Computer Vision Homework 3

Ruixin Liu

3/22/2019

Q1.1:

This question could be treated as given equations below, we would like to find Δp that minimizes the first equation:

$$p^* = argmin_p = \sum_{x \in N} \|I_{t+1}(x+p) - I_t(x)\|_2^2$$

$$I_{t+1}(x' + \Delta p) \approx I_{t+1}(x') + \frac{\delta I_{t+1}(x')}{\delta x'^T} \frac{\delta W(x;p)}{\delta p^T} \Delta p$$

Minimizing the first equation is a nonlinear optimization problem, and to simplify such a problem we could solve for increments to current estimate with the equation below:

$$E = \sum_x [I_{t+1}(x+p) - I_t(x)]^2 = \sum_x \left[I_{t+1}(x' + \Delta p) + \frac{\delta I_{t+1}(x')}{\delta x'^T} \frac{\delta W(x;p)}{\delta p^T} \Delta p - I_{t+1}(x) \right]^2$$

Writing equation above in the form of $argmin_{\Delta p} \sum_x \|A\Delta p - b\|_2^2$:

$$argmin_{\Delta p} E = argmin_{\Delta p} \sum_x \left\| \frac{\delta I_{t+1}(x')}{\delta x'^T} \frac{\delta W(x;p)}{\delta p^T} \Delta p - \left(I_{t+1}(x) - I_{t+1}(x' + \Delta p) \right) \right\|_2^2$$

$\frac{\delta W(x;p)}{\delta p^T}$ stands for the Jacobian of warped image, which is helpful in calculating this equation as well as the Hessian matrix which will be derived below. A is $\frac{\delta I_{t+1}(x')}{\delta x'^T} \frac{\delta W(x;p)}{\delta p^T}$ (steepest descent) and b is $\left(I_{t+1}(x) - I_{t+1}(x' + \Delta p) \right)$(difference between warped and template image).

Setting the derivative of $\sum_x \|A\Delta p - b\|_2^2$ to zero, we have:

$$\sum_x 2A^T(A\Delta p - b) = 0$$

In order to calculate Δp from such a form, A needs to be invertible and its determinant should be non-zero to make sure that there exists one unique solution.

Q1.3:

The results of Question 1.3 are shown below. It can be observed that Lucas-Kanade slowly loses track of the vehicle as error accumulates. This issue will be resolved in Question 1.4.

Figure 1: Results of Question 1.3.

Q1.4:

The results of Question 1.4 are shown in Figure 2. The blue boxes are Lucas-Kanade tracking without template correction and and the red boxes contain template correction. The results are much more accurate with template correction. However, it is also noticed that the run time of the algorithm with template correction is almost twice as the run time of the other one. This is because Lucas-Kanade is performed twice in one frame. Given that there are only 415 frames, the run time is not long (around 20 seconds).

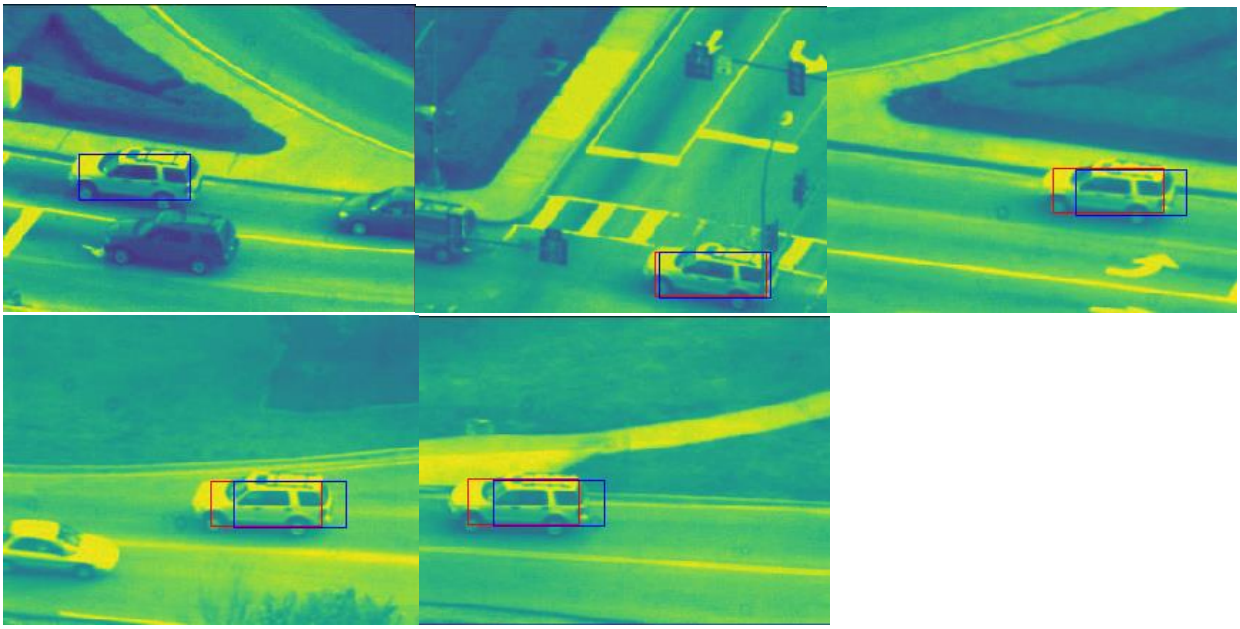

Figure 2: Results of Question 1.4.

Q2.1:

The given equation is in the form:

$$I_{t+1}(x) = I_t(x) + \sum_{k=1}^{K} w_k B_k(x)$$

And we would like to write w as a function of the other variables. According to the given hint that B_k are orthogonal to each other, which means their dot products are zero, we first move $I_t(x)$ to the lefthand side and then multiply both sides by one of the bases, say B_i:

$$B_i\big(I_{t+1}(x) - I_t(x)\big) = B_i \sum_{k=1}^{K} w_k B_k(x)$$

$\sum_{k=1}^{K} w_k B_k(x)$ contains $w_i B_i$ and all other bases, and multiplying these bases with B_i makes them all equal to zero. The equation then becomes:

$$B_i\big(I_{t+1}(x) - I_t(x)\big) = B_i w_i B_i = \big||B_i|\big|^2 w_i$$

$$w_i = \frac{B_i\big(I_{t+1}(x) - I_t(x)\big)}{\big||B_i(x)|\big|^2 w_i}$$

Here, $i \epsilon \{1 \dots K\}$.

Q2.3:

Figure 3 shows the results of Question 2.3. The images should show two rectangles for normal Lukas-Kanade and Lukas-Kanade with bases. However, the difference between them are very small that they overlap with each other. Both trackers track the object well.

Figure 3: Results of Question 2.3.

Q3.3:

The results of Question 3.3 are shown below. The vehicles can be easily observed at the highlighted pixels in the images. The threshold to determine object movement is set to be 0.1, so that tracker eliminates most noises while keeps object tracking as good as possible.

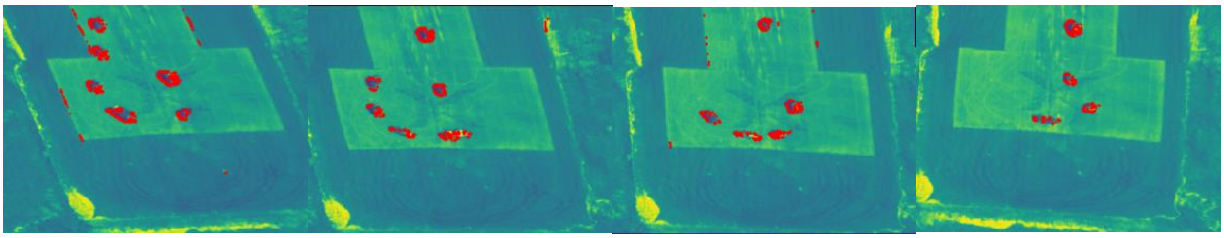

Figure 4: Results of Question 3.3.

Q4.1

For conventional Lucas-Kanade tracking, a big portion of computation is used in re-evaluating Hessian Matrix. On the other hand, in inverse compositional algorithm, Hessian is precomputed before iteration starts. This reduces Hessian calculation from N times to one time as well as steepest descent calculation. As a result, the run time in my implementation is reduced from 23 seconds to 15 seconds.