Computer Vision 2023
Assignment 3

Sebastian Bergner

June 2, 2023

Task 1

(5 points) Make a very short video with your camera.

a. Compute and show the disparity map using OpenCV for two nearby frames. OpenCV disparity map tutorial.
The two near frames are one arbitrary image and the next one so i and $i + 1$.

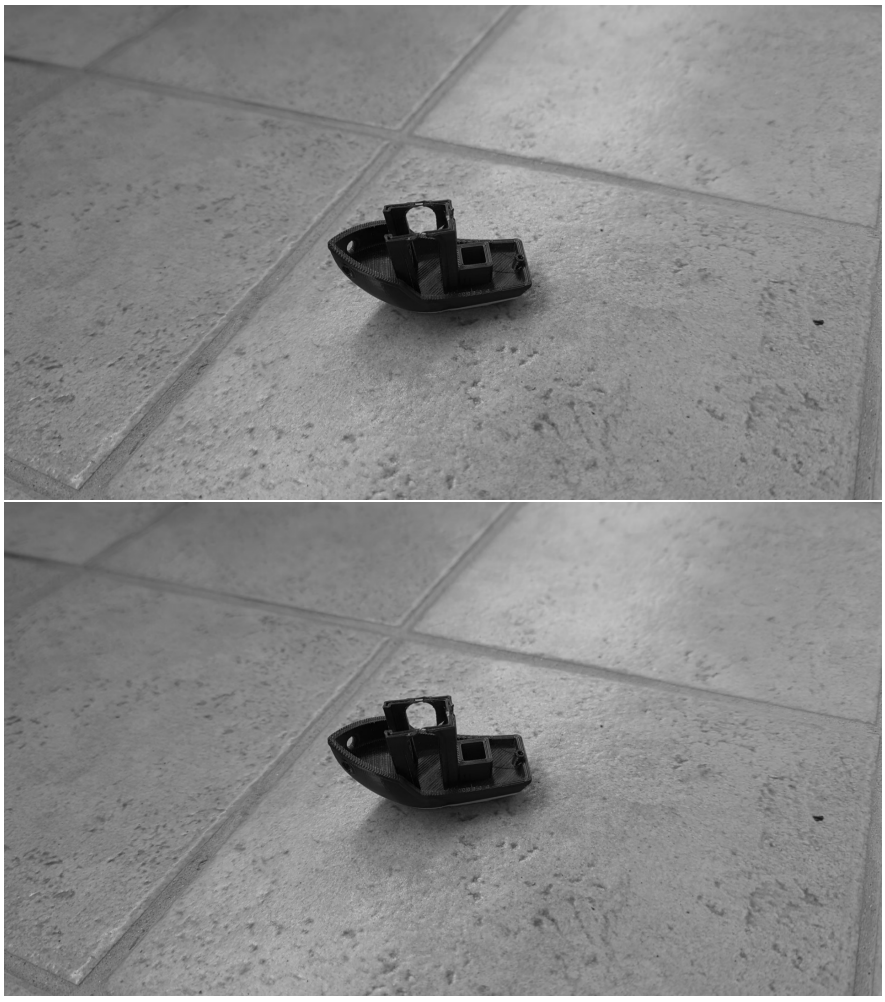

Figure 1: Near input images.

And the resulting disparity map.

Figure 2: Disparity map using near frames.

The result doesn't contain a lot of useful information. This is due to only very subtle changes of the pixels. This could be very much only noise.

b. Use now two far frames and compute and show the new disparity map. For the far frames I've used the same first image and tried to find a good second one. In the following case the 11th image yielded some promising results.

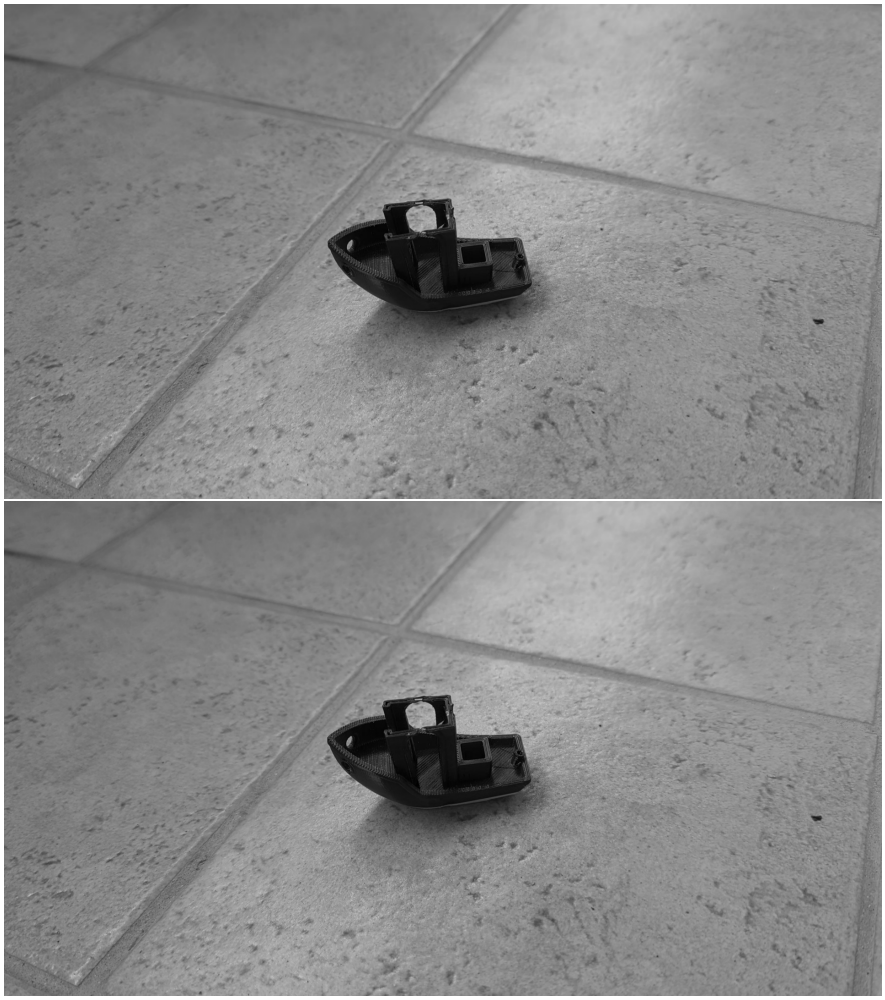

Figure 3: Far input images (11 frames difference).

And the resulting disparity map.

Figure 4: Disparity map using far frames.

c. Comment on how the results relate to the "distance" (i.e. near vs far frames) between cameras.

For the sake of it and to better compare the runs I made another video with quicker movements. Using the 11th image in that case resulted in a worse disparity map than using just 5 images difference.

Figure 5: Input images

Figure 6: Disparity map of the near (left) and far (right) images

The movement speed has a impact on how many images are ideal for generating the disparity map. Also the surface plays a role as can be seen in the second example where the bottle of glue has a homogenous surface. This is probably the reason why the disparity map has black areas where wide parts are homogenous.

d. Extend the code for stereo rectification.
Note: You can use OpenCV's cvStereoRectifyUncalibrated() function.

To apply stereo rectification I've used the of the second task. The steps that are required are:

1. Finding key points and their descriptors in both images.
2. Match the given key points in both images using the descriptors.
3. After that use the points to find the fundamental matrix of the system.
4. With the matched points and the fundamental matrix cvStereoRectifyUncalibrated() to get two rectification homography matrices.
5. (optional) Using these rectification matrices the images can be warped using a OpenCV function to bring the epilines to the same position in both images.

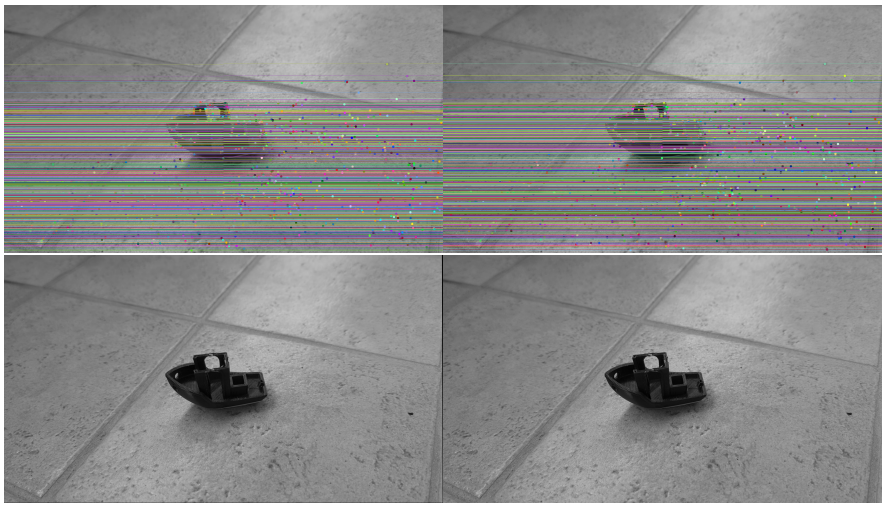

Figure 7: Epilines in both far frames and below resulting warped frames.

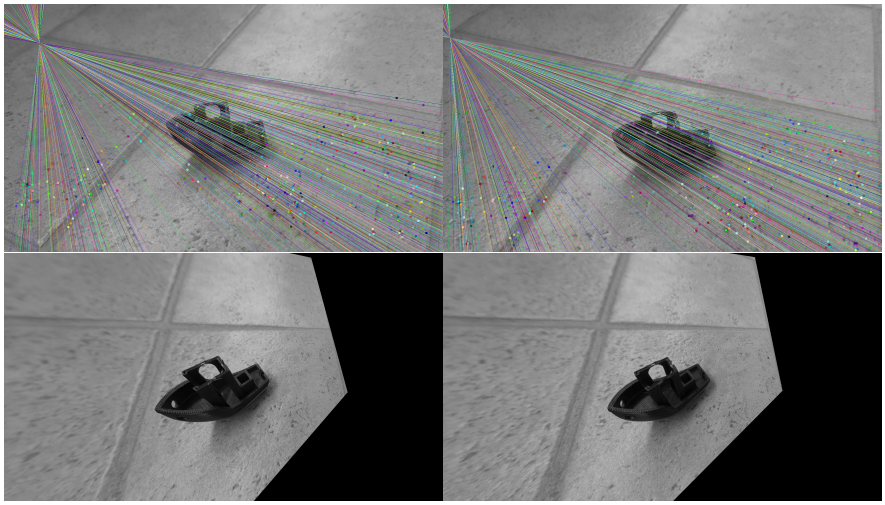

Figure 8: Epilines in much further (150 images difference) frames and below resulting warped frames.

Task 2

(10 points) Work with Epipolar geometry and multi view geometry. Use the frames you generated in the previous exercise:

a. Analyze and test the code from OpenCV Epipolar Geometry Tutorial.

b. Change SIFT with two different descriptors of your choice and compare the results with SIFT.

I've decided to use ORB and BRISK as replacements for SIFT as I've worked last semester with SURF and HoG and wanted to check out something new. Also this signal processing StackExchange Page helped a bit.

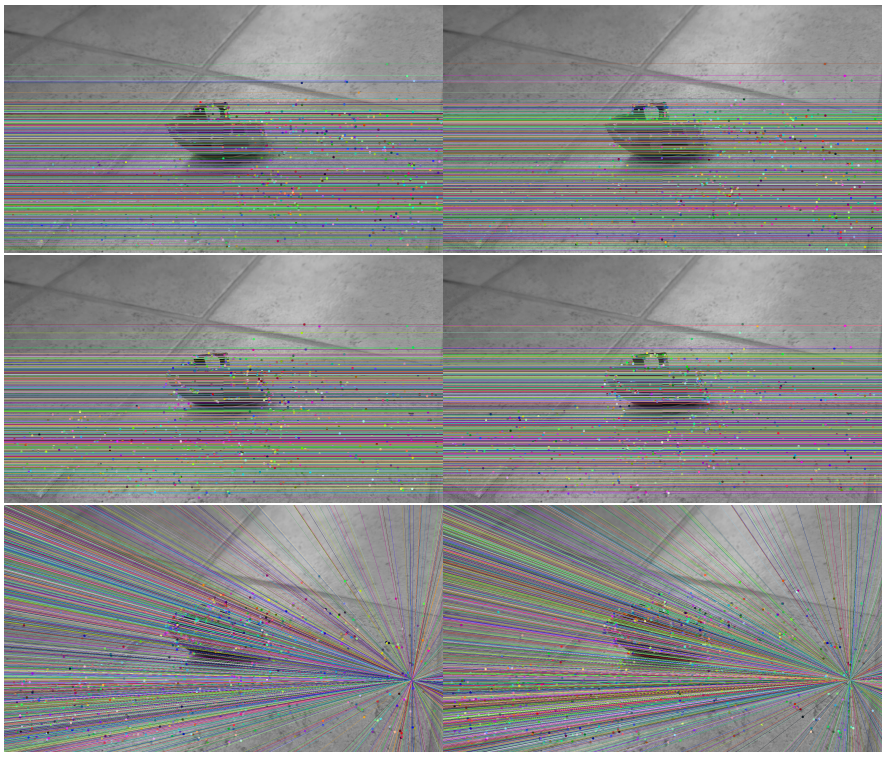

Figure 9: Output of the execution with SIFT (1), BRISK (2) and ORB (2) using LMEDS.

Both SIFT and BRISK seem to find good points for LMEDS to work because the lines are parallel, or aligning with to movement of the camera, whilst the epilines calculated with descriptors from ORB seem to be worse.
Between SIFT and BRISK I can't find a great difference, so they are both the "winners".

c. Change the computation of the fundamental matrix and compare the results of RANSAC and LMEDS. What is LMEDS?

LMEDS is short for Least-Median-of-Squares. LMEDS estimates parameters of linear equations with great robustness against outliers. It estimates by solving the following non-linear minimization problem:

$$\min \underset{i}{\text{med}}\, r_i^2 \tag{1}$$

To execute LMEDS there are several steps.

1. select n random samples from all the datapoints
2. fit datapoints to a linear (or polynomial) function using a method like least-squares to minimize the distances
3. before using the above equation, the residual r has to be evaluated
4. using the residual all points can be evaluated to get their median and save it
5. repeat the previous steps for i iterations
6. take the model of the minimal median

LMEDS relies on the median as it is much more robust to outliers than using the mean.

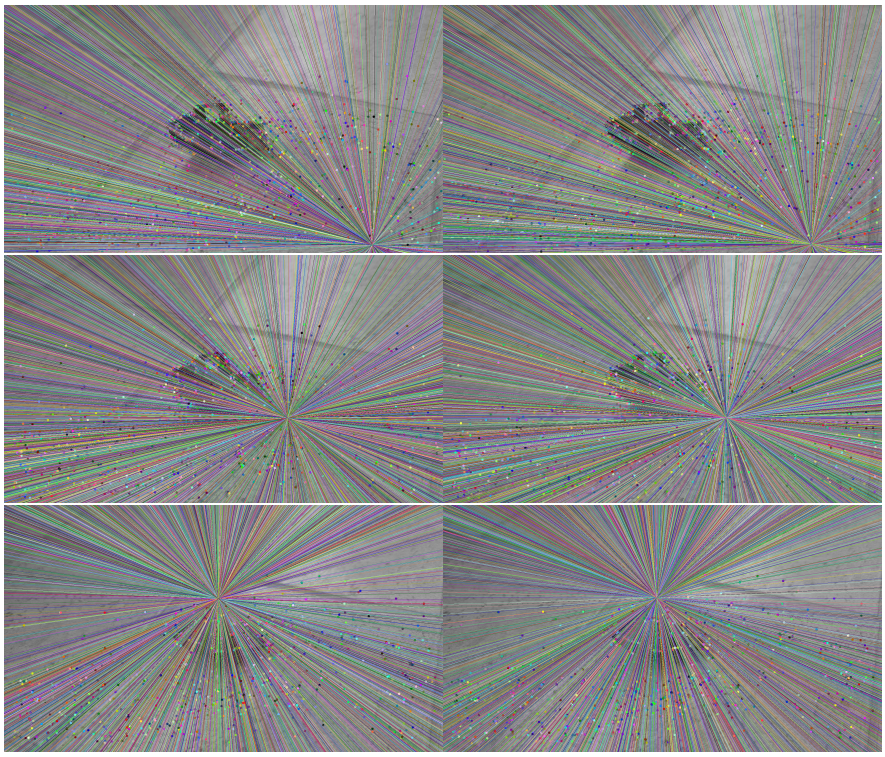

Figure 10: Output of the execution with SIFT (1), BRISK (2) and ORB (2) using RANSAC.

Using RANSAC seems to be much more prone to outliers or errors at least in this example. When trying to warp using these epilines just garbage would come out. I guess from these SIFT performed the best but still a lot worse when compared to using LMEDS instead of RANSAC.

d. Apply the best setup you have so far to different images of your choice from the PhotoTourism challenge dataset Image Matching Challenge 2021.

Figure 11: Samples from the Image Matching Challenge (Prague Parks Lizard).

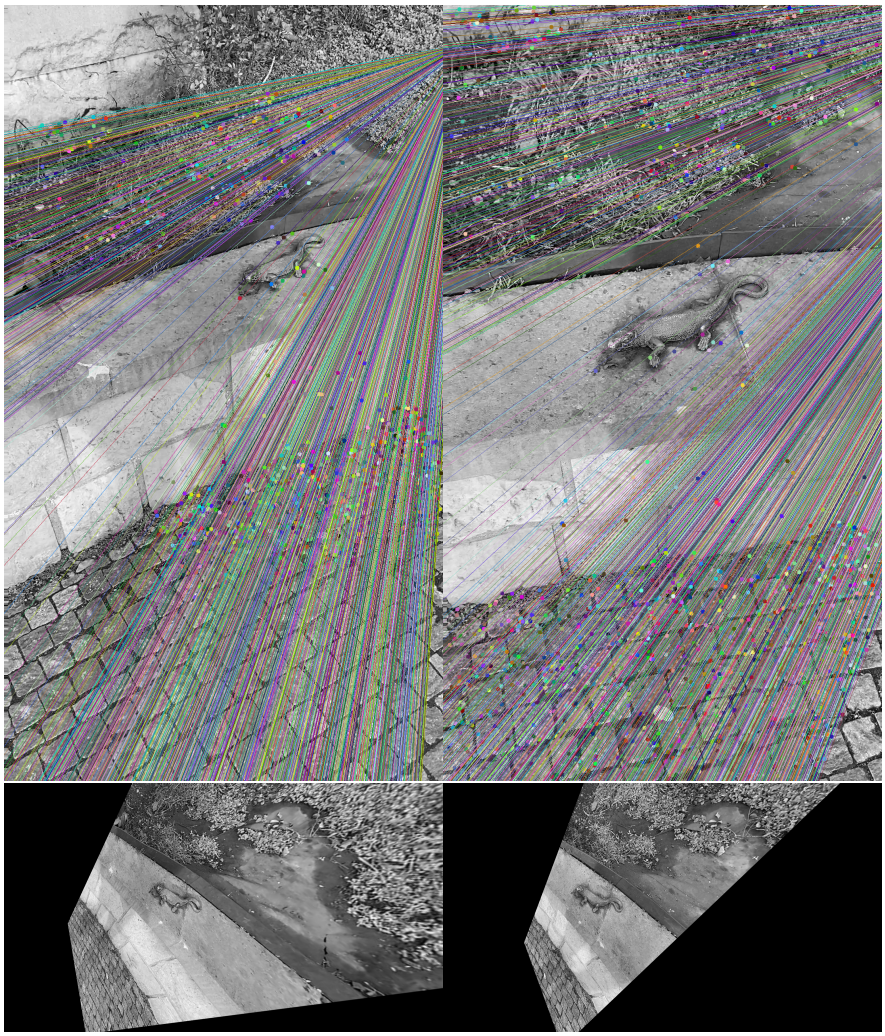

Figure 12: Output SIFT/LMEDS combination and warped result.

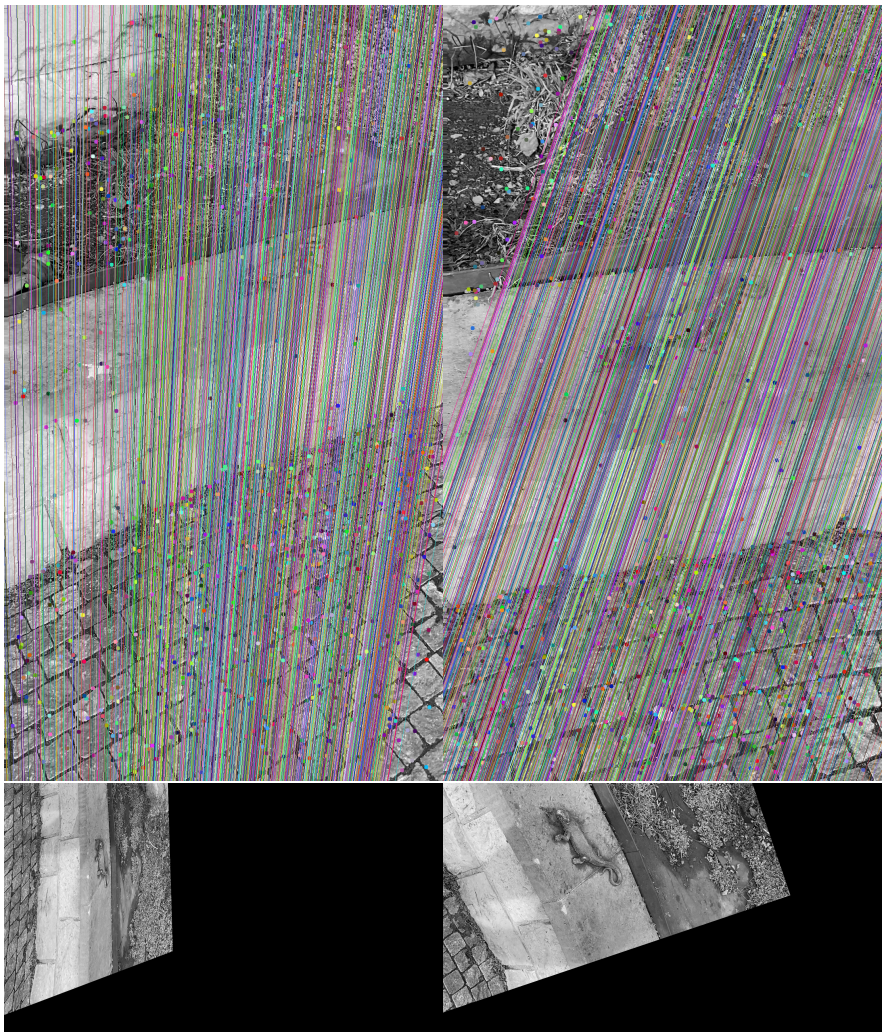

Figure 13: Output BRISK/LMEDS combination and warped result.

I used both combinations LMEDS with SIFT and BRISK as they performed equally good in the previous exercise. Here in this case SIFT seems to do a better job with the resulting warped image being less squashed together. It would be very interesting if we would use the entire dataset and tried to make a 3D model or reconstruction to see if BRISK/LMEDS would then with more data make a better job or be equally good estimating the epilines compared to SIFT/LMEDS.

Discussion

What you learnt about this assignment? and how the method can be improved or extended? What problems did you run into?
I learned how LMEDS works and what it is, and how it compares to RANSAC in a practical manner. Also I've deepened my understanding how RANSAC works. And how to use a matcher in this case FLANN to match keypoints in two images.
I played a bit around with the first exercise as I have a OAK-D Lite laying around, this was interesting to see how a (uncalibrated) stereo camera compares to a much simpler approach using just a video.

I had just one problem with the first exercise as I didn't know how many frames I would have to use in between to generate a "good" disparity map. I started with a few hundred as the video was recorded with 60 fps but this was too much as the output wasn't any good. To figure this out I just tried a bit around by guessing.