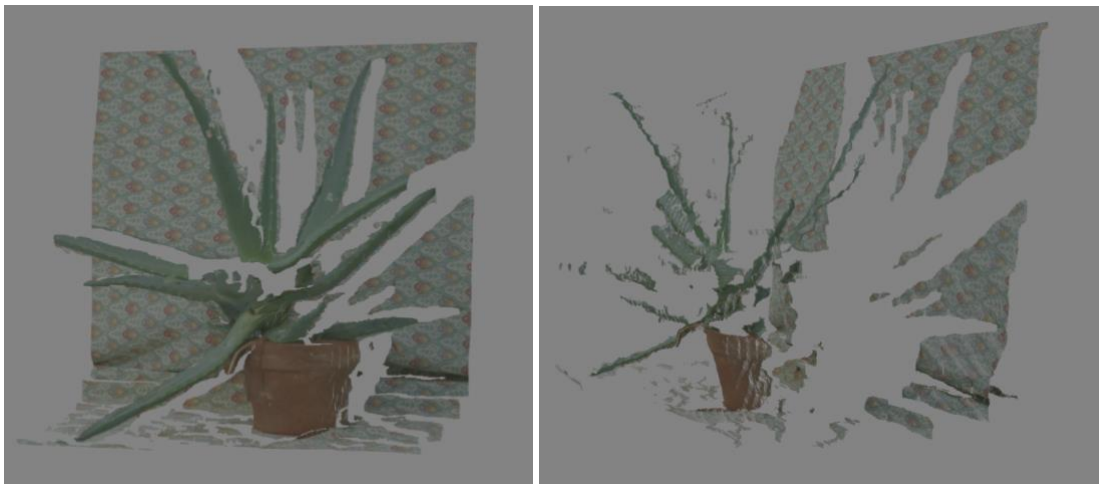
To write ply file, I create a function depth_map. It takes disp, which compute by cv2.StereoSGBM_create, frame name, which define by ourselves, and the base image, which is the left image of our image pair. Then we need to retrieve the height and width of the image and also approximate the focal length. The Q is 4×4 perspective transformation matrix. By using cv2.reprojectImageTo3D function, it reprojects a disparity image to 3D space. Then we can stack the points information with color information to write our ply file.



The baby polygon file shows good depth information



The ball polygon file shows good depth information



The plant polygon file shows good depth information

I used iPhone 12 Pro to shoot a pair of images of my bag. However, because we don't know the parameter of the camera. It is very difficult to tune the parameters. The disparity image looks not so great.

For the polygon file, the porjection points are very scattering, which make the ply file very bulrry to see. In addition, the depth inforamtion and the relationship between bag and wall are not correct.