EE576 Project #5

Mehmet Yiğit Avcı, *2017401033*

DENSE OPTICAL FLOW

OpenCV provides a function calcOpticalFlowFarneback to calculate the dense optical flow. For dense optical flow, vector fields around the whole image are calculated. In Fig.1 a consecutive image pair is given. In Fig.2 corresponding drawn vector fields can be seen. As expected, the vector fields are mainly on road lines and shadows where the motion can be seen easily. Code draws the vector field on the previous image or blank image as specified by the user.

Fig 1. Left: Previous image. Right: Current image.

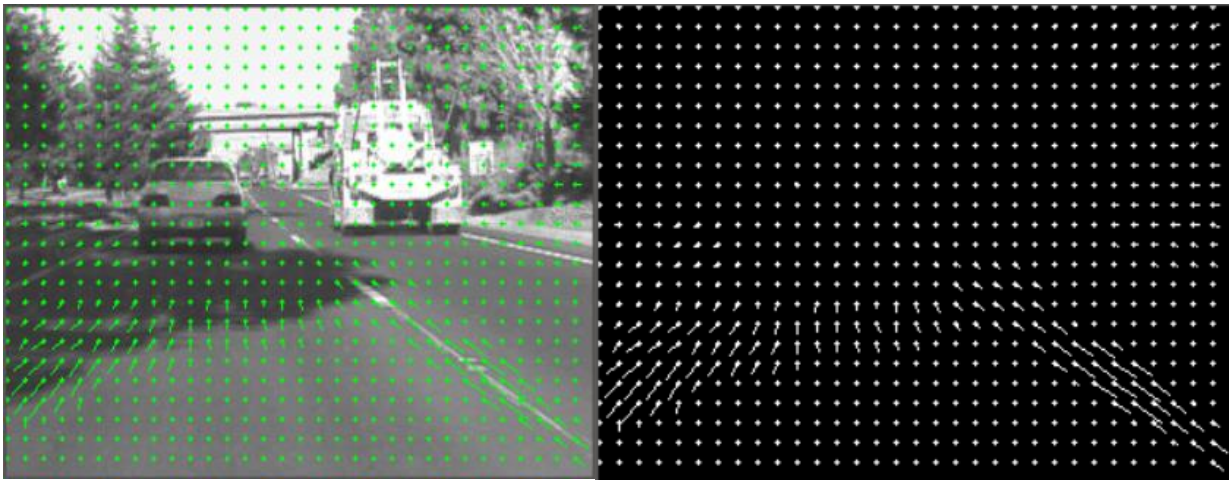

Fig 2. Left: Vector field drawn on previous image. Right: Vector field drawn on blank image.

For the human dataset, since camera moves arbitrarily and roughly, the optical flow is usually around the camera direction instead of the movement of the human or the cars passing around.

Fig 3. A sample of dense optical flow on human dataset.

Fig 5. A sample of sparse optical flow on human dataset.

SPARSE OPTICAL FLOW

For sparse optical flow, I have used Lucas-Kanade optical flow method. For this, OpenCV has a method called calcOpticalFlowPyrLK. With goodFeaturesToTrack method, I obtained the good corner points, then with the above method, sparse optical flow is found. As can be seen in Fig. 4 and 5, it first detects some points and tracks them. This method is useful for tracking objects. In car result, it tracks the car's and truck's corner points. In the human dataset, as in dense optical flow, the motion is essentially due to camera motion, sparse optical flow represents the camera motion. However, as can be seen, it is good at tracking the points.

Fig 4. A sample of sparse optical flow on car dataset.

FEATURE MATCHING

For this part, I have used SIFT methods for finding the first keypoints, and by using a FLANN based matcher, I have found the matches between two consecutive image pairs. Then, with a threshold, only good points are selected and two images with features are drawn and shown in Fig.6 and 7. There are too many features that are matched because it compares almost two identical images as two consecutive images does not differ too much.

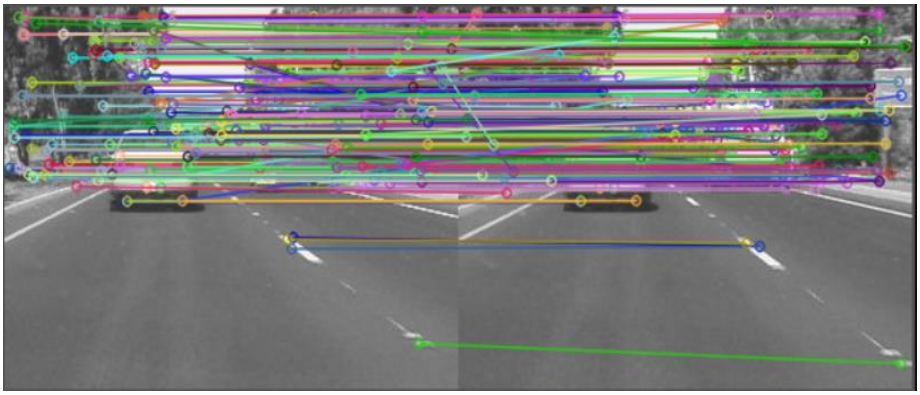

Fig 6. Drawn matched features on a car dataset sample.

Fig 7. Drawn matched features on a human dataset sample.

IMAGE STITCHING

For stitching two images we need several matched points. As we have found matched features in the last section, next step is to calculate a homography matrix between two images and stitch two images according to this matrix with warpPerspective method. After stitching the first two images, I have stitched the third image to first stitching result and continued this in an iterative manner. Also, I have eroded the unnecessary black parts of the resulting images with a bounding rectangle method.

The resulting car image with first 50 images are given below. When all images are stitched for car dataset, the resulting image is very wide as expected, therefore it is not shown.

For the human dataset, the result can be seen in Fig.9. The image has gotten larger in the opposite direction of the motion as expected.

Fig 8. Image stitching results for car dataset. (Only first 50 images).

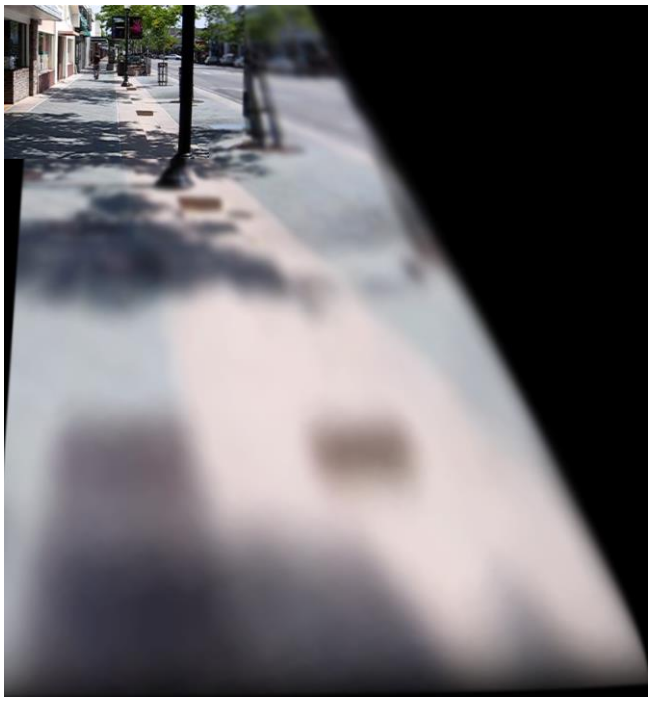

Fig 9. Image stitching results for human dataset.

REFERENCES

https://github.com/yangwangx/denseFlow_gpu/blob/master/denseFlow_gpu.cpp

https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html

https://docs.opencv.org/3.4/d5/d6f/tutorial_feature_flann_matcher.html

https://github.com/Manasi94/Image-Stitching