ECSE 683 Liqiang Ding 260457392

Question 1

Implement CONSTANT_FLOW. Test it on frame pair pepsi04-pepsi05 and plot the resulting flow field. Since the field corresponds to pure translation (Tz = 0), recovery of depth up to a scale factor should be possible. Devise a method for recovering depth in this case and plot the resulting depth map. Note, since the height of the Pepsi can is known, determining an overall scale factor should not be too difficult.

Figure 1. Optical flow vectors from pepsi04 and pepsi05

The picture above is the result for calculating the optical flow vectors from pepsi04 and pepsi05 implementing CONSTANT_FLOW. As we can see, the direction of the moving camera gets correctly detected.

It is possible to recover the depth to a scale factor due to the optical flow field is known now. Intuitively, if the pixels are closer in depth, they will move faster. In other words, they will have a bigger optical flow vectors. Thus, we can use the optical vectors and the dimensions of the Pepsi can (K) to recover the depth.

$$Depth = \frac{K}{Optical\ Flow\ Vectors}$$

Question 2

Implement Algorithm FEATURE_POINT_MATCHING, which essentially bootstraps optical flow estimates. Use this to construct the 2N x n measurement matrix, W. Implement and apply the factorization algorithm, MOTSTRUCT_FROM_FEATS.

Given that we know the camera is translating parallel to the scene, what is the expected structure of R? How does the value of R estimated by factorization compare to the ideal case? The shape matrix S is defined relative to a coordinate frame with origin at the centroid of P and defined up to a scale factor. Again, with knowledge of any dimension in the image, this scale factor can be recovered. Devise an

appropriate method of comparing the dense depth map you recovered in Question 1 with the sparse point set recovered here.

Is the shape of the Pepsi can qualitatively correct? Which approach is more “accurate”? Explain any discrepancies in your results.

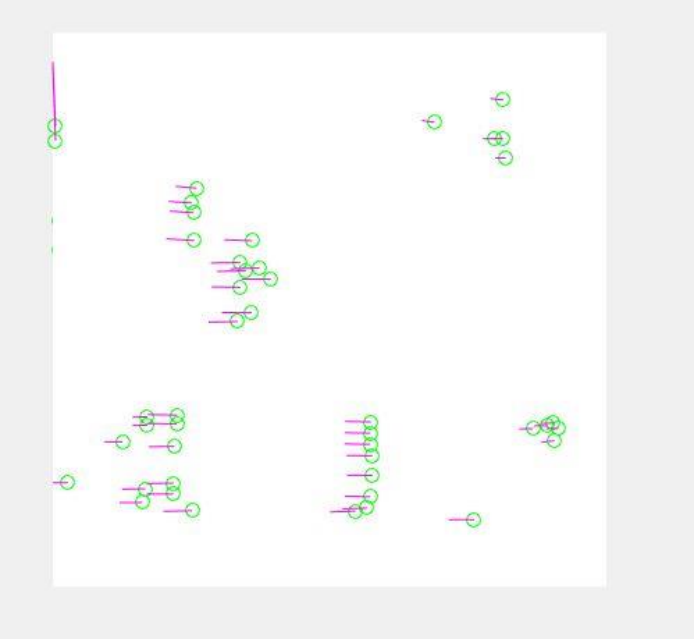

Figure 2. Optical flow vectors calculated based on multiple images.

I spent a lot of time on this question, but was still not able to fully implement the function. The above graph is my result of the optical flow vector calculated based on multiple images, which I believe can be better if possible. Please check my code for detailed information.

Even though the function of the feature matching is not fully calculated, it is valid to say the feature matching is more accurate, because there are more frames involved and the features (which are unique) are more easily to track. However, the feature matching method is more complex to implement and requires longer time to run.