COMP3056 Computer Vision Coursework Report

Xinyi OUYANG / 20030865

The link of the complete project (contains the test video and result videos) is https://nottinghamedu1-my.sharepoint.com/:u:/g/personal/scyyo1_nottingham_edu_cn/EShl86gqlOxIjV1Zj1-EGg4BARPfCFkEcePKxtw7XRhkQQ?e=rW8M9T

1. Objectives & Features

Objectives

The main objective in this project is to write a program using sparse optical flow to track objects from given video. The basic steps are finding good feature points and then compute the optical flow of those points to track the locations in the next frame. The input of the program is a short video, and the output is the same video with the trajectories of the good feature points. There are two mode in my project, one is the standard mode (without user interaction), the other can allow user to select the tracking area and objects.

Features

Original feature:

Detecting the good feature points and computing the optical flow to track, then draw the tracking line of those points in the video.

New features:

a. **User interface**.
It allows user to select which mode to run in the program (MODE 1 is standard mode and MODE 2 allows user interactions). Illustration contains the details of the two modes. Additionally, each mode contains two features.

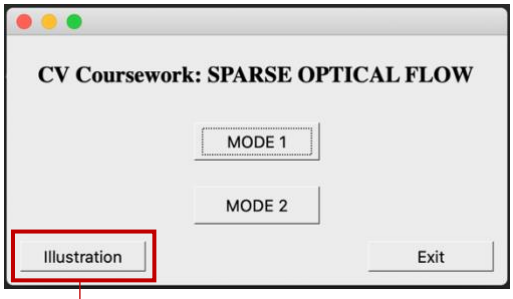
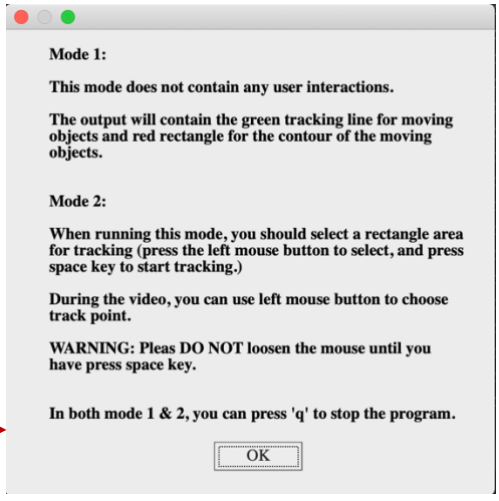


Figure 1. User Interface



b. Updating tracking points to implement real-time detecting (can accept longer input videos)

Detecting new objects appeared in the video and removing the trajectories of disappeared objects. Also, in order to avoid the confusion of the image, it set a tracking length, which means that the max length of the shown trajectories will not beyond the tracking length.

It not just tracking the good features that detected in the first frame, every 5 frame it would re-detect new objects. And if the object moved out of the video, the trajectory of this object would be removed. This feature will make the project accept a longer input video without confusing the picture.

c. Contour detection (In MODE 1) with frame preprocessing

In addition to draw the tracking lines of objects, mode 1 also did the contour detection, and would draw red rectangles around the moving objects in each frame.

d. Tracking certain area (In MODE 2)

Allow user to draw a tracking area (rectangle) and track the objects (good feature points) in this area.

e. Tracking certain points (In MODE 2)

In mode 2, after drawing the tracking area, if user wants to track other objects out of the area, this feature can be used. It allows user drawing points on certain objects and the points would move followed the movement of the objects.

2. Implementation Details

Good feature points

The first method that was thought to use for finding good feature points in this project is SIFT. It uses Gaussian filter to convolve the image in different scales and takes the DoG images, and the key points are found as the local minimum or maximum of the DoG image. Some points at the edge of the image are unstable, to get stable key points, it uses Hessian Matrix to eliminate the edge response of DoG (remove the points with low contrast or along the edge). OpenCV provides an API to create SIFT and detect the key points. The detected features can be ranked by their scores and the number of the features to retain can be set as the parameter. And sift.detect() will return the coordinates of the key points.

Since SIFT need to cost long time, another method, Shi-Tomasi corner detector, is chosen to detect the corners. Shi-Tomasi algorithm is an improvement of Harris algorithm. Harris algorithm determines the corners through the physical variation in each direction of a slide window, while Shi-Tomasi directly computes the feature value of matrix M and compares the small value with the threshold and get the strong corners. OpenCV provides function *goodFeaturesToTrack()* to implement the Shi-Tomasi corner detector. The main parameters of this function are the max number of corners, quality level (usually between 0.01 to 0.1), minimum distance between two corners and the size of the computing area.

Optical flow

Lucas-Kanade algorithm is used to estimate the optical flow between two continuous frames. The preconditions of lucas-kanade algorithm are the brightness is constant between the two continuous frames, and the motion of the object between two frames is small. It assumes the flow in a local neighborhood of pixels is constant and solves the optical flow equations for the pixels in that neighborhood by the least squares criterion.

calcOpticalFlowPyrLK() can compute the sparse optical flow by pyramid lucas-kanade iteration. Inputting the previous good feature points and the two frames (current and previous), it will return the new position of the points and the status vectors of points (if it can find the flow of the feature, status is set to 1, else set to 0). The points with status = 1 are considered to be the points that should be tracked (the points are moving). Considering the problem of camera shake, an error value is set, which means that each time we will calculate the distance between the new position and the previous position of the points, if the distance is large than 0.5, it is considered as a moving point, else, we assume the motion of this point due to the camera shake.

Updating tracking points (For feature_b)

There is an array to store the variation of the coordinates for all the good feature points. Each row denotes to the variation of one point. When the point keeps moving, the new coordinate (computed by Lucas-Kanade) will be added at the end of this row and if the length of the row beyond the tracking length, the first coordinate will be removed in order to accept a new coordinate. If the new coordinate is out of the boundary, it means the point disappears from the video, so the row of this point will be deleted.

When reading the first frame, SIFT is used to detect the good feature points because it can generate vast of feature points (compared with Shi-Thomasi) and avoid missing some key points. Every 5 frames, Shi-Tomasi algorithm is used to re-detect the good feature points to detect the new objects appeared in the video (it is fast than SIFT and more stable). After re-detecting the good feature points, it will check whether the new-detected points are in the current points or not. If the new-detected points are not in the current points, they will be added into the array as a new row (as new feature points), and in the next frame, it will compute the optical flow for the updated points, which can implement the real-time detecting.

Contour detection with image preprocessing (For feature_c)

To implement the contours detection, a function *findContours()* provided by OpenCV can be used. The input image of this function is the processed binary frame by *threshold()*, parameter CV_RETR_EXTERNAL is set to only detect the external contours. The returned value of the function is hierarchy vectors of the contours. With the obtained contours, using *boundingRect()* can get the smallest normal rectangles that wraps the contours. Using the vertex coordinate, width and height of the rectangle (obtained by the *boudingRect()*), the rectangle of the contours of the objects can be drawn.

But we only want to draw the contours of the moving objects instead of all the objects in the frame. Therefore, before do the contour detection, the frame should be preprocessed. Background subtractor can be used to subtract the background of the image. Before reading the video, *createBackgroundSubtractorMOG2()* is used to create a background object, and when capturing each frame of the video, *backgroundsubstractor.apply()* is used to get the foreground mask. This background segmentation algorithm based on mixed Gaussian model. It creates a probability density function for every pixel, for the pixel of a new image, if the value can be described by the probability density function, it can be regarded as the background. Additionally, *backgroundSubtractorMOG2* adds the shadow detecting function, with which the shadow will not be recognize as the foreground object and make the result better (the shadow will not influence the shape of the foreground object).

After the above preprocessing steps, it will be implemented to only draw the contours of the moving objects in each frame.

Tracking certain area (For feature_d)

selectROI() is used to capture the user's mouse events. When reading the first frame,

the program will call *selectROI()* function to allow user to select a part of the image (also draw the rectangle of the selected part). The function will return a tuple that contains the coordinates of the top left vertex (the minimum x and minimum y), the width and the height of the selected area. Using the obtained information of the selected rectangle, the program will crop the original frame to the selected area and transform it to gray-scale image, then only detect the good feature points in that area and computes the optical flow of those points to implement tracking the object in certain area which selected by user.

It is important to note that the coordinates of the good feature points in the selected area is not the same coordinates in the original fame. When getting the coordinates of the tracked points by *goodFeaturesToTrack()*, the coordinates should be transformed to the coordinates in the original frame since the trajectories of the points should be drawn on the original image. For example, the obtained coordinate of a key point in the selected area is (x, y), the coordinate of that point in the original frame is (x+w, y+h).

Tracking certain points (For feature_e)

OpenCV provides EVENT_LBUTTONDOWN to capture the mouse click event. There is a Boolean to save whether it captures a mouse event or not. When capturing mouse event, it will draw a circle in that clicked point, and the coordinate of the clicked point will be saved into an array, which stores a series of coordinates and the variations for the points that drawn by user. In each frame, the program will compute the optical flow for those points and add the new coordinates of the moving points to re-draw the tracking circle. If the object stops moving, the tracking circle will be remained at the place it stops and if the object disappears from the video, the tracking circle will also disappear. And users can select as many tracking objects (points) as they want.

3. Result & Explanation

The following figures are the screenshots of two modes from the test video that captured by myself. Figure 2 shows the result of mode 1. All the moving objects are tracked in this mode, and the trajectories are drawn by green lines, and the contours of the moving objects are drawn by red rectangles. While in mode 2, shown as figure 3 & 4, only the objects in the selected area are tracked (the two people with green lines), and there are three red circle drawn by user (please zoom the figure to check) to track the electro mobile and other two people. In figure 3, the blue rectangle is drawn by user.

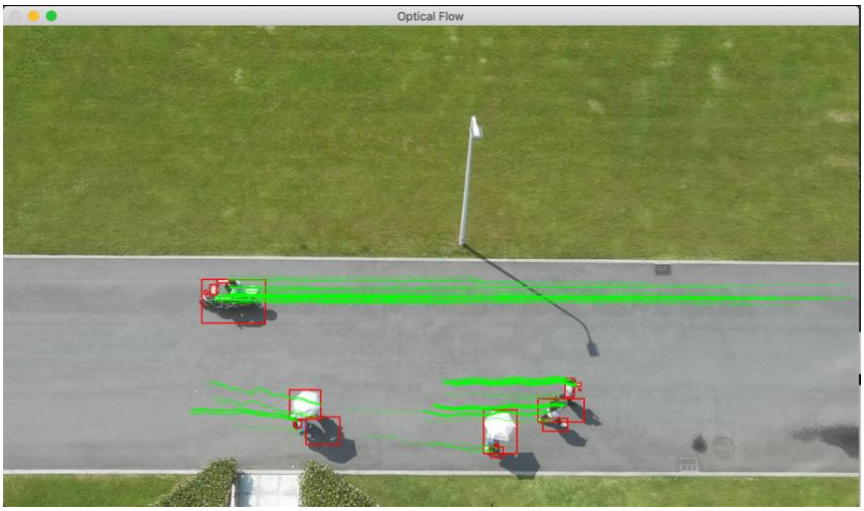


Figure 2. Screenshot of Mode 1

Figure 3. Screenshot of Mode 2 (user selected tracking area)

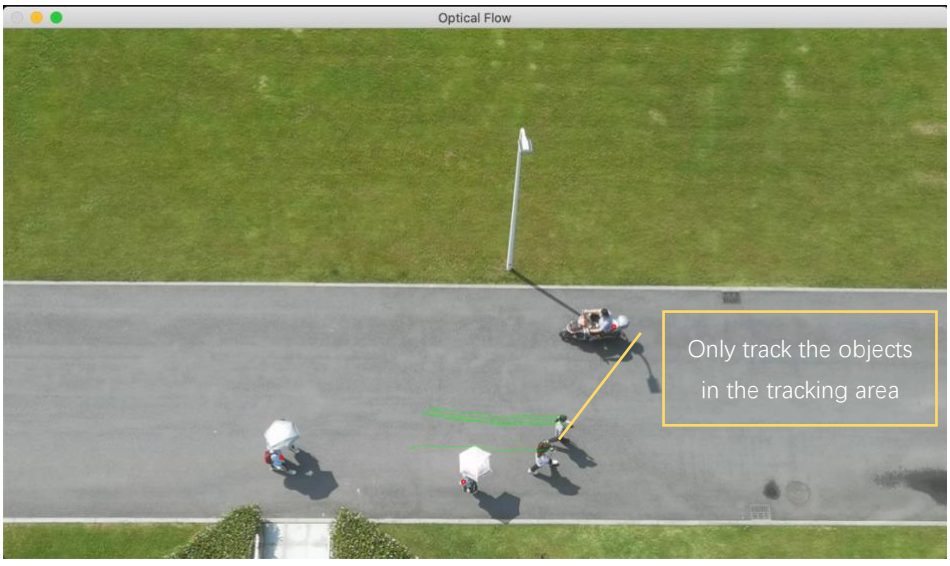
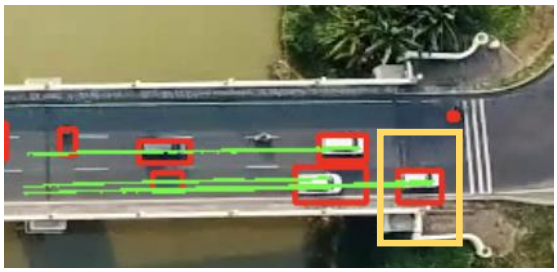
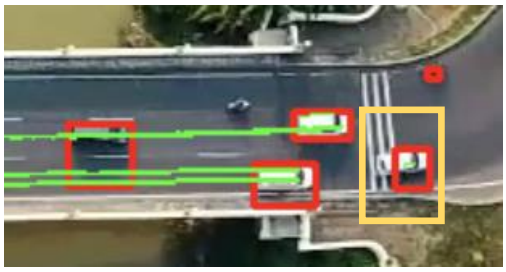


Figure 4. Screenshot of Mode 2 (tracking result)

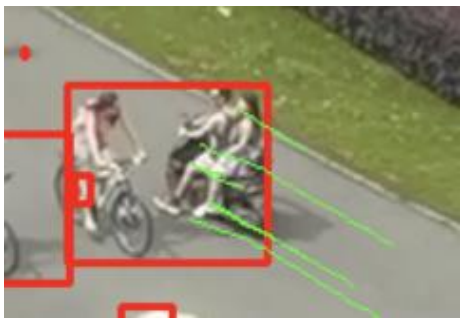
The above results show that the algorithm basically realizes all the features I want, but when analyzing a more complex videos, it will encounter some problem. And the following screenshots are when using other more complex videos to test.

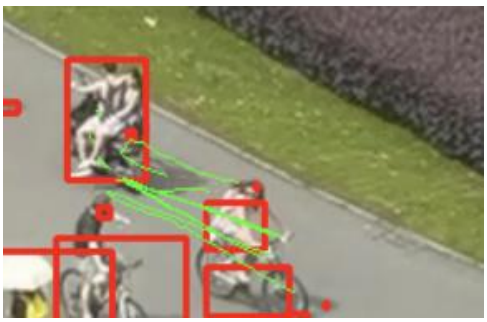
Firstly, in the above result, there are some moving cars do not have the trajectories (framed by the yellow rectangles) since the points on the cars are not be detected as the good feature points. It might because the max number of corners are set to small, or the objects are very similar to the background so it will not be regarded as good feature points. Increase the number of detecting corners or use SIFT can alleviate the problem to some extent.





Secondly, noting the white care (framed by a yellow rectangle) in the above figure, before the zebra crossing, it has the trajectories, but when moving on the zebra crossing, the tracked points lost. It might because of the limitation of Lucas-Kanade that the brightness between the two frames changes or the points move not like their neighbors, it cannot find the optical flow and might lose the points.





Another problem is caused when two objects overlap. The left above figure shows that the trajectories are on the electro mobile, but after the overlap of the electro mobile and the bicycle, some of the tracking points on the electro mobile move onto the bicycle, makes the image confusion. It because the algorithm calculates the new position by the current coordinate but cannot tell whether it is on the same object.

4. Discussion

Corner detection and updating

Strengths: it achieves real-time detection that can not only tracking the good feature points in the first frame but can track the new objects appeared in the video. Combining SIFT and Shi-Tomasi makes the speed of detection faster (compared with only using SIFT), and the accurate of the detection is good. Also, the extracted feature points are uniform and reasonable.

Weakness: though using SIFT or set a larger number of corners can extend the richness of the feature points, there still are some objects cannot be detected. Especially, when the color of the object is similar to the background, it may not be regarded as a good feature point.

Lucas-kanade optical flow

Strengths: can be implemented to track the moving objects by calculate the optical flow.

Limitations: based on the precondition of the Lucas-Kanade algorithm, when the motion of the object is large or it move to the shadow (cause the change of the brightness), the tracking points will be lost. And when two objects overlap, the tracking points in each object might be in disorder.

Draw contours

Strengths: because of the background subtraction (image preprocessing), it only draws the contours of moving objects.

Limitations: sometimes the camera might shake and cause the little motion of the background. In this case, the algorithm has difficulty distinguishing the background from the foreground, some objects that are supposed to be the background are also

sketched out. And though there are image preprocessing steps, there still would be some noise points. Additionally, it cannot ensure each object has only one contours, sometimes one object may have two or three different contours in different parts of the object.

Tracking certain area or certain objects

Strengths: implement the user interaction. Allow user to select as many objects as they want to track.

Limitation: the tracking area only can be select in the first frame and cannot to be change after selecting. Because it also uses the Lucas-Kanade to calculate the optical flow to track, when the object has a large motion or move to shadow, it might lose the track.