CS4186 COMP VISION & IMAGE PROCESSING

Assignment 2 Report

CHAN Wai Hong
56272737

Table of Contents

Table of Figure

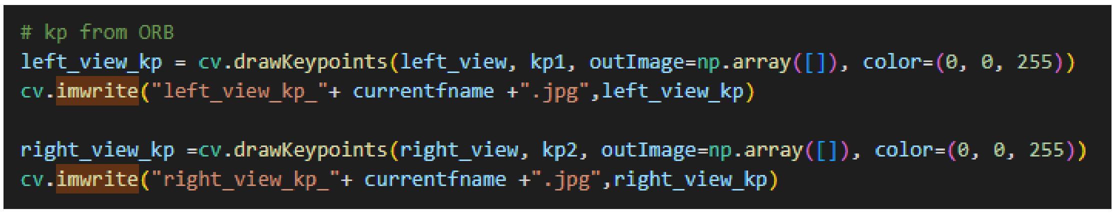
Introduction

This report aims to description how to achieve disparity (depth) map for given pair of left view and right view. There are three pairs “Art”, “Dolls” and “Reindeer”

Image Rectification

Feature Extracting

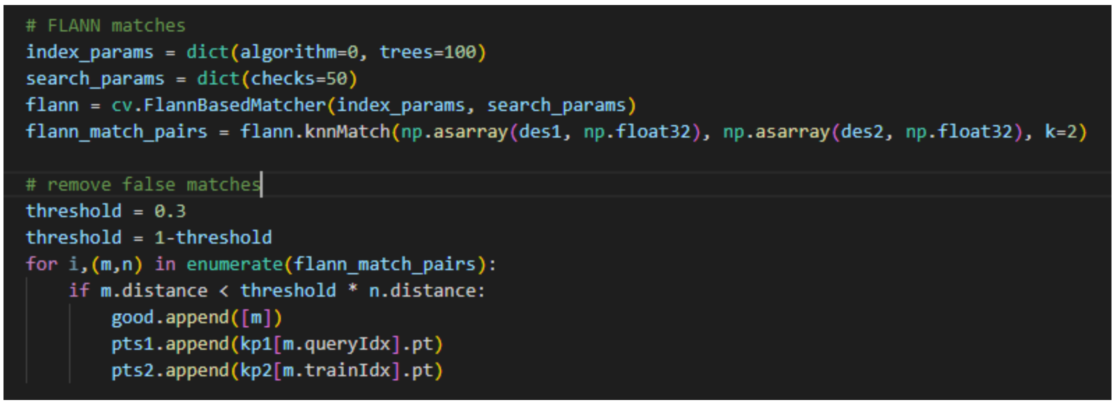
The ORB (Oriented FAST and Rotated BRIEF) is applied for matching keypoint from the OpenCV library.



```
# kp from ORB
left_view_kp = cv.drawKeypoints(left_view, kp1, outImage=np.array([]), color=(0, 0, 255))
cv.imwrite("left_view_kp_"+ currentfname +".jpg",left_view_kp)

right_view_kp =cv.drawKeypoints(right_view, kp2, outImage=np.array([]), color=(0, 0, 255))
cv.imwrite("right_view_kp_"+ currentfname +".jpg",right_view_kp)
```

Apply FLANN for kNN (k-nearest neighbours) search and set threshold for good matches filtering:



```
# FLANN matches
index_params = dict(algorithm=0, trees=100)
search_params = dict(checks=50)
flann = cv.FlannBasedMatcher(index_params, search_params)
flann_match_pairs = flann.knnMatch(np.asarray(des1, np.float32), np.asarray(des2, np.float32), k=2)

# remove false matches
threshold = 0.3
threshold = 1-threshold
for i,(m,n) in enumerate(flann_match_pairs):
    if m.distance < threshold * n.distance:
        good.append([m])
        pts1.append(kp1[m.queryIdx].pt)
        pts2.append(kp2[m.trainIdx].pt)
```

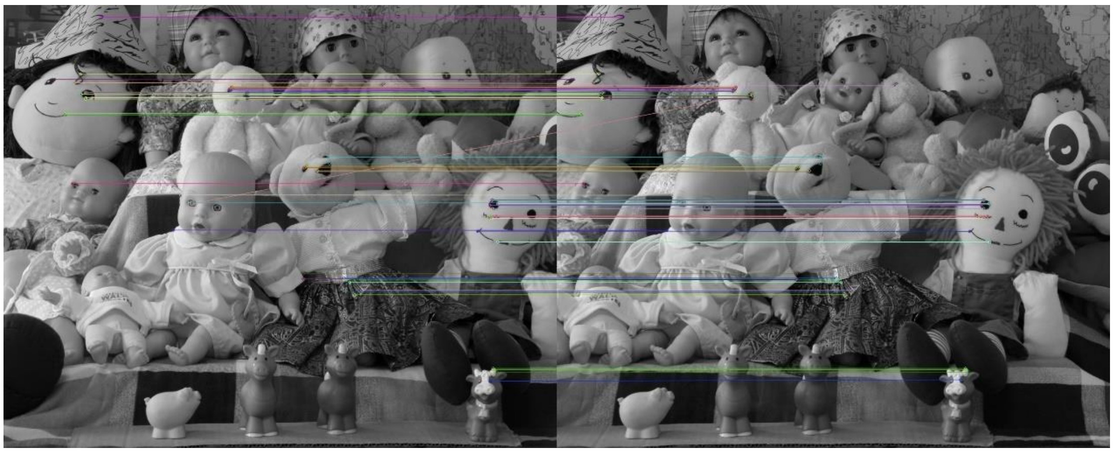
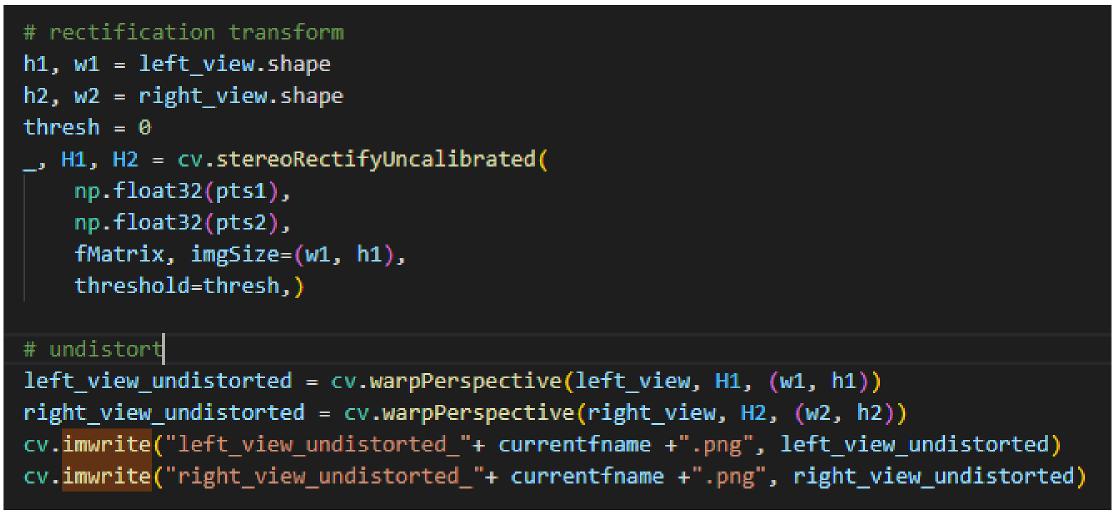


Figure 1 matches of Dolls

Rectification Transform

The perspective will be warped to vertically aligned which only horizontal motion need to be cared. Since the given image already rectified, this step may not be necessary for this task, but is an important step for forming stereo depth map.

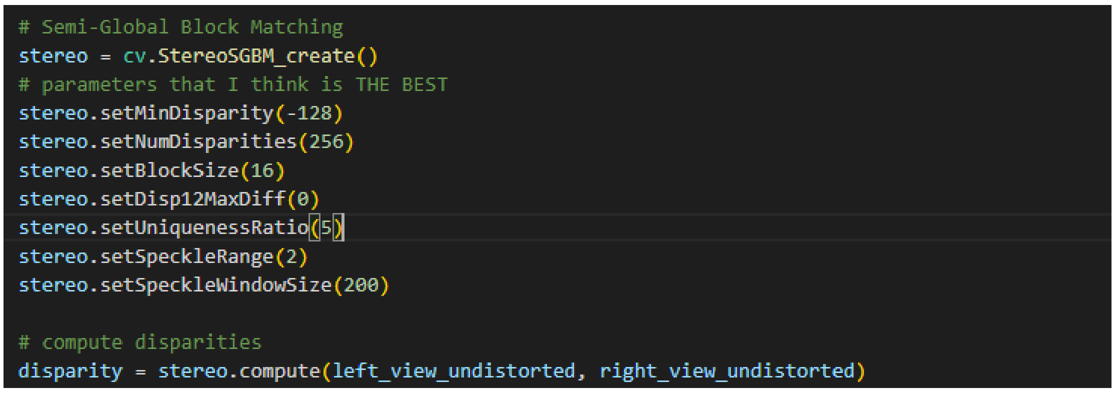


```python
# rectification transform
h1, w1 = left_view.shape
h2, w2 = right_view.shape
thresh = 0
_, H1, H2 = cv.stereoRectifyUncalibrated(
    np.float32(pts1),
    np.float32(pts2),
    fMatrix, imgSize=(w1, h1),
    threshold=thresh,)

# undistort
left_view_undistorted = cv.warpPerspective(left_view, H1, (w1, h1))
right_view_undistorted = cv.warpPerspective(right_view, H2, (w2, h2))
cv.imwrite("left_view_undistorted_"+ currentfname +".png", left_view_undistorted)
cv.imwrite("right_view_undistorted_"+ currentfname +".png", right_view_undistorted)
```

Semi-Global Block Matching

The SGBM (Semi-global matching) is applied for the calculate the disparity. Horizontal gradient filtering will be processing block by block during SGBM and it is called SobelX. The below show the parameter that I tested to receive a good result. The number of disparities refer to the total depth of the mapped 3D space and the block size is the windows size for each block calculation.

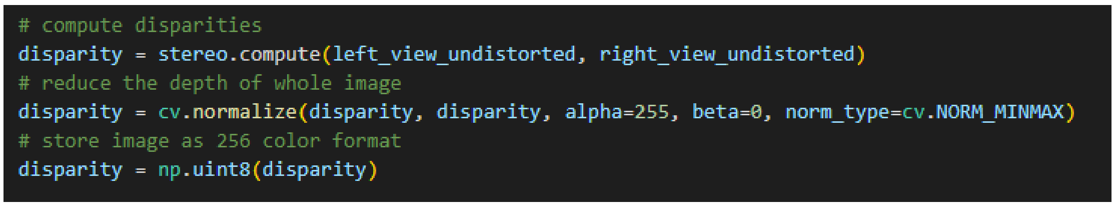


```python
# Semi-Global Block Matching
stereo = cv.StereoSGBM_create()
# parameters that I think is THE BEST
stereo.setMinDisparity(-128)
stereo.setNumDisparities(256)
stereo.setBlockSize(16)
stereo.setDisp12MaxDiff(0)
stereo.setUniquenessRatio(5)
stereo.setSpeckleRange(2)
stereo.setSpeckleWindowSize(200)

# compute disparities
disparity = stereo.compute(left_view_undistorted, right_view_undistorted)
```

Post Processing

The result is normalized and stored as 256 colour images.



```python
# compute disparities
disparity = stereo.compute(left_view_undistorted, right_view_undistorted)
# reduce the depth of whole image
disparity = cv.normalize(disparity, disparity, alpha=255, beta=0, norm_type=cv.NORM_MINMAX)
# store image as 256 color format
disparity = np.uint8(disparity)
```

Result

Art

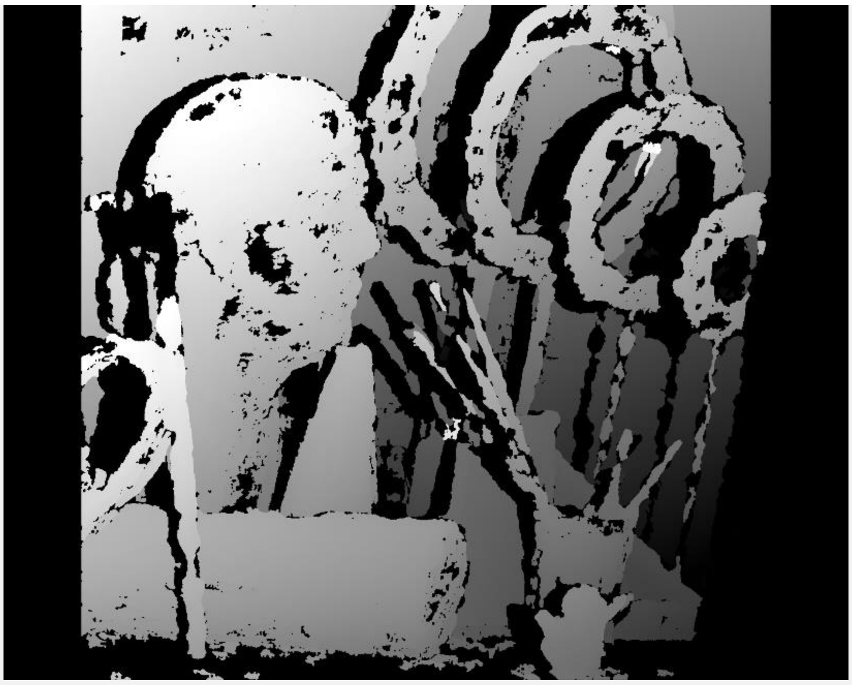


Figure 2 Art Result

Dolls

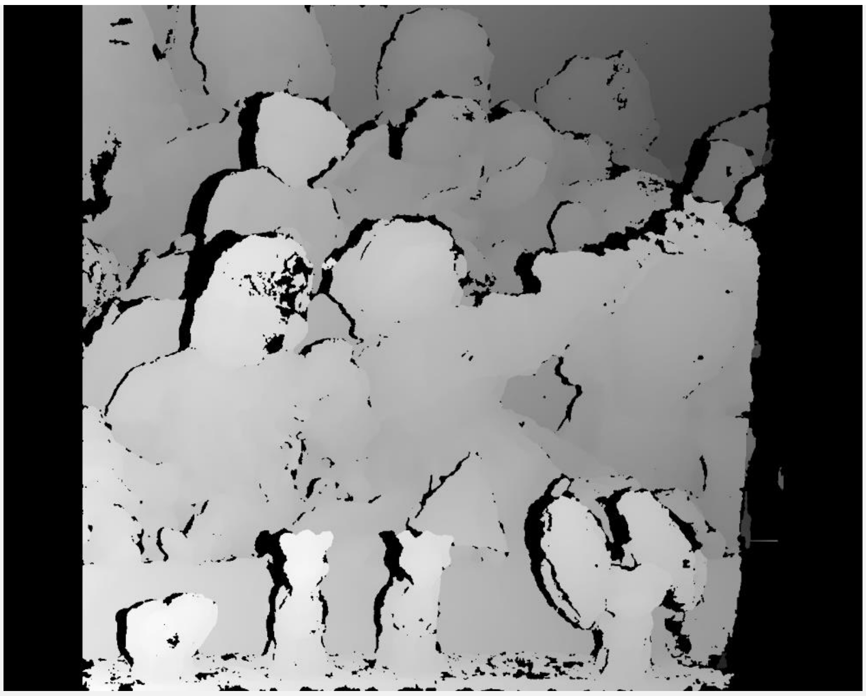


Figure 3 Dolls Result

Reindeer

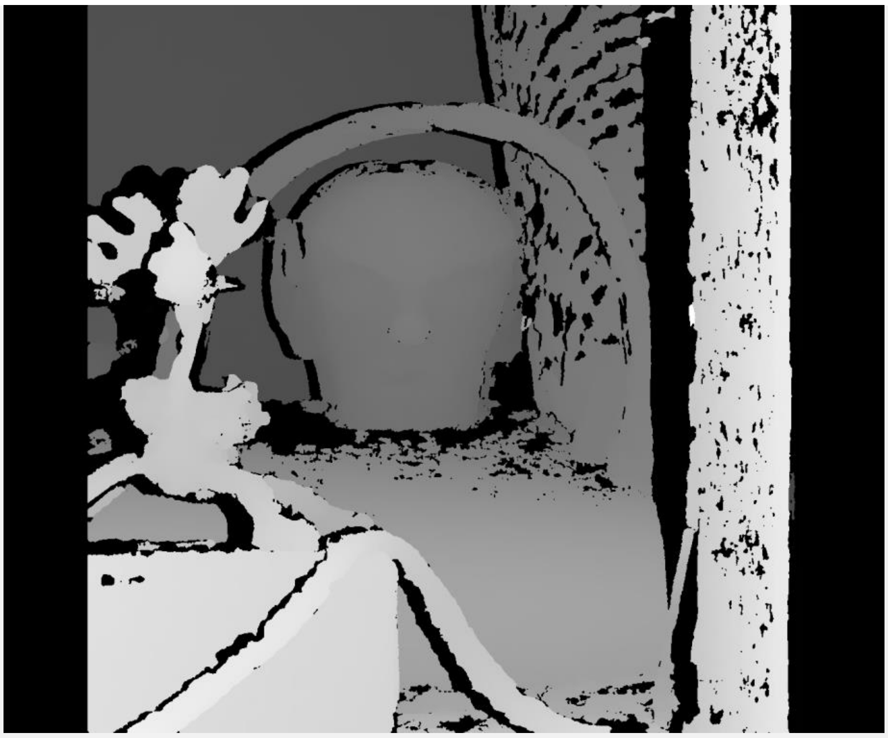
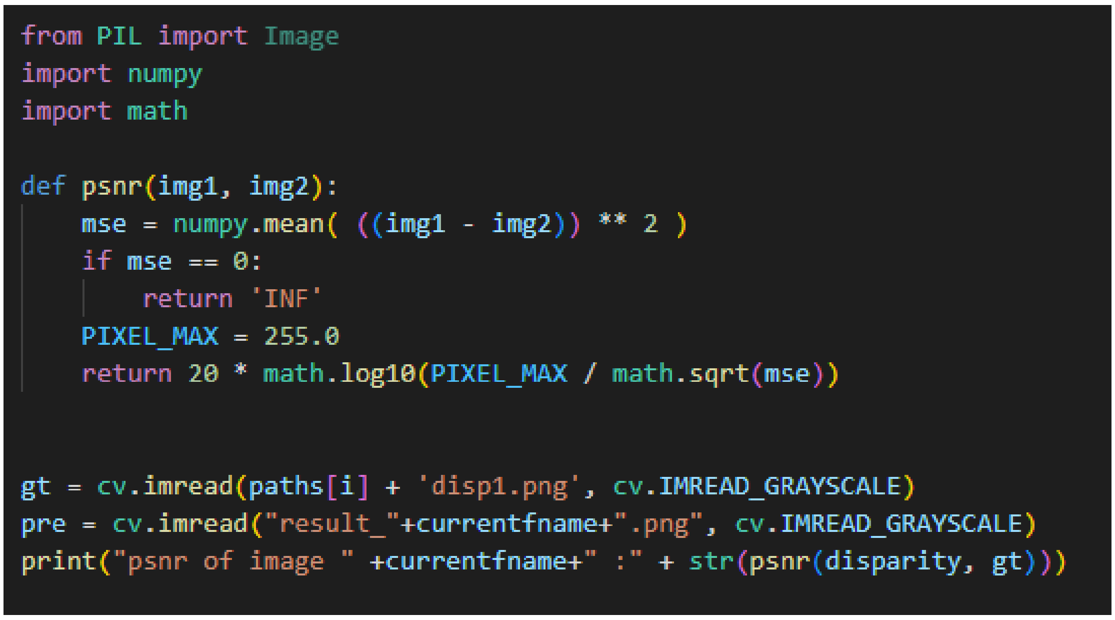


Figure 4 Reindeer Result

PSNR Score


```python
from PIL import Image
import numpy
import math

def psnr(img1, img2):
    mse = numpy.mean( ((img1 - img2)) ** 2 )
    if mse == 0:
        return 'INF'
    PIXEL_MAX = 255.0
    return 20 * math.log10(PIXEL_MAX / math.sqrt(mse))


gt = cv.imread(paths[i] + 'disp1.png', cv.IMREAD_GRAYSCALE)
pre = cv.imread("result_"+currentfname+".png", cv.IMREAD_GRAYSCALE)
print("psnr of image " +currentfname+" :" + str(psnr(disparity, gt)))
```

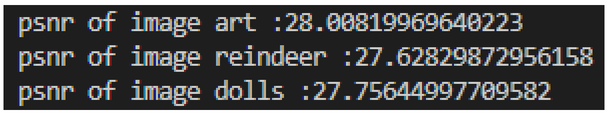

```
psnr of image art :28.00819969640223
psnr of image reindeer :27.62829872956158
psnr of image dolls :27.75644997709582
```

Figure 5 PSNR Result