Documentation - Task3

Peter Lorenz, Ivona Jambrečić

August 13, 2017

This documentation provides a short introduction to our code, the ways we have found to solve the given problem. Basically, the problem is about to read in stereo images (6 pairs in sum) and extract depth information of them. This depth information is crucial to know in the final point cloud, where the 2d point on the corresponding image pair is located in the 3d space. The following enumaration describes our implementation in detail.

1 Keypoint Detection and Description

First of all, the image pairs are read in as numpy objects for further processing. Then, image masks for left and right image are created, which marks where we should detect features. For every pixel, which has a gray-value bigger than 250 (from a range 0...255) must be almost white or in other words it must be a highlight. Moreover, all the neighbours (-5, +5) from the found highlighted pixel are set to 0. So it will only take the non-highlights. Takeing this masks and the related image to compute the keypoints and descriptors seperately. The number of keypoints/descriptors depending on the upper bound *nfeatures*, which we limited to 20.000 and a *contastThreshold* of 0.0001. We tested those input values with the function *checkout_contrast(im_left, im_right)*, which returns the best values by a maximum number of keypoints.

```
sift = cv2.xfeatures2d.SIFT_create(nfeatures=20000, contrastThreshold=0.0001)
```

On the opencv page (opencv.org) it is written that L2-Norm is good for SIFT, what we are currently using. The second parameter *crossCheck* returns only those matches with value(i,j), such that i-th descriptor in set A has j-th descriptor in set B as the best match and vice-versa.

```
cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)
```

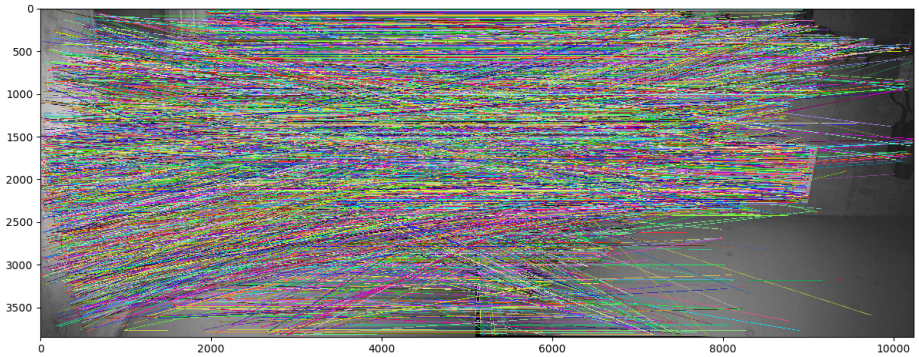

Figure 1: Matches

2 Sparse Stereo Matching

At this point, we know that we have rectified stereo images, that means that y values of two corresponding matches should not differ up to some few pixels. Those matches are thrown away. Another fact is that the difference of x-values of two corresponding matches can not be negative. A negative means that they are situated behind the camera. So, those matches are thrown away as well.

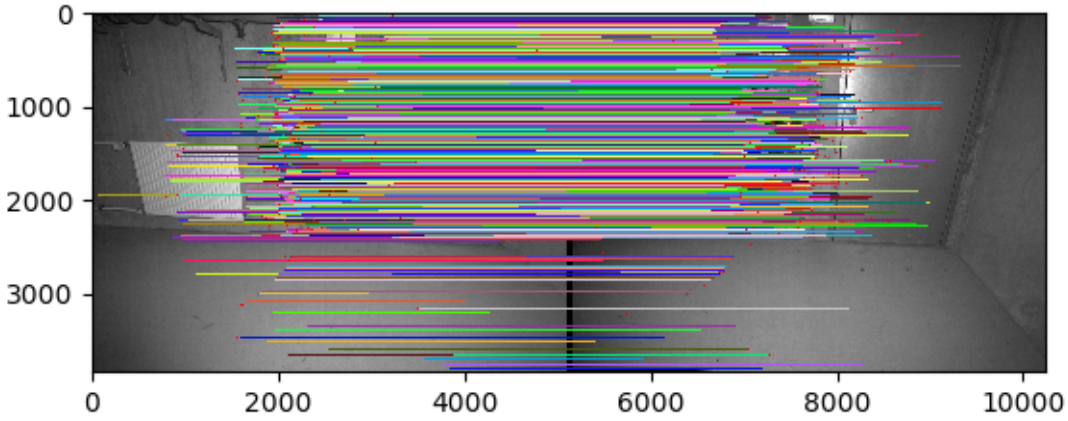

Figure 2: Stereo Matching

With those sorted matches we can calclulate the depth from disparity. We need homogenous coordinates of the left image to extend the image coordinates vector by 1.

$$hom_{left} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

The depth value z is calculated with the following formula:

$$z = \frac{f \cdot b}{x_L - x_R}$$

where f is the focal length from the given matrix K, b is the distance from both cameras and x_L as well x_R are the x-values of the left and corresponding right keypoint. The view ray must be calculated:

$$ray = K^{-1} \circ hom_{left}$$

With this ray with transform the image coordinates to perspective coordinates.

3 Forward Matching

Forward matching is in principle the brute force matcher, but with knn (k nearest neighbour). In our case, we take the k=2 (proposed by David Lowe) best matches, so that we can apply the ratio test. We multiply the second best distance with a threshold (0.75) and look the first best match distance is still smaller. So we can filter out the false second best matches.

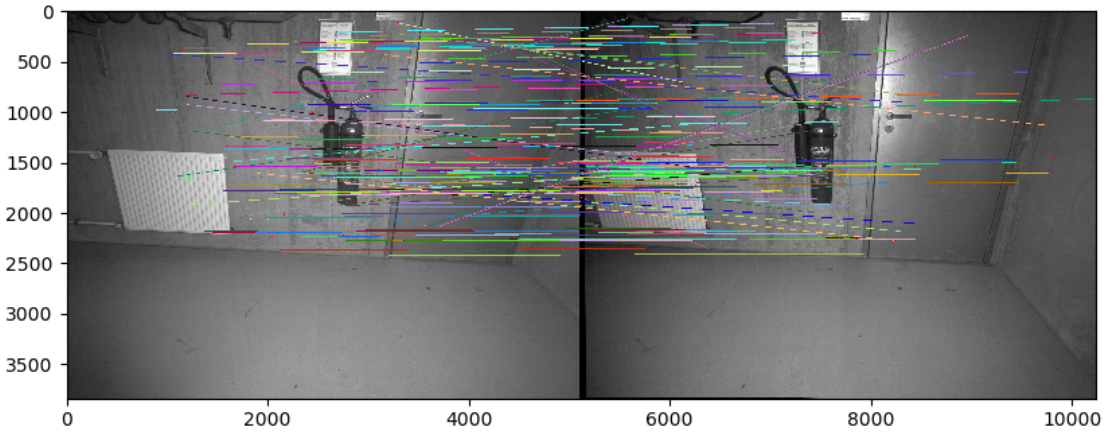

Figure 3: Forward Matching

4 Rigid Motion Estimation and RANSAC

The ransac is implemented and calls in every loop the rigid motion estimation to recompute rotation (R) and translation (t). Within a big loop, we take 6 random points from both point clouds and compute the translation and rotation with them. 6 points are needed, because of the degree of freedom (DoF): 3 rotation and 3 translation.
In the function *rigid_motion(X, Y)* we calculating the center (center of mass $\frac{\sum_i p_i}{\#p}$) of each point cloud and then the covariance matrix $cov = (X - \bar{X})(Y - \bar{Y})^T$.
After calculation the $U, s, V = svd(cov)$, the rotation $R = V \circ U$ and translation $t = \bar{X} - R \circ \bar{Y}$.
Some error checks helped to found errors in previous tasks. The rotation R and translation t can be verified by checking:

$$U^T \circ U = I, \quad V \circ V^T = I, \quad R \circ R^T = I$$

5 Result

Figure 4: Result 3D Point Cloud