COMP3065 Computer Vision
Coursework Report

Chen Liao 20030694

1. Aims and Objectives

Project (b) is chosen for this coursework. An object tracking program is designed and implemented based on feature identification and optic flow algorithms. This project aims at tracking salient feature points appearing in the first frame throughout the whole video with their trajectories plotted. The program is compliant with online fashion, which means that the rendered video will be played in real-time with a new window prompted when the program starts running. There are multiple extra multiple features implemented to enhance its functionalities, which are listed as follows:

(a) Self-implemented Harris corner detection algorithm for feature identifications.

(b) Self-implemented Lucas-Kanade algorithm for optic flow calculations.

(c) Self-defined feature cleaner including:
  * An adaptive threshold mechanism
  * A minimal distance discriminator to reject excessively dense features.
  * A K-means clustering for optimizing the selection of feature points.

(d) Image warping and iterative LK algorithm for higher accuracies.

(e) Pyramid Lucas-Kanade implementation for large movement robustness.

(f) Real-time video capturing and tracking.

2. Methodologies

The general framework of this project involves three stages, including the feature selection stage, the motion detection stage, and the position update stage. The feature selection stage extracts salient features worth tracking from the first frame of the video with algorithms such as the Harris corner detection and the Shi-Tomas corner detection algorithm. Only coordinates of feature points will be recorded and tracked throughout the whole video. The motion detection phase utilizes the Lucas-Kanade optic flow calculation algorithm in an iterative and pyramid fashion to estimate the velocities of feature points between frames. According to its results, the positions of these points will be updated as the input for the next iteration. To plot the tracking, all positions of a feature point are saved as a mask on the image, which will be connected by lines to show the trajectory when visualizing. An intuitive illustration of the system workflow could be found in Figure 1.

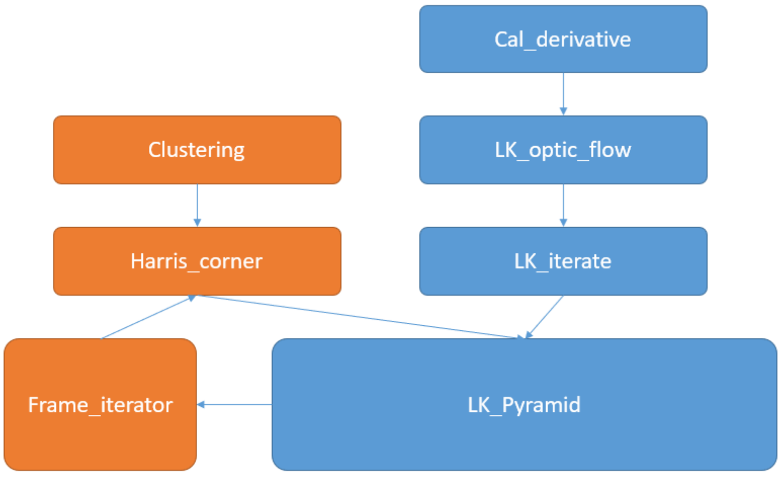


Figure 1: overall system workflow

2.1 Harris Corner Detection

Harris Corner detection algorithm identifies corners on a raw image through a computational approach. A corner to be detected is a point whose first-order derivative varies the most at every direction, which could be interpreted as Equation 1. This could be transformed to a formula with a Harris matrix through Taylor expansion as shown in Equation 2. In this way, the response values of potential corners could be evaluated as Equation 3, as k is a constant factor ranging from 0.04 to 0.06. By filtering points with a predefined threshold, features to be tracked are thus determined.

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2 \qquad \text{Equation 1}$$

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \qquad \text{Equation 2}$$

$$R = \det(M) - k(trace(M))^2 \qquad \text{Equation 3}$$

For specific implementations, derivatives along both x and y directions of the image are first calculated through Sobel filters. Then, values for the Harris matrix $(I_x^2, I_y^2, I_x I_y)$ for each pixel in the image could be derived by calculating dot products of proper derivatives and applying a Gaussian blur. The window size specified in this project is 5*5, as Harris matrices for all queries in the window should be summed to get the matrix for the patch. To achieve this, the maps of

I_x^2, I_y^2, and I_xI_y are convoluted with a sum filter of the patch size with 1 in all its queries, giving maps of all possible$\sum_{x\in p} I_x^2$, $\sum_{y\in p} I_y^2$, and $\sum_{x,y\in p} I_xI_y$. Finally, the map of response values is calculated through Equation 3 for all patches. There are some tricks when selecting features. Firstly, since the threshold for filtering varies drastically across different videos, a dynamic threshold is proposed. Instead of a predefined constant, the threshold in this project is set to $0.1 * \max(response)$. Secondly, to avoid dense aggregation of feature points, a minimal distance is enforced. A feature point would be accepted only if it is adequately far from all selected feature points.

2.2 K-means Feature Clustering

Considering the program feasibility, dense feature tracking is trivial and resource-consuming. Therefore, it is beneficial to decrease the number of features. In this project, K-means clustering is deployed to select optimal feature points in a heuristic manner. K-means is an unsupervised clustering algorithm that automatically aggregates scattered points into clusters. For all points detected by the Harris corner algorithm, K-means divides them into a stipulated number of clusters based on their spatial distributions on the images. As it is assumed that one cluster includes all features for a single object, the feature point nearest to the center of the cluster could be regarded as an equivalent point for the object to be tracked. In this way, dense tracking could be converted into sparse tracking which improves the overall performance.

2.3 Lucas-Kanade Optic Flow (LK)

An LK algorithm calculates the current directions and velocities of pixels based on two consecutive frames of the video. By assuming that all movements are smaller than 1 pixel, the intensities of feature points within the two frames are constant, and all pixels in a local window have the same movements, Equation 4 could be formulated. This could be converted into a least-square problem as shown in Equation 5, which could be solved through Equation 6. Following these ideas, a basic LK algorithm could be merged into this project.

$$0 = I_t(p_i) + \nabla I(p_i) * [u \quad v]$$

Equation 4

$$\begin{aligned} &\min\left(\left||Ax - b|\right|^2\right) \\ &subject\ to: \\ &A = \nabla I(p_i) \\ &x = [u \quad v] \\ &b = I_t(p_i) \end{aligned}$$

Equation 5

$$\hat{x} = (A^TA)^{-1}A^Tb$$

Equation 6

Firstly, derivatives of x, y directions are calculated through Sobel filters, and the derivative of time could be calculated through the difference between the two consecutive frames. Since features are identified through the corner detection algorithm, only the least-square problem of patches centered at these points is solved. For every 5*5 patch, 25 linear equations could be represented in matrix form and $A^T A$ thus becomes the Hessian matrix of the patch. In this way, $\hat{x}$ could be identified through linear algebra techniques with its first component indicating the velocity along the x direction and the second along the y direction. Therefore, two maps storing all values of u and v could be constructed as the result of optic flow for the current frame pair.

2.4 Iterative LK Optic Flow (ILK)

A salient drawback of the rudimentary LK algorithm is that it deprecates higher-order derivatives when applying the Taylor expansion series as shown in Equation 7. This inevitably causes errors in the calculation of u and v. To address this issue, the iterative LK Optic Flow (ILK) with image warping is implemented in this project. Image warping mainly refers to the spatial transformations of images, whereas in this project it twists frames by moving pixels along its u and v. The basic idea of ILK is to iteratively update values of u and v based on the image warped by the current u and v. This keeps terms of higher-order partial derivatives along both x and y directions, thus increases precisions.

$$I(x+u, y+v) \approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v \qquad \text{Equation 7}$$

Although the general idea of image warping twists the whole image, the implementation in this project only twists image patches centered at selected feature points to guarantee the efficiency. Patches of these features are firstly identified on the old frame. Then, these patches are displaced by u and v which specify new local windows on the new frame. The new patches are thus selected as current positions of the original feature on the old frame. In theory, the difference between the new patches and the original patches should be small. ILK thus applies basic LK algorithm to corresponding patches in the two sets to derive a new set of u' and v'. Finally, the current set of u and v could be fixed up by $u = u + u'$ and $v = v + v'$. The above process could be iterated to constantly update motion indicators until their convergences.

2.5 Multi-Resolution LK Optic Flow (Pyramid)

LK algorithm assumes that all possible movements within a unit time are less than 1 pixel, which violates the real circumstances. Therefore, the multi-resolution optic flow is implemented in this program in a Gaussian pyramid fashion. Its basic idea is to leverage the results of LK from images with smaller resolutions (usually higher levels in the image pyramid) to initiate values of u and v for images with higher resolution (lower levels in the pyramid). The steps of its implementation in this project is as follows:

a) Build image pyramids for both the old frame and the new frame through down sampling.
b) Down sampling the input features.
c) Starting from the smallest level, use ILK to compute its optic flow.
d) Up sample the features and the motion maps from the current level.

e) Use this as the initial input of the ILK of the subsequent level.
f) Repeat c) to e) until the last level of the pyramid.

2.6 Live Tracking

Since the OpenCV library has been deployed for video and image processing, its powerful capability of hardware access is also utilized. By specifying paths of the input video into 0, one could activate the camera system through OpenCV and start tracking automatically. The video in the prompted window will be synchronized with the captures of the camera. Feature trajectories are also plotted in the same way as any other offline videos. In this way, the program allows live tracking with hardware equipped with camera systems.

3. Results and Evaluations

3.1 Primitive Results

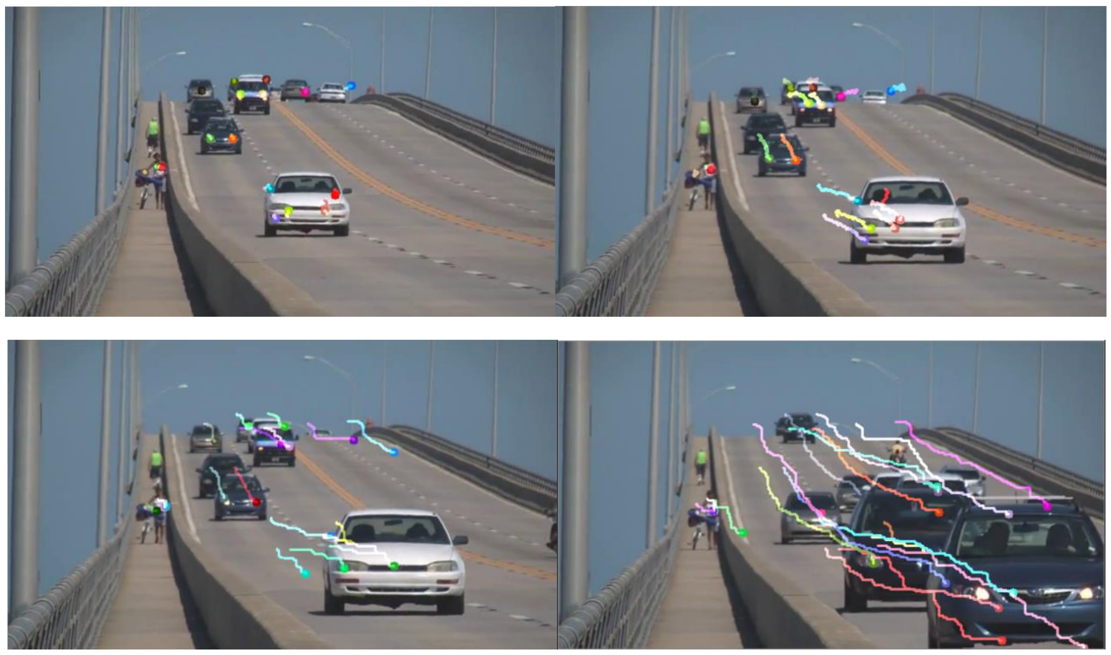


Figure 2: preliminary results of the program.

As shown in Figure 2, the program stands for its efficacy generally. It succeeds in tracking identified feature points continuously, with their trajectories plotted with different colors. However, although most of the features are accurately tracked, outliers still exist. Some of them lose track of the original features during the video and stick to other pixels by mistake, others just meander around when the feature disappears. The program is set up on a hardware of Intel(R) Core i7-7700HQ @2.80GHz, Nvidia GTX1060 (6 gigabytes of memory) and a DDR4 random access memory unit of 16 gigabytes. It achieves a speed of 8.33 frames per second, which is acceptable but far from the standard of real-time tracking.

3.2 Ablation Test

The ablation test investigates the effect of individual components to the overall performance of the program. There are major parts to be examined, including the Harris corner detection, feature clustering, iterative Lucas-Kanade optic flow (ILK), and the multi-resolution optic flow.

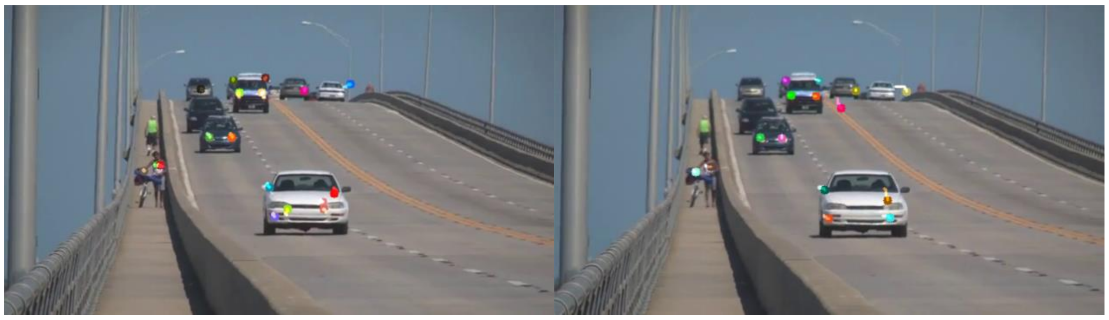
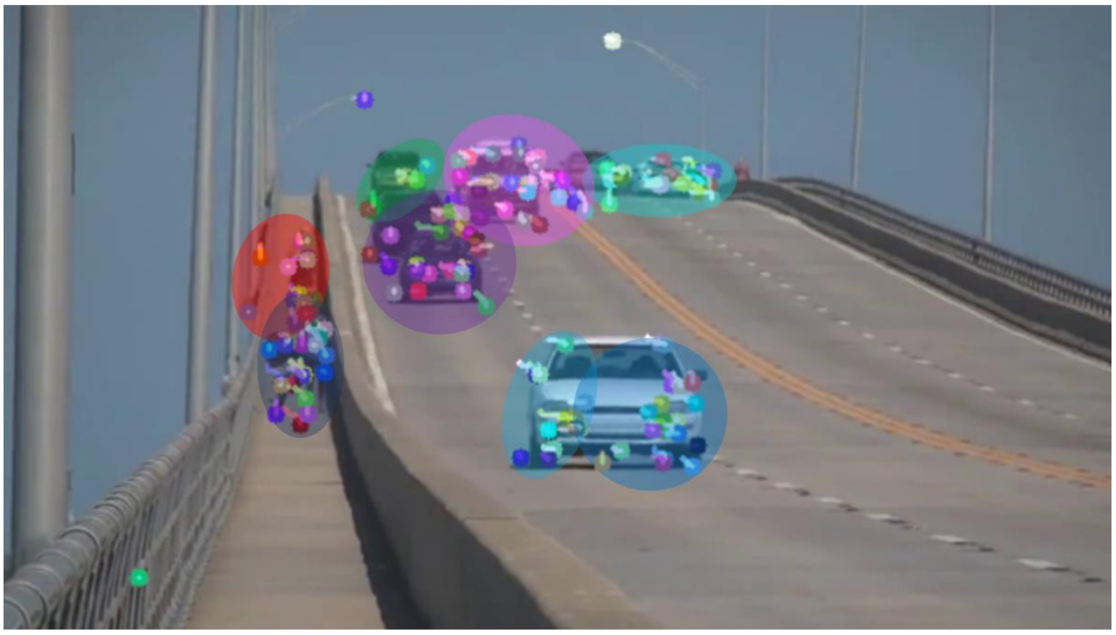


Figure 3: corner detections of Harris detector (left) and Shi-Tomas detector (right).

For corner detections, the self-implemented Harris corner detector is compared to the Shi-Tomas detector from the OpenCV library. It turns out that two algorithms deliver resemble performances when thresholds are properly set. As shown in Figure 3, preferable feature points for both algorithms are acute corners and spots where intensities of adjacent pixels vary drastically, such as the boundary of a van, and the headlight of a car. However, the Harris corner detector is significantly slower than the Shi-Tomas detector, as the former one takes averagely 0.1192 seconds to finish on a 640*360 image whereas the latter one only takes averagely 0.0077 seconds. Although varying in time span, the choice of feature detector has no substantial impact on overall efficiency of the program since it is only leveraged once for the first frame of the video.



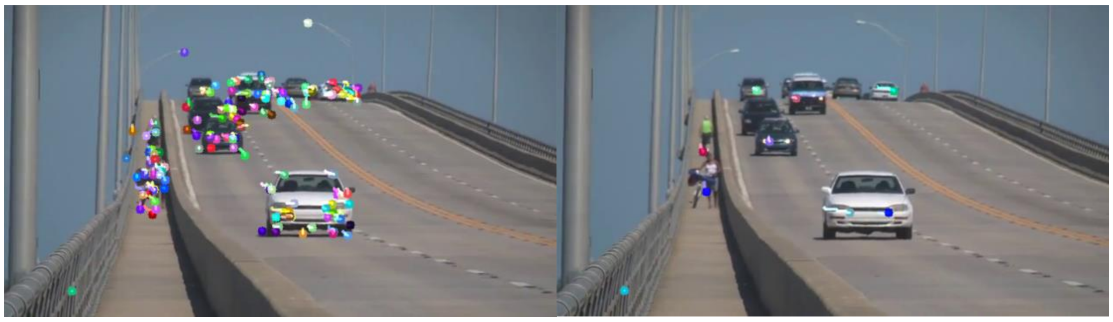


Figure 4: features before clustering (lower-left), during clustering (upper) after clustering with equivalent points (lower-right).

The comparison between clustered features with equivalent points and un-clustered features are shown in Figure 4. It could be observed that one object is only assigned with one feature point after applying the K-means, except for those whose boundaries are blur or whose scale is adequately large. This simplifies the outlook of the window and reduces the burden of calculation, as the frame rate increases from 5.60 to 11.53 frames per second. However, the accuracy and robustness of this strategy is still open to improvement. As an unsupervised machine learning method, clustering algorithms estimate objects rather than intentionally identify objects. This inevitably generates outliers under some circumstances. Therefore, deep learning frameworks could be involved for object identification as a future work.

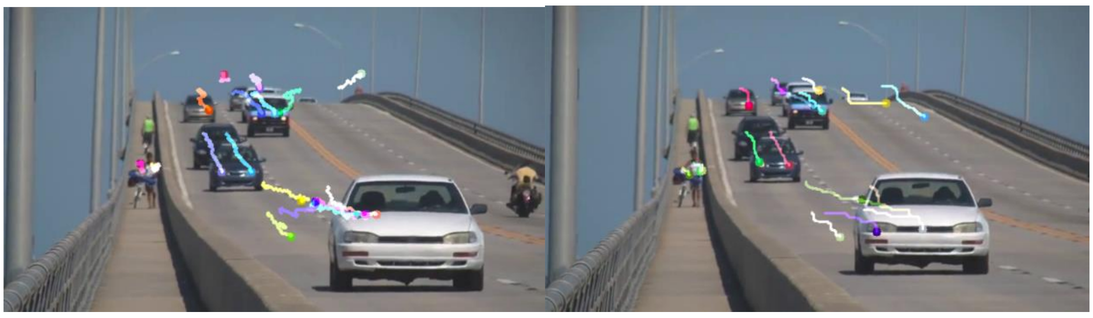


Figure 5: results without ILK (left) and with ILK (right).

Iterative Lucas-Kanade optic flow aims at fixing up tiny movements to increase tracking accuracies. Figure 5 demonstrates the results from the program running with and without the iterative fashion. Differences from the two approaches could barely be identified under mild circumstances. However, when the movement is complex or the velocity is fast, the ILK becomes to stand for its priority. For example, the rudimentary approach gradually loses track of the tracking points on the white car as it cannot keep up with the pace of its speed of movement, but the ILK still maintains the tracking under similar conditions. One possible reason to explain this is that errors could be accumulated. As mentioned, basic LK deprecates the higher order derivatives for simplicity. The deprecated term could be accumulated temporally, finally causes visible deviations. However, ILK iterates to fix up u and v with higher order derivatives, which reduces the error.

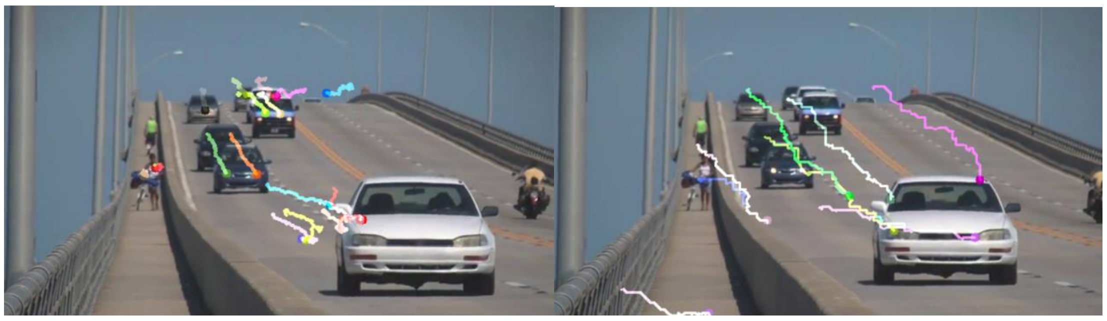


Figure 6: results of ILK without pyramid fashion (left) and with pyramid fashion (right).

The pyramid fashion of ILK algorithm is implemented to tackle with large movements, with its comparison of results with naïve ILK illustrated in Figure 6. It could be noticed that the implementation allows the program to deal with long-range movements, as all feature points on the white car (white, yellow, and pink) are properly tracked throughout the whole video even though the velocities of pixels speed up at the end. On the contrary, the rudimentary ILK implementation loses the track in the later period of the video. However, it could also be observed that the program is no longer sensitive to the small movements. The green line, for example, steps out of the car which is abnormal. This may be due to the upscaling of magnitudes of u and v in each level of the pyramid, which causes threshold problems in the position updating stage.

4. Discussions

Multiple techniques from computer vision contribute to this project with their advances and drawbacks intertwining. The Harris corner detector delivers features with high quality. Corners contain abundant information, providing robustness in rotations, variance in illuminations, sheltering conditions, and geometric transformations. However, this algorithm is vulnerable to scaling variances with a relatively high time complexity. Shi-Tomas corner detector could be a better replacement. K-means, as a naïve approach in machine learning, is intuitive and explicable. However, it is difficult to specify a proper k value when the program is blind to high-level information of the video. Additionally, K-means could fall into local optima with the influences of noise and outliers. An object identification model based on deep leaning approaches could be a better alternative. The iterative Lucas-Kanade approach and pyramid Lucas-Kanade approach both complex the system, as they involve processes of iterative refinements, which slows down the program in running time. However, they also contribute to the accuracy and robustness of the system, improving tracking performances.