16-720 Computer Vision: Homework 3
Lucas-Kanade Tracking & Background Subtraction

Anbang Hu
anbangh@andrew.cmu.edu
Due: March 3, 2016

1 Lucas-Kanade Tracking

Q1.1 (5 points): In the following, we use ∂_x to denote $\frac{\partial}{\partial x}$.
With first-order Taylor approximation, we have

$$I_{t+1}(x+u, y+v) \approx I_{t+1}(x, y) + u\partial_x I_{t+1}(x, y) + v\partial_y I_{t+1}(x, y) \tag{1}$$

Substitute Eq(1) into the target function, we obtain

$$J(u, v) = \sum_{(x,y)\in R_t} \left(u\partial_x I_{t+1}(x, y) + v\partial_y I_{t+1}(x, y) + I_{t+1}(x, y) - I_t(x, y)\right)^2 \tag{2}$$

Taking derivatives of J with respect to u and v, we have

$$\partial_u J(u, v) = \sum_{(x,y)\in R_t} 2\partial_x I_{t+1}(x, y)(u\partial_x I_{t+1}(x, y) + v\partial_y I_{t+1}(x, y) + I_{t+1}(x, y) - I_t(x, y)) \tag{3}$$

$$\partial_v J(u, v) = \sum_{(x,y)\in R_t} 2\partial_y I_{t+1}(x, y)(u\partial_x I_{t+1}(x, y) + v\partial_y I_{t+1}(x, y) + I_{t+1}(x, y) - I_t(x, y)) \tag{4}$$

To simplify the notation, we let $I_{t+1} = I_{t+1}(u, v)$ and omit the summation range in the following derivation. Equating Eq(3,4) to zero, we get

$$\sum u(\partial_x I_{t+1})^2 + \sum v\partial_x I_{t+1}\partial_y I_{t+1} = -\sum \partial_x I_{t+1}(I_{t+1} - I_t) \tag{5}$$

$$\sum u\partial_x I_{t+1}\partial_y I_{t+1} + \sum v(\partial_y I_{t+1})^2 = -\sum \partial_y I_{t+1}(I_{t+1} - I_t) \tag{6}$$

Eq(5,6) can be put into matrix form:

$$\begin{bmatrix} \sum(\partial_x I_{t+1})^2 & \sum \partial_x I_{t+1}\partial_y I_{t+1} \\ \sum \partial_x I_{t+1}\partial_y I_{t+1} & \sum(\partial_y I_{t+1})^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} \sum \partial_x I_{t+1}(I_{t+1} - I_t) \\ \sum \partial_y I_{t+1}(I_{t+1} - I_t) \end{bmatrix} \tag{7}$$

From Eq(7), we can see that

$$A = \begin{bmatrix} \sum(\partial_x I_{t+1})^2 & \sum \partial_x I_{t+1}\partial_y I_{t+1} \\ \sum \partial_x I_{t+1}\partial_y I_{t+1} & \sum(\partial_y I_{t+1})^2 \end{bmatrix}$$

$$\Delta p = \begin{bmatrix} u \\ v \end{bmatrix}$$

$$b = -\begin{bmatrix} \sum \partial_x I_{t+1}(I_{t+1} - I_t) \\ \sum \partial_y I_{t+1}(I_{t+1} - I_t) \end{bmatrix}$$

We can further compute $A^T A$:

$$A^T A = \begin{bmatrix} \left(\sum(\partial_x I_{t+1})^2\right)^2 + \left(\sum \partial_x I_{t+1}\partial_y I_{t+1}\right)^2 & \sum \partial_x I_{t+1}\partial_y I_{t+1}\left(\sum(\partial_x I_{t+1})^2 + \sum(\partial_y I_{t+1})^2\right) \\ \sum \partial_x I_{t+1}\partial_y I_{t+1}\left(\sum(\partial_x I_{t+1})^2 + \sum(\partial_y I_{t+1})^2\right) & \left(\sum \partial_x I_{t+1}\partial_y I_{t+1}\right)^2 + \left(\sum(\partial_y I_{t+1})^2\right)^2 \end{bmatrix} \tag{8}$$

$A^T A$ must be invertible (i.e. nonsingular) matrix so that so that the template offset can be calculated reliably.

Q1.2 (15 points): Implemented in `LucasKanade.m`. Note that the inverse compositional version of the Lucas-Kanade tracker is implemented.

Q1.3 (10 points): Implemented in `testCarSequence.m`. Figure 1 shows the result.

Figure 1: From left to right shows results for frame 1, 100, 200, 300, 400

Q1.4 (Extra credit, 10 points): Implemented in `testCarSequenceWithTemplateCorrection.m`. Figure 2 shows the performance after template drifting correction. We can see that there is still some degree of drifting, but the result is much better than that in **Q1.3**.

Figure 2: From left to right shows results for frame 1, 100, 200, 300, 400

2 Lucas-Kanade Tracking with Appearance Basis

2.1 Appearance Basis

The performance of the baseline implementation on `sylvseq.mat` either with or without template drifting correction is disastrous, since the object being tracked now is subject to drastic appearance variance.

Q2.1 (5 points): We can write the equation as

$$I_{t+1} - I_t = B\mathbf{w} \tag{9}$$

where B is a matrix, whose columns corresponds to bases. Images I_{t+1} and I_t are viewed in vector form, i.e. assembled as a column vector.

By rewriting Eq(9), we obtain

$$\mathbf{w} = B^{-1}(I_{t+1} - I_t) = (B^T B)^{-1} B^T (I_{t+1} - I_t) \tag{10}$$

Since B is a set of orthogonal image bases, we know that $B^T B$ is a diagonal matrix, which is invertible. Then we can express each w_c in $\mathbf{w}$ as the cth row of $(B^T B)^{-1} B^T (I_{t+1} - I_t)$.

2.2 Tracking

Q2.2 (15 points): Implemented in `LucasKanadeBasis.m`. During the implementation, I consulted section III A of paper

`http://www.patricklucey.com/Site/Publications_files/MMSP_2008_2.pdf`

Q2.3 (15 points): Implemented in `testSylvSequence.m`. The result is shown in Figure 3. Yellow boxes show results from LucasKanadeBasis and green boxes show results from LucasKanade. They are pretty close mainly because the image set is not challenging enough. When I reduce the frame rate to $\frac{1}{8}$ of the original, the last frame shows that yellow box wins, i.e. LucasKanadeBasis is better. (See Figure 4)

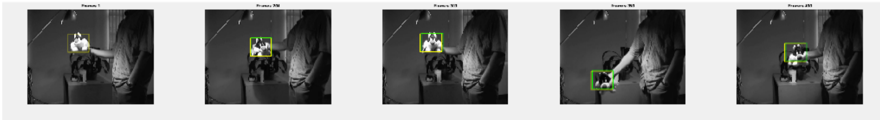

Figure 3: From left to right shows results for frame 1, 200, 300, 350, 400

Figure 4: Last frame from reducing frame rate to $\frac{1}{8}$ of the original (it shows frame 56 because dataset is shrunk)

3 Affine Motion Subtraction

3.1 Dominant Motion Estimation

Q3.1 (15 points): Implemented in `LucasKanadeAffine.m`.

3.2 Moving Object Detection

Q3.2 (10 points): Implemented in `SubtractDominantMotion.m`.

Q3.3 (10 points): Implemented in `testAerialSequence.m`. The result is shown in Figure 5.

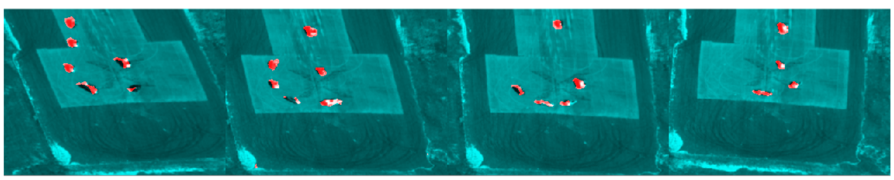

Figure 5: From left to right shows results for frame 30, 60, 90, 120