16-720B Homework 3 Write-up

Gu, Qiao

October 23, 2019

Q1.1

- $\frac{\partial \mathcal{W}(\mathbf{x};\mathbf{p})}{\partial \mathbf{p}^T}$ is the graident of the warped coordinates over the warping parameter $\mathbf{p}$, which is:

$$\frac{\partial \mathcal{W}(\mathbf{x};\mathbf{p})}{\partial \mathbf{p}^T} = \frac{\partial \mathbf{x}+\mathbf{p}}{\partial \mathbf{p}^T} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{1}$$

- For the iterative process, replace $\mathbf{p}$ with $\mathbf{p}+\Delta\mathbf{p}$ in Eq. (2) of the handout, and then

$$\begin{aligned} \mathcal{I}_{t+1}(\mathbf{x}+\mathbf{p}+\Delta\mathbf{p}) - \mathcal{I}_t(\mathbf{x}) &= \mathcal{I}_{t+1}(\mathbf{x}+\mathbf{p}) + \frac{\partial \mathcal{I}_{t+1}(\mathbf{x}+\mathbf{p})}{\partial(\mathbf{x}+\mathbf{p})^T}\Delta\mathbf{p} - \mathcal{I}_t(\mathbf{x}) & (2)\\ &= \nabla\mathcal{I}_{t+1}(\mathbf{x}+\mathbf{p})\Delta\mathbf{p} - (\mathcal{I}_t(\mathbf{x}) - \mathcal{I}_{t+1}(\mathbf{x}+\mathbf{p})). & (3) \end{aligned}$$

  Therefore the Eq. 2 of the handout in vector form is (Note that each $\nabla\mathcal{I}_{t+1}(\mathbf{x}+\mathbf{p})$ are of shape 1×2.)

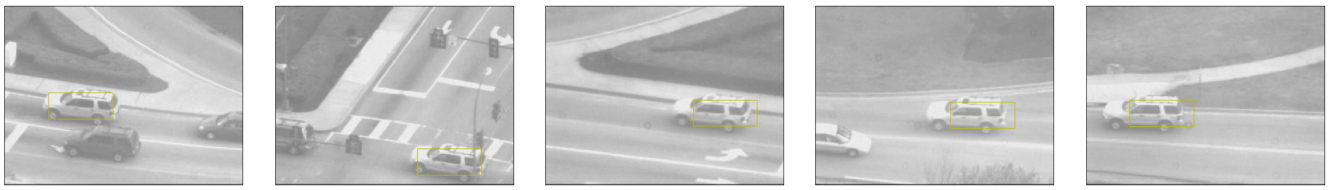
$$\arg\min_{\Delta\mathbf{p}} \left\| \begin{bmatrix} \nabla\mathcal{I}_{t+1}(\mathbf{x}_1+\mathbf{p}) \\ \nabla\mathcal{I}_{t+1}(\mathbf{x}_2+\mathbf{p}) \\ \cdots \\ \nabla\mathcal{I}_{t+1}(\mathbf{x}_D+\mathbf{p}) \end{bmatrix} \Delta\mathbf{p} - \begin{bmatrix} \mathcal{I}_t(\mathbf{x}_1) - \mathcal{I}_{t+1}(\mathbf{x}_1+\mathbf{p}) \\ \mathcal{I}_t(\mathbf{x}_2) - \mathcal{I}_{t+1}(\mathbf{x}_2+\mathbf{p}) \\ \cdots \\ \mathcal{I}_t(\mathbf{x}_D) - \mathcal{I}_{t+1}(\mathbf{x}_D+\mathbf{p}) \end{bmatrix} \right\| = \arg\min_{\Delta\mathbf{p}} \|\mathbf{A}\Delta\mathbf{p} - \mathbf{b}\| \tag{4}$$

  The big matrix and the big vector on the L.H.S. of the above equation are the $\mathbf{A}$ and $\mathbf{b}$.

- To solve for the least square solution of Eq. 4, we need to compute $(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$. Therefore, we must have $\mathbf{A}^T\mathbf{A}$ to be invertible.

Q1.3

Please find the results of Lucas-Kanade tracking results in Figure. 1



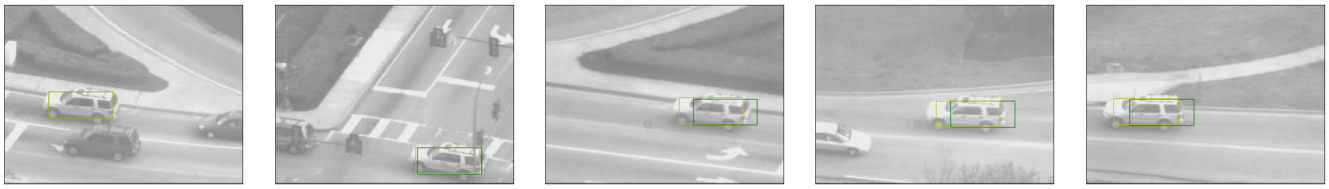
(a) frame 1 (b) frame 100 (c) frame 200 (d) frame 300 (e) frame 400

Figure 1: Lucas-Kanade Tracking Results with One Single Template.

Q1.4

Please note that for the implementation for this question, I wrote a function `LucasKanadeTrackerWithTemplateCorrec` in `LucasKanade.py`, which handles the routine of Lucas Kanada tracking with template correction.

Please find the results from Lucas-Kanade Tracking with template correction in Figure. 2, with the comparison to that without template update. We can clearly see that the tracking with template correction yields a better result for later frames.



(a) frame 1 (b) frame 100 (c) frame 200 (d) frame 300 (e) frame 400

Figure 2: Lucas-Kanade Tracking Results with Template Correction (in yellow boxes). The performance of the baseline tracker in **Q1.3** is in green boxes.

Q2.1

From the Eq. 6 of the handout, for w_i

$$\mathcal{B}_i^T(\mathbf{x})(\mathcal{I}_{t+1}(\mathbf{x}) - \mathcal{I}_t(\mathbf{x})) = \mathcal{B}_i^T(\mathbf{x})(\sum_{k=1}^{K} w_k \mathcal{B}_k(\mathbf{x})) \tag{5}$$

$$\Rightarrow \mathcal{B}_i(\mathbf{x})^T(\mathcal{I}_{t+1}(\mathbf{x}) - \mathcal{I}_t(\mathbf{x})) = w_i \mathcal{B}_i^T(\mathbf{x})\mathcal{B}_i(\mathbf{x}) = w_i \tag{6}$$

$$\Rightarrow w_i = \mathcal{B}_i^T(\mathbf{x})(\mathcal{I}_{t+1}(\mathbf{x}) - \mathcal{I}_t(\mathbf{x})). \tag{7}$$

Therefore, for the vector $\mathbf{w}$

$$\mathbf{w} = \begin{bmatrix} \mathcal{B}_1^T(\mathbf{x}) \\ \mathcal{B}_2^T(\mathbf{x}) \\ \cdots \\ \mathcal{B}_K^T(\mathbf{x}) \end{bmatrix} (\mathcal{I}_{t+1}(\mathbf{x}) - \mathcal{I}_t(\mathbf{x})). \tag{8}$$

Q2.2

The optimization task can be converted to:

$$\arg\min_{\mathbf{\Delta p}} \|\mathbf{B}^{\perp}(\mathbf{A}\mathbf{\Delta p} - \mathbf{b})\|_2^2 = \arg\min_{\mathbf{\Delta p}} \|(\mathbf{I} - \mathbf{B}\mathbf{B}^T)\mathbf{A}\mathbf{\Delta p} - (\mathbf{I} - \mathbf{B}\mathbf{B}^T)\mathbf{b}\|_2^2. \tag{9}$$

And the above problem can be solved as a least square problem.

Q2.3

Please find the results from Lucas-Kanade Tracking with appearance basis in Figure. 3, with the comparison to the baseline tracker in **Q1.3**. We can see that the performance of the two trackers are almost the same, and their bounding boxes basically overlaps with each other. This is probably because as the baseline tracker update its template in every iteration, it can account for some changes of template appearance.

(a) frame 1 (b) frame 200 (c) frame 300 (d) frame 350 (e) frame 400

Figure 3: Lucas-Kanade Tracking Results with Appearance Basis (in yellow boxes). The performance of the baseline tracker in **Q1.3** is in green boxes.

Q3.1

Previously, for only translation, the gradient of warping function $\mathcal{W}$ w.r.t the parameters $\mathbf{p}$, is in Eq. 1. Now, since

$$\mathcal{W}(\mathbf{x};\mathbf{p}) = \begin{bmatrix}(1+p_1)x + p_2 y + p_3 \\ p_4 x + (1+p_5)y + p_6\end{bmatrix} \tag{10}$$

$$\Rightarrow \frac{\partial \mathcal{W}(\mathbf{x};\mathbf{p})}{\partial \mathbf{p}^T} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix}. \tag{11}$$

Now go back to the Eq. 4 in the handout and Eq. 2:

$$\mathcal{I}_{t+1}(\mathcal{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p})) \approx \mathcal{I}_{t+1}(\mathbf{x}') + \frac{\partial \mathcal{I}_{t+1}(\mathbf{x}')}{\partial (\mathbf{x}')^T}\frac{\partial \mathcal{W}(\mathbf{x};\mathbf{p})}{\partial \mathbf{p}^T}\Delta\mathbf{p} \tag{12}$$

$$\Rightarrow \mathcal{I}_{t+1}(\mathcal{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p})) - \mathcal{I}_t(\mathbf{x}) \approx \mathcal{I}_{t+1}(\mathbf{x}') + \frac{\partial \mathcal{I}_{t+1}(\mathbf{x}')}{\partial (\mathbf{x}')^T}\frac{\partial \mathcal{W}(\mathbf{x};\mathbf{p})}{\partial \mathbf{p}^T}\Delta\mathbf{p} - \mathcal{I}_t(\mathbf{x}) \tag{13}$$

$$= \nabla \mathcal{I}_{t+1}(\mathbf{x}')\frac{\partial \mathcal{W}(\mathbf{x};\mathbf{p})}{\partial \mathbf{p}^T}\Delta\mathbf{p} - (\mathcal{I}_t(\mathbf{x}) - \mathcal{I}_{t+1}(\mathbf{x}')), \tag{14}$$

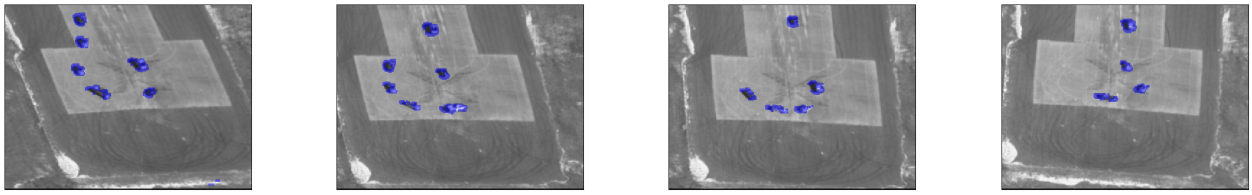
where $\mathbf{x}' = \mathcal{W}(\mathbf{x};\mathbf{p})$. Therefore, the optimzation object in Eq. 4 becomes:

$$\arg\min_{\Delta\mathbf{p}} \left\| \begin{bmatrix} \nabla \mathcal{I}_{t+1}(\mathbf{x}'_1)\frac{\partial \mathcal{W}(\mathbf{x}_1;\mathbf{p})}{\partial \mathbf{p}^T} \\ \nabla \mathcal{I}_{t+1}(\mathbf{x}'_2)\frac{\partial \mathcal{W}(\mathbf{x}_2;\mathbf{p})}{\partial \mathbf{p}^T} \\ \cdots \\ \nabla \mathcal{I}_{t+1}(\mathbf{x}'_D)\frac{\partial \mathcal{W}(\mathbf{x}_D;\mathbf{p})}{\partial \mathbf{p}^T} \end{bmatrix} \Delta\mathbf{p} - \begin{bmatrix} \mathcal{I}_t(\mathbf{x}_1) - \mathcal{I}_{t+1}(\mathbf{x}'_1) \\ \mathcal{I}_t(\mathbf{x}_2) - \mathcal{I}_{t+1}(\mathbf{x}'_2) \\ \cdots \\ \mathcal{I}_t(\mathbf{x}_D) - \mathcal{I}_{t+1}(\mathbf{x}'_D) \end{bmatrix} \right\| = \arg\min_{\Delta\mathbf{p}} \|\mathbf{A}_{affine}\Delta\mathbf{p} - \mathbf{b}\| \tag{15}$$

Q3.3

Please find the results of moving object detection in Figure. 4. Please note that I used a threshold of 0.1 in `SubtractDominantMotion()` to determine whether the difference is dominant.

ATTENTION: in the figures shown below, the masks have been dilated using a structure of 3×3 `np.ones()` matrix for better demostration, but the masks stored in `aerialseqrects.npy` are not dilated. And I have removed the false detection on the image border because of the camera motion.



(a) frame 30 (b) frame 60 (c) frame 90 (d) frame 120

Figure 4: Lucas-Kanade Tracking with Affine Motion Subtration.

Q4.1

Consider the inverse composition, the optimzation target in Eq. 12 becomes

$$\mathcal{I}_t(\mathcal{W}(\mathbf{x}, \mathbf{0} + \Delta\mathbf{p})) \approx \mathcal{I}_t(\mathbf{x}) + \frac{\partial \mathcal{I}_t(\mathbf{x})}{\partial \mathbf{x}^T} \frac{\partial \mathcal{W}(\mathbf{x}; \mathbf{0})}{\partial \mathbf{p}^T} \Delta\mathbf{p} \tag{16}$$

$$\Rightarrow \mathcal{I}_{t+1}(\mathcal{W}(\mathbf{x}; \mathbf{p})) - \mathcal{I}_t(\mathcal{W}(\mathbf{x}, \mathbf{0} + \Delta\mathbf{p})) = -(\frac{\partial \mathcal{I}_t(\mathbf{x})}{\partial \mathbf{x}^T} \frac{\partial \mathcal{W}(\mathbf{x}; \mathbf{0})}{\partial \mathbf{p}^T} \Delta\mathbf{p} - (\mathcal{I}_{t+1}(\mathbf{x}') - \mathcal{I}_t(\mathbf{x}))). \tag{17}$$

And then the optimzation object in Eq. 15 becomes

$$\arg\min_{\Delta\mathbf{p}} \left\| \begin{bmatrix} \nabla \mathcal{I}_t(\mathbf{x}_1) \frac{\partial \mathcal{W}(\mathbf{x}_1; \mathbf{0})}{\partial \mathbf{p}^T} \\ \nabla \mathcal{I}_t(\mathbf{x}_2) \frac{\partial \mathcal{W}(\mathbf{x}_2; \mathbf{0})}{\partial \mathbf{p}^T} \\ \cdots \\ \nabla \mathcal{I}_t(\mathbf{x}_D) \frac{\partial \mathcal{W}(\mathbf{x}_D; \mathbf{0})}{\partial \mathbf{p}^T} \end{bmatrix} \Delta\mathbf{p} - \begin{bmatrix} -\mathcal{I}_t(\mathbf{x}_1) + \mathcal{I}_{t+1}(\mathbf{x}'_1) \\ -\mathcal{I}_t(\mathbf{x}_2) + \mathcal{I}_{t+1}(\mathbf{x}'_2) \\ \cdots \\ -\mathcal{I}_t(\mathbf{x}_D) + \mathcal{I}_{t+1}(\mathbf{x}'_D) \end{bmatrix} \right\| = \arg\min_{\Delta\mathbf{p}} \|\mathbf{A}_{inv}\Delta\mathbf{p} - \mathbf{b}_{inv}\| \tag{18}$$

By solving the above equation, we can get $\Delta\mathbf{p}$ and thus $\Delta\mathbf{M}$. Then we can update $\mathbf{M} = \mathbf{M}(\Delta\mathbf{M})^{-1}$.

Compare to the method in section **Q3.1**, in the computation of A_{inv} in the inverse composition algothm, both $\nabla \mathcal{I}_t(\mathbf{x})$ and $\frac{\partial \mathcal{W}(\mathbf{x};\mathbf{0})}{\partial \mathbf{p}^T}$ do not depend on the value of $\mathbf{p}$, and thus not related to the update of $\mathbf{M}$. Therefore, the inverse composition method can pre-compute the matrix A_{inv} before iterations start, and the computation cost of each iteration can be reduced.

Q4.2

The expression

$$\frac{1}{2}\|\mathbf{y} - \mathbf{X}^T\mathbf{g}\|_2^2 + \frac{\lambda}{2}\|\mathbf{g}\|_2^2 \tag{19}$$

is convex. Take the gradient of the expression over $\mathbf{g}$ and make it equal to $\mathbf{0}$, we get

$$-\mathbf{X}(\mathbf{y} - \mathbf{X}^T\mathbf{g}) + \lambda\mathbf{g} = \mathbf{0} \tag{20}$$

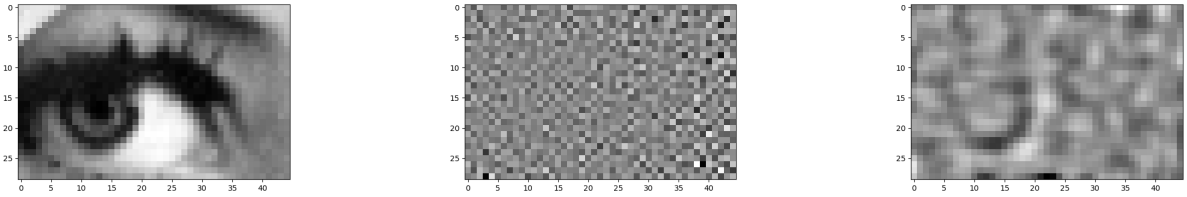
$$\Rightarrow (\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})\mathbf{g} = \mathbf{X}\mathbf{y} \tag{21}$$

$$\Rightarrow \mathbf{g} = (\mathbf{S} + \lambda\mathbf{I})^{-1}\mathbf{X}\mathbf{y} \tag{22}$$

Q4.3

ATTENTION: According to https://stackoverflow.com/questions/3810865/matplotlib-unknown-projection-3d-error, the provided code is actually not compatible `matplotlib` of version higher than 0.99, which is a quite old version. To make the code compatible with newer version, I have made additional `import` and changed some of the provided code. Please see comments in the script for detail and make necessary changes to run the program.

As we can see from the visualization in Figure. 5, the result $\mathbf{g}$ from $\lambda = 0$ is noisier than that from $\lambda = 1$.



(a) Template (b) $\mathbf{g}$ when $\lambda = 0$ (c) $\mathbf{g}$ when $\lambda = 1$

Figure 5: Visualization of the Ground Truth Template and the Linear Discriminant for Different λ's

The results from correlation are shown in Figure. 6, from which we can see that the filter from $\lambda = 1$ gives a better result as there is a clear white pixel in the left eye region in Figure. 6(b).

After examining the response values of these two version, we find that actually in the template region, the response are almost the same, but for $\lambda = 0$, the correlation response has high values (larger than 1) outside the template region, which makes the maximum of the whole response not at the desired position (center of the template).

Considering that we are to a learn a filter that can discriminate the template from its shifted version, the regularization term $||\mathbf{g}||_2^2$ here probably provides better ability to generalize such discriminating skill outside the template region. And this is also called preventing overfitting using regularization. I believe this is the reason why the correlation filter works better when $\lambda = 1$.

(a) $\lambda = 0$ (b) $\lambda = 1$

Figure 6: Correlation Results with Different λ.

Q4.4

Comparing Figure. 6(b) and Figure. 7(a), we can see the difference between the correlation result and convolution result. This is because the convolution can be regarded as correlation with a flipped filter:

$$(g * x)[i, j] = \sum_m \sum_n g[m, n]x[i - m, j - n] \tag{23}$$

Therefore, we can just flip the filter (in both height and width dimensions) to make convolution yield the result as that from correlation, as shown in Figure. 7(b).

(a) convolution with $\mathbf{g}$

(b) convolution with flipped $\mathbf{g}$

Figure 7: Convolution Results with $\mathbf{g}$ and Flipped $\mathbf{g}$ while $\lambda = 1$.