Program 5 Report

Ellis Chen

ECPE 124

Abstract— **This report explains the procedures and results of various digital image processing techniques used to track selected features within an image and track the motion of these features across successive images.**

I. Lucas Kanade Algorithm Explained

The Lucas-Kanade algorithm is a method used to estimate how interesting features within an image moves across a series of consecutive images. This method works by comparing the feature points of the current image with the next image, and guessing in which direction the feature point will move.

Since this method is an optical flow algorithm, the intensity levels between successive images should change smoothly. In other words, the intensity level should not rise or drop sharply in any direction. It is also important to note that the movement of the feature point, will also exhibit movement of its surroundings, so there should be little to no displacement between successive images. So, to make this algorithm work properly, the transition from one image to another cannot change drastically.

To perform the algorithm, the current image needs to be smoothed over. This will produce a magnitude image that is feasible enough to perform the optical flow algorithm. The optical flow algorithm is an implementation of a mathematical linear system used to solve the change in direction of the feature points. The algorithm does this by looking within a "window" of the successive images and as the "window" moves across both images, the algorithm will "guess" in which direction the feature point will move to. This process is known as interpolation, and the direction at which the feature moves is used in solving the linear system mentioned previously.

The linear system stems from the optical flow constraint where the change in pixel intensity over a certain amount of displacement is equal to the sum of the products of the change in the vertical and horizontal pixel location and intensity changes. Solving the linear system results in an incremental displacement vector that is used to update the location of the feature point of interest to the next image.

II. Lucas Kanade Implementation

The implementation of the Lucas-Kanade optical flow algorithm is broken down into multiple steps as follows:

1) The MATLAB function ginput() was used to extract the specified number of feature points. These interesting features are cherrypicked by the user. The features are stored into a vector that expands for every feature point added. The cherrypicked points are highlighted by an 'x'.
2) The Lucas-Kanade function call occurs for each successive image within the same path directory, so the function arguments take in updated feature points and new images.
3) The current image is smoothed over with a gaussian and gaussian derivative kernel
4) The gradient matrix is computed for each feature point by interpolating the horizontal and vertical intensity changes
5) The error vector is computed by interpolating the gradient images and successive images to determine the margin of error between the two images
6) The gradient matrix and error vector are solved by linear algebra techniques and outputs a displacement vector
7) The vector is used to update the feature point coordinates until the change is insignificant or the number of times the vector is updated exceeds the threshold count
8) The Lucas-Kanade function returns the updated feature point locations for the successive image and the process repeats until there are no more successive image available

III. Tracking Results

Figure 1: 21 cherrypicked feature points using flower garden image as test case

Figure 2: Tracking results of selected features after 29 frames of flower garden

Comparing the before and after images of flower garden movie, there are some feature points that were properly tracked and some that were not. The features that were lost mainly were mainly the ones that were cherrypicked to the left of the tree. As the tree moved across the frames, the feature points to the left of the tree were lost, which explains why the features a jumbled together on the tree. Features that were tracked correctly were the corners of the tree branches, maybe some of the leaves on the ground and corners along the houses. These results show that cherry-picking corners of objects in the image provide better tracking results than smooth objects and edges.

Figure 3: Results of feature points when smoothing images before Lucas Kanade algorithm

The results in figure 3 have the same cherry-picked points as in figure 1. Visually, the transitions of the feature points seem sharper. Smoothing the images first before passing them as parameters to the lucas-kanade function causes the feature point locations to jump more sporadically than without smoothing the image first. Comparing the results of figure 3 to figure 2, figure 2 seems to output better tracking results.

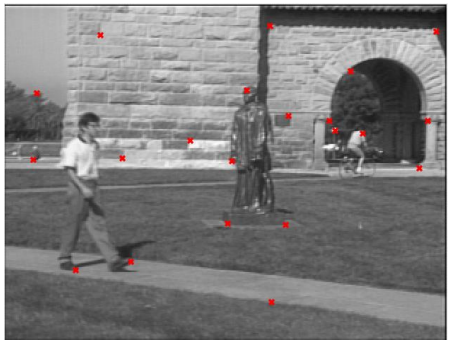

Figure 4: 21 cherrypicked feature points using statue image as a test case

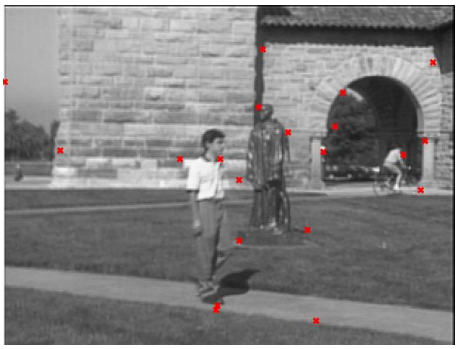

Figure 5: Tracking results of selected feature points after 31 frames of statue images

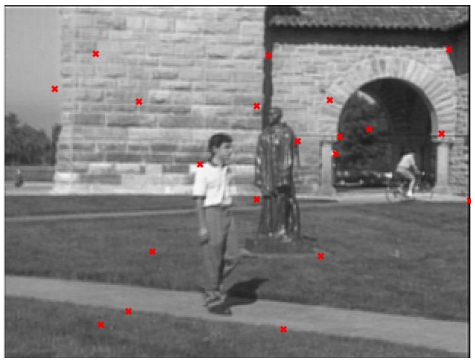

Figure 6: Results of feature points when smoothing images before Lucas Kanade algorithm

The features in figure 5 were able to successfully track the human walking, the statue, the building structure, and the human in the bike. Most of the features, if not all, were very much tracked throughout the whole sequence, and almost all of the feature points picked from figure 4 are corners. This again shows that corners are good features to track.

Smoothing the images first resulted in poorer tracking. Similar to figure 3, figure 6 shows that the tracking is more chaotic, in which the transition between frames resulted in more drastic changes in the feature point locations. By the end of the sequence, the only visually tracked features were some of the statue and building structures.