# 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.