# 16-720 Computer Vision Spring 2016

# Azarakhsh Keipour (akeipour@andrew) Assignment 3

## Q 1.1 Lucas-Kanade Derivation

We have to find *u* and *v* that minimize the following cost function:



By linearizing  by first order Taylor expansion around the point  we have:



In which  and  are partial derivatives of  at point . Now by replacing  in the above cost function, we have:



Therefore, converting the equation to the matrix form, we need to compute:



To solve this system, we have:



Where:



and



The  is shown above with green color. It is the Gauss-Newton approximation to the Hessian matrix. It is not important if individual  matrices for the points have certain conditions or not; however, the  matrix should not be singular (should be invertible). Also if it is near to singular conditions (determinant is near to zero), then numerical error in our estimation of *u* and *v* will be high and the template offset cannot be calculated reliably.

## Q 1.2 Lucas-Kanade Algorithm

For each iteration we want to find ,  such that the following cost function is minimized:



To simplify the notation, I will use  instead of . By linearizing  by first order Taylor expansion around point  we have:



In which  and  are partial derivatives of  at point . By replacing it in the cost function above we have:



To simplify the notation, I will use *I* instead of  and  instead of  :



To find the minimum, differentiate w.r.t.  and:





Rewriting the above equations in matrix notation and eliminating the constant multiplier 2, we have:



The code that implements this formulation is provided as requested. To increase speed, I used efficient matrix operations in the algorithm. I also implemented the step-by-step algorithm Lucas-Kanade algorithm of [1] and the Inverse Compositional algorithm of [2]. All the codes work in real-time; however, I did not find much difference in performance between the three implementations. I commented the first two implementations and let the Inverse Compositional version stay uncommented.

## Q 1.3 Testing Lucas-Kanade Algorithm

The results are shown in Figure 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| C:\Users\Azarakhsh\Desktop\CMU\Vision\Homework 3\My Code\results\q3_2_frame_1.jpg | C:\Users\Azarakhsh\Desktop\CMU\Vision\Homework 3\My Code\results\q3_2_frame_100.jpg | C:\Users\Azarakhsh\Desktop\CMU\Vision\Homework 3\My Code\results\q3_2_frame_200.jpg | C:\Users\Azarakhsh\Desktop\CMU\Vision\Homework 3\My Code\results\q3_2_frame_300.jpg | C:\Users\Azarakhsh\Desktop\CMU\Vision\Homework 3\My Code\results\q3_2_frame_400.jpg |

Figure 1. The result of Lucas-Kanade tracking algorithm on frames 1, 100, 200, 300 and 400 of the given sequence.

## Q 1.4 Lucas-Kanade Algorithm with Template Correction

The results are shown in Figure 2. It is clearly shown how the corrected template (yellow box) is able to track the car without a drift, while the normal Lucas-Kanade template (green box) drifts from the correct position after a while.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| C:\Users\Azarakhsh\Desktop\CMU\Vision\Homework 3\My Code\results\q1_4_frame_1.jpg | C:\Users\Azarakhsh\Desktop\CMU\Vision\Homework 3\My Code\results\q1_4_frame_100.jpg | C:\Users\Azarakhsh\Desktop\CMU\Vision\Homework 3\My Code\results\q1_4_frame_200.jpg | C:\Users\Azarakhsh\Desktop\CMU\Vision\Homework 3\My Code\results\q1_4_frame_300.jpg | C:\Users\Azarakhsh\Desktop\CMU\Vision\Homework 3\My Code\results\q1_4_frame_400.jpg |

Figure 2. The result of Lucas-Kanade tracking algorithm with template correction (yellow box) vs the simple Lucas-Kanade algorithm (green box) of Q1.3 on frames 1, 100, 200, 300 and 400 of the given sequence.

## Q 2.1 Lucas-Kanade Tracking with Appearance Basis

We want to find weights  () in the following equation:



Since the bases are orthogonal and of the same size, to calculate a weight  we can multiply both sides of the equation by :



Therefore, for each  we can compute  as .

## Q 2.2-2.3 Implementation of L-K Tracking with Appearance Basis

The results are shown in Figure 3. On the given sequence, both the simple Lucas-Kanade and the Lucas-Kanade with Appearance Basis algorithms can track the toy perfectly.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| C:\Users\Azarakhsh\Desktop\CMU\Vision\Homework 3\My Code\results\q2_3_frame_1.jpg | C:\Users\Azarakhsh\Desktop\CMU\Vision\Homework 3\My Code\results\q2_3_frame_200.jpg | C:\Users\Azarakhsh\Desktop\CMU\Vision\Homework 3\My Code\results\q2_3_frame_300.jpg | C:\Users\Azarakhsh\Desktop\CMU\Vision\Homework 3\My Code\results\q2_3_frame_350.jpg | C:\Users\Azarakhsh\Desktop\CMU\Vision\Homework 3\My Code\results\q2_3_frame_400.jpg |

Figure 3. The result of Lucas-Kanade tracking algorithm with appearance basis (yellow box) of Q 2.2 vs. the simple Lucas-Kanade algorithm (green box) of Q1.3 on frames 1, 200, 300, 350 and 400 of the given sequence.

The results on the faster FPS are shown in Figure 4. With faster FPS, the algorithm skips 4 frames to obtain the next frame (i.e. it tracks the toy in frames 1, 6, 11, 16, etc.) in order to provide a harder problem to be able to show differences between the two algorithms.

On the given sequence, with faster FPS, the simple Lucas-Kanade loses the toy in the middle of the sequence, while the Lucas-Kanade with Appearance Basis algorithm can track the toy without losing it.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| C:\Users\Azarakhsh\Desktop\CMU\Vision\Homework 3\My Code\results\q2_3_frame_fps5_1.jpg | C:\Users\Azarakhsh\Desktop\CMU\Vision\Homework 3\My Code\results\q2_3_frame_fps5_201.jpg | C:\Users\Azarakhsh\Desktop\CMU\Vision\Homework 3\My Code\results\q2_3_frame_fps5_301.jpg | C:\Users\Azarakhsh\Desktop\CMU\Vision\Homework 3\My Code\results\q2_3_frame_fps5_351.jpg | C:\Users\Azarakhsh\Desktop\CMU\Vision\Homework 3\My Code\results\q2_3_frame_fps5_401.jpg |

Figure 4. The result of Lucas-Kanade tracking with appearance basis (yellow box) of Q 2.2 vs. the simple Lucas-Kanade algorithm (green box) of Q1.3 on frames 1, 200, 300, 350 and 400 of the given sequence with faster FPS.

## Q 3.1 – 3.3 Affine Motion Subtraction

The results are shown in Figure 5. I used the warping function (warpH) from previous homeworks to warp images.

|  |  |  |  |
| --- | --- | --- | --- |
| C:\Users\Azarakhsh\Desktop\CMU\Vision\Homework 3\My Code\results\q3_3_frame_30.jpg | C:\Users\Azarakhsh\Desktop\CMU\Vision\Homework 3\My Code\results\q3_3_frame_60.jpg | C:\Users\Azarakhsh\Desktop\CMU\Vision\Homework 3\My Code\results\q3_3_frame_90.jpg | C:\Users\Azarakhsh\Desktop\CMU\Vision\Homework 3\My Code\results\q3_3_frame_120.jpg |

Figure 5. The result of Lucas-Kanade affine algorithm on frames 30, 60, 90 and 120 of the given sequence.