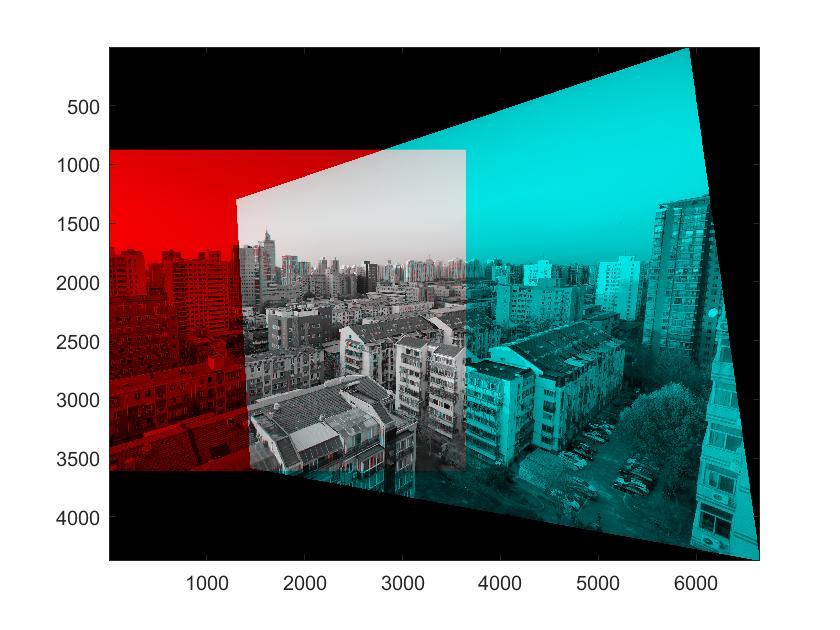
Q1

A pair of images taken by me





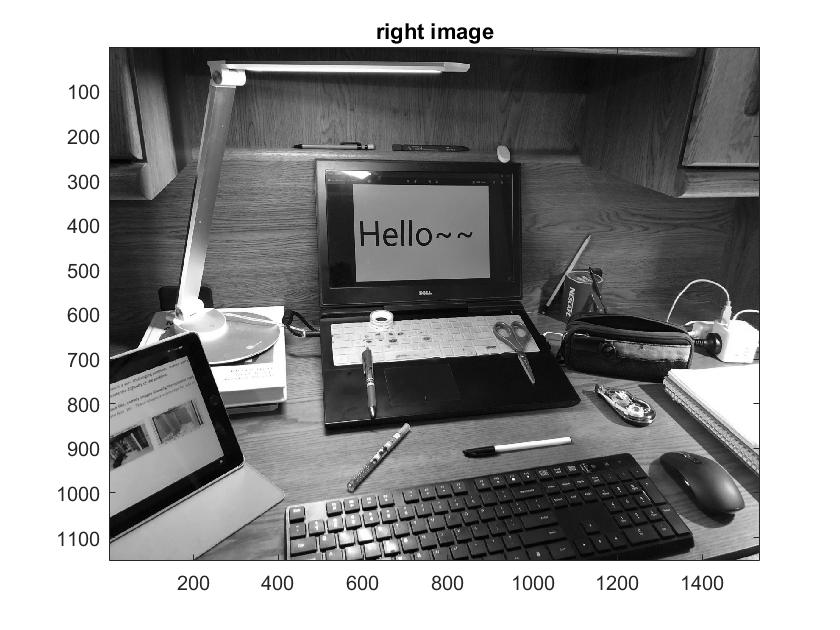


Q2

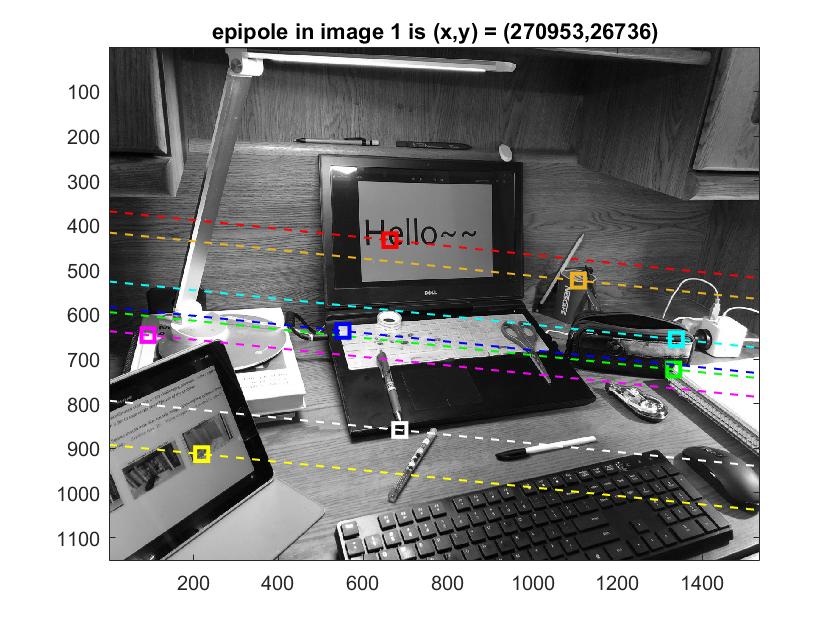
Original left image



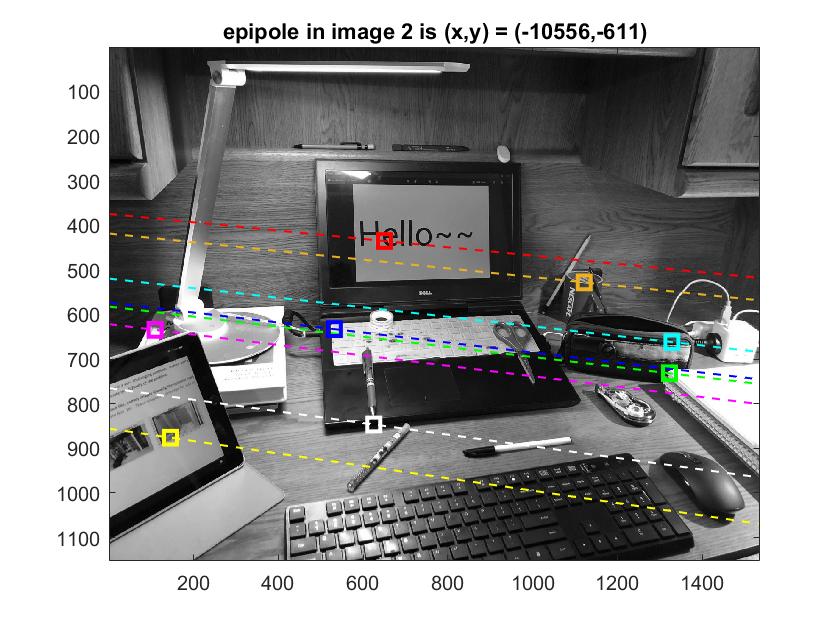
Original right image



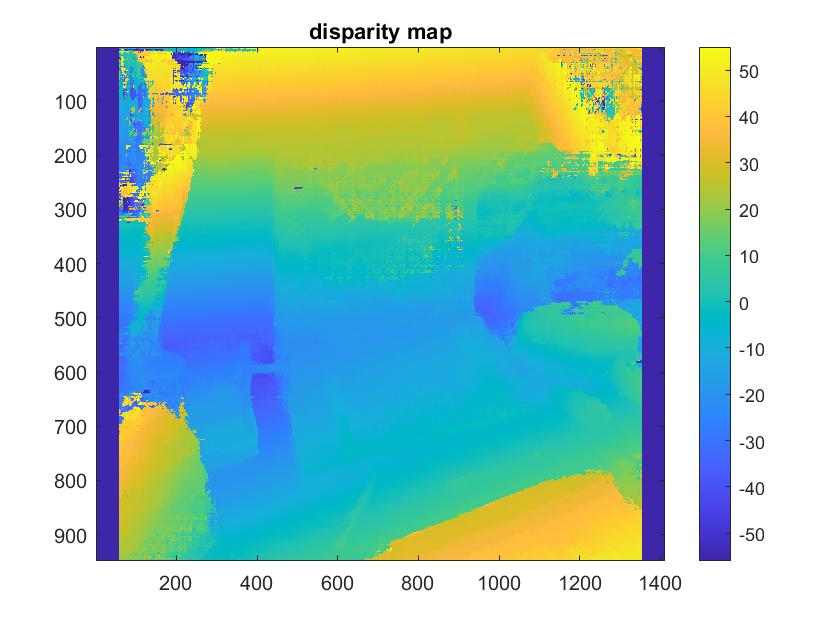
Left image with epipolar lines



Right image with epipolar lines

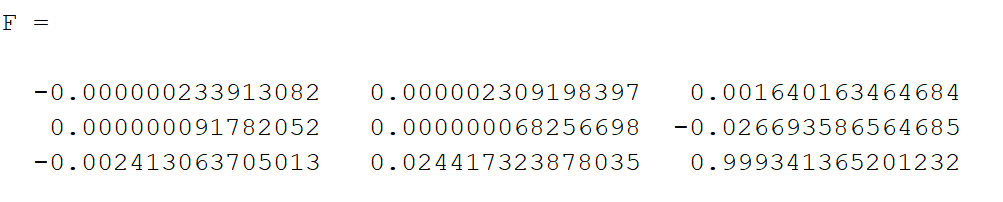


Estimated disparity image



Discussion

For some cases, the program successfully calculates a fair F, so the epipolar lines and epipoles are fine; since disparity map heavily depends on F(F is an input of function estimateUncalibratedRectification), a good F leads to a good disparity map; we can see that, indeed, the closer objects has a positive disparity, and farther objects have negative disparities. As a result, everything works. For example, the connection between keyboard and screen is relatively further to most things, including screen, pencil case, and ipad, which shows yellow, and the black keyboard indeed is very close to camera as its yellow color suggested. None of the results differs from what I expected, except for some error, like top left and top right of the image, caused by failed measurement of disparity between two pictures, because the failure in detecting and matching features in some areas, but it is minor.



A Typical F matrix

However, the program sometimes does not find a reliable disparity; the cause is probably on estimating F. Though by using RANSAC on matched SURF features gives a good F generally, it does not give a good estimate on epipole: usually F(1:2, 1:2) will be close to zero, and F(3,3) will be close to 1. If we do SVD on F, the last column of V or U, calling it vector e, will have two comparatively large number in first two elements, and nearly 0 in third element. They(two e’s) are left and right null space of F, which is geometically the homogenous coordinate of epipoles. As a result, the normal coordinate of epipoles will be in scale of tens or hundreds of thousands, coming from the quotient of two small numbers, and the sign only depends on the third element, which is very close to zero overall, so the sign of it may not be very reliable, so coordinates of epipole is numerically unstable. However, epipole is the position of camera A in camera B’s extended image, which is important for rectifying images; as a result, a poor or unstable estimate of epipole will lead to an inaccurate rectification, which will cause error in disparity map. For example, sometimes x coordinate of two epipoles are in the same sign, which is geometrically impossible, probably caused by the issue discussed above. In this case, the disparity map is bad, even pretty much inversed.

In fact, this way of getting disparity map has constraints. If two cameras are close to each other and nearly pointing to the same direction, it will be optimistic in rectifying two images because they are already quite rectified. However, coordinates of epipoles will be very large, because the cameras are pointing nearly in the same direction, the estimation of epipole is possibly bad, so overall we get a bad disparity map because of failure in estimating a good F, leading to an inaccurate epipole. However, if we separate two cameras with some distance and still pointing to the same scene, estimation of epipole will be good, because the camera B is closer to the view of camera A, so a good F will lead to a good epipole estimation. However, not only the shared features between two views are less, which makes the whole result a bit less confident(including matching), but there will be inaccurate and small-size result from rectification, because of the lack of shared features and view(shared features helps rectification, and only shared view can have a comparison in disparity, so the resulting disparity map will be small, especially in x direction). That’s why there is a number of cases where this program fails to produce an accurate and big-size disparity map.