Batuhan Tosun

2017401141

EE 576 MACHINE VISION | HOMEWORK 5

The goal of this project is to work on the computation of dense and sparse optical flow. To accomplish our goal, in this project, we will be using two different datasets “car2.zip” and “human9.zip”.

Q1) In the first part of the project, we will implement a method that computes the dense optical flow between the two consecutive images. The method should also display the resulting vector field either on the image or on a blank image (this will be specified by the user). To do this, I have implemented a method/function called denseOpticalFlow() [1].denseOpticalFlow method takes two inputs from the user: class_name and flow_on_image. class_name is for specifying the dataset to be worked on and flow_on_image is for specifying the place on which the vector fields should be drawn. To compute the dense optical flow, OpenCV’s calcOpticalFlowFarneback() method is used and to display the vector field my drawOptFlowMap() method is being used. Below, one can see the results obtained on different datasets with different flow_on_image parameter.

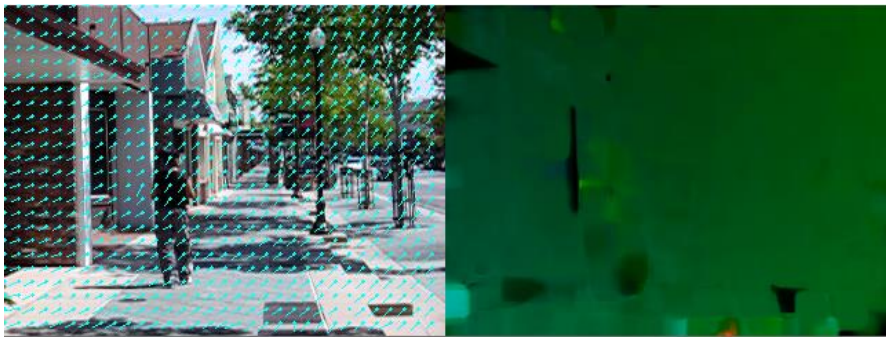

Fig1: Dense Optical Flow computation on the “human9” dataset (vector fields are displayed on the image)

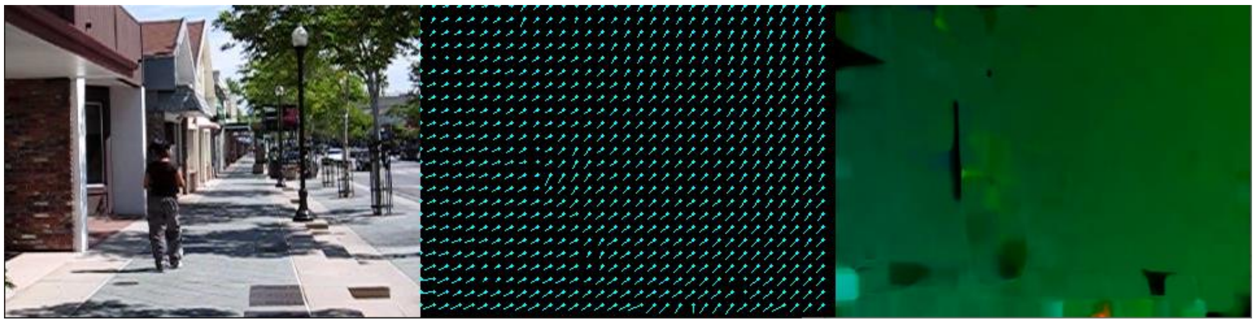

Fig2: Dense Optical Flow computation on the “human9” dataset (vector fields are displayed on the blank image)

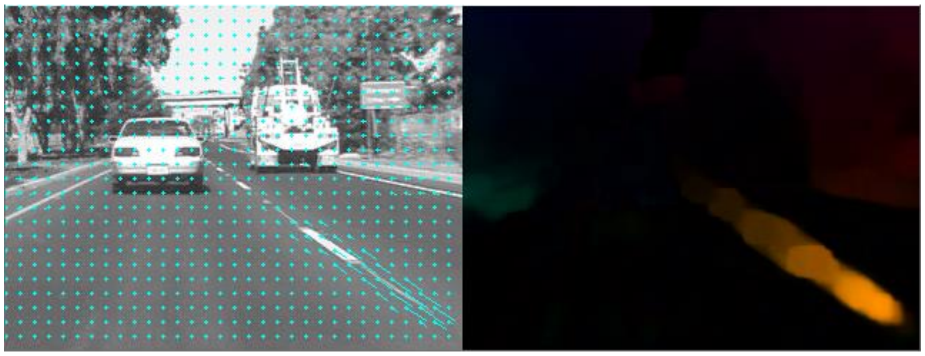

Fig3: Dense Optical Flow computation on the “car2” dataset (vector fields are displayed on the image)

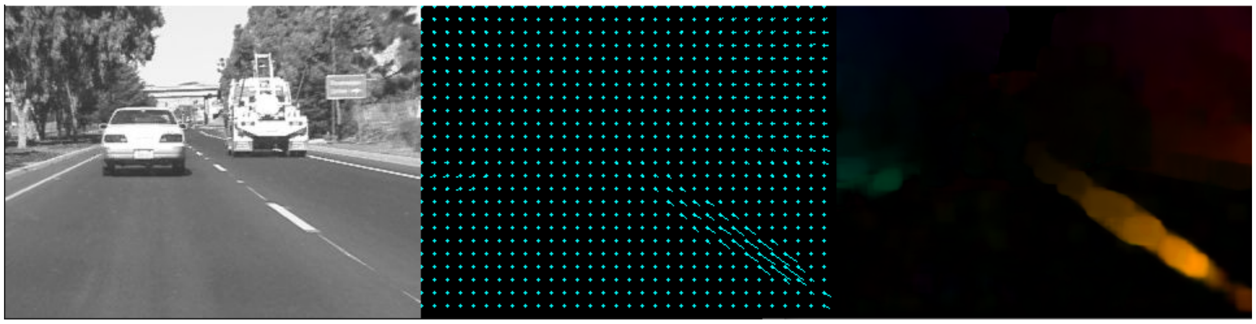

Fig4: Dense Optical Flow computation on the “car2” dataset (vector fields are displayed on the blank image)

From the resulting images, we can observe that the vector fields in the “car2” dataset are in direction of the incoming road lines/lanes most of the times. Also, the vector fields in the “human9” dataset are mostly in the direction of the camera motion/shake.

Q2) In the second part of the project, we will implement methods for sparse optical flow computation and image stitching. For sparse optical flow computation, I have implemented a method/function called sparseOpticalFlow() [1]. The features/points to be tracked in sparse optical flow are obtained via OpenCV’s goodFeaturesToTrack() method, which gives information about the corner points, instead of SIFT features. Also, to compute the sparse optical flow between the two consecutive frames in KLT Tracking manner, OpenCV’s calcOpticalFlowPyrLK() is used. The results for sparse optical flow computation with different datasets are given below.

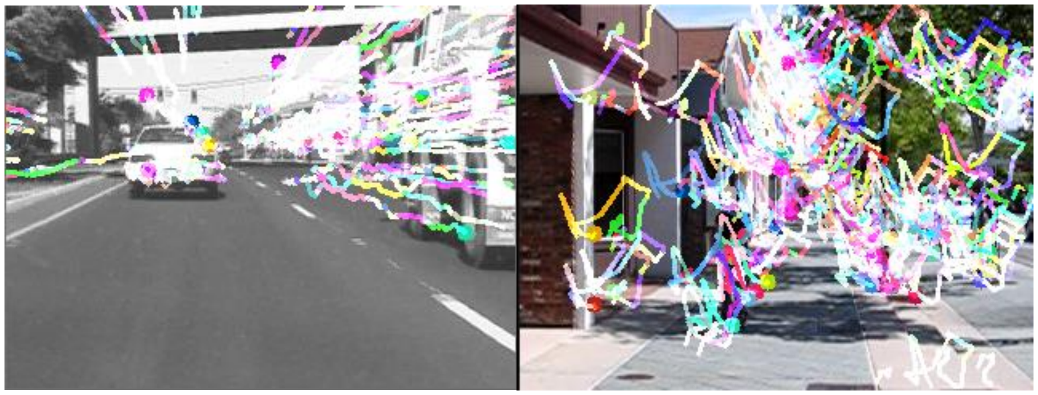

Fig5: Resulting images from the datasets after sparse optical flow computation

Since sparse optical flow computation and image stitching tasks are different challenging tasks, I have implemented a separate function called imageStitching() for only image stitching. To do this, the first step is to extract the SIFT features of the two consecutive frames with OpenCV's SiftFeatureDetector() and SiftDescriptorExtractor() methods. Then, we match the SIFT features of these frames with knnMatch() method. To filter the good matches, Lowe's ratio test is applied [2].

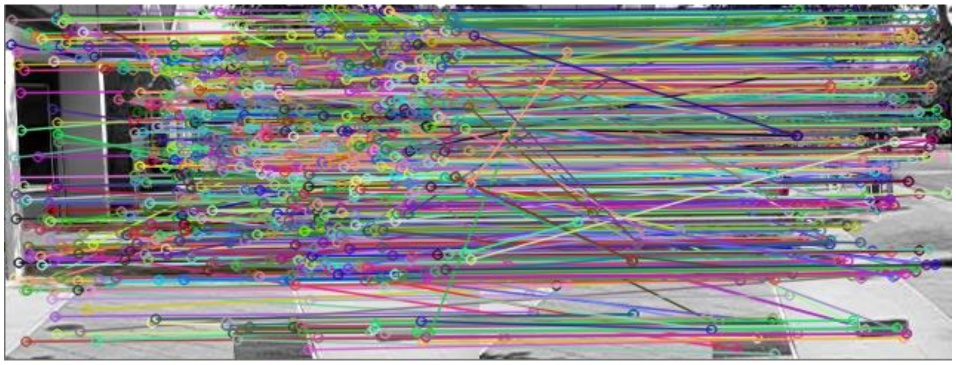

Fig6: The matching features between the two consecutive frames in "human9" dataset

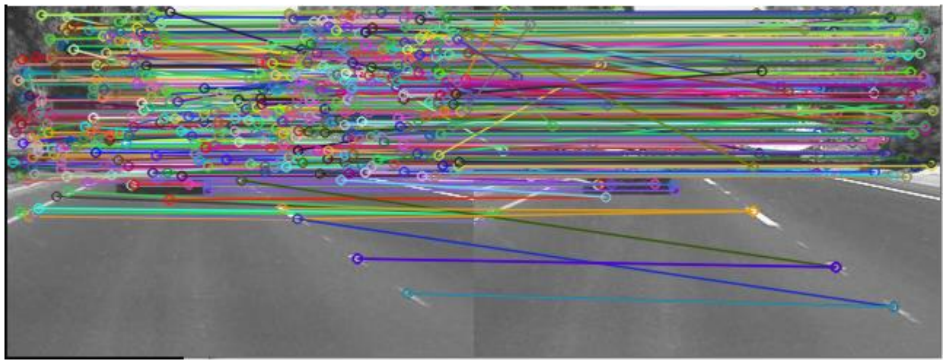

Fig7: The matching features between the two consecutive frames in "car2" dataset

Using these good matching points, we compute the homography map via OpenCV's warpPerspective() method + RANSAC algorithm. Then, we use homography map to apply perspective transformation via OpenCV's warpPerspective() method. The perspective transformed previous frame and the untouched next frame are stitched and as a result warpedImg is generated. In the consecutive iterations, the warpedImg will be used as the previous frame to stitch all the images in the dataset [3]. The resulting images after stitching all the images in the two datasets are given below:

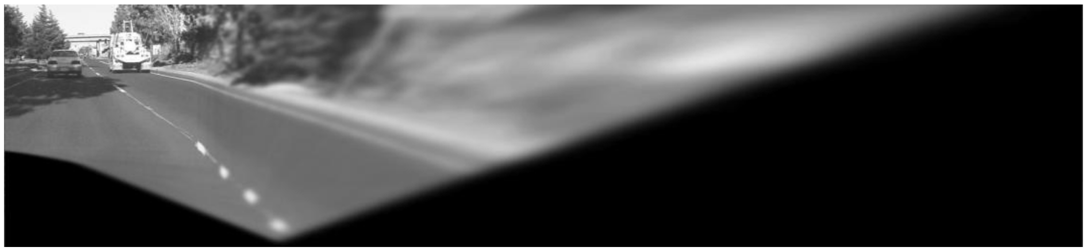

Fig8: The resulting image after stitching the images up to a certain frame in the "car2" dataset

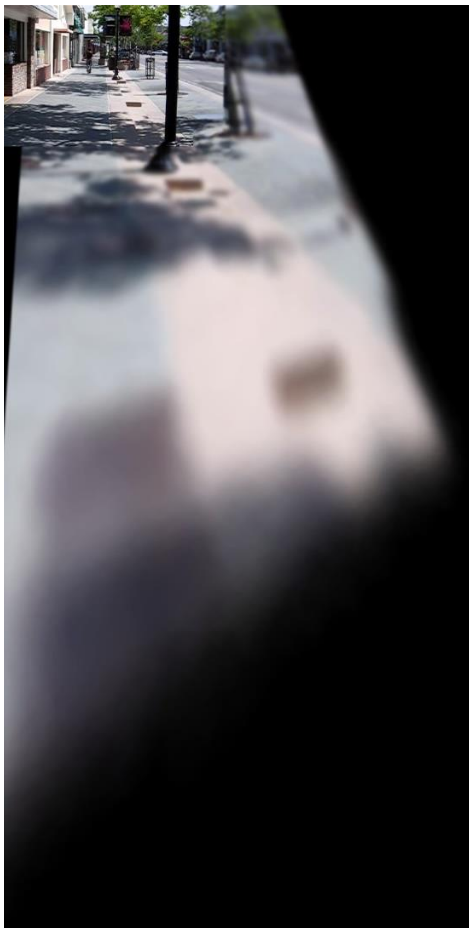
Fig9: The resulting image after stitching all the images in the "human9" dataset

REFERENCES

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

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

[3]: https://docs.opencv.org/4.x/d9/dab/tutorial_homography.html