16-820 Advanced Computer Vision: Homework 2 (Fall 2024)
Lucas-Kanade Tracking

Ryan Wu (Andrew ID: weihuanw)

Wednesday October 2, 2024

Q1.1

1. What is $\frac{\partial W(x;p)}{\partial p^T}$?

$\frac{\partial W(x;p)}{\partial p^T}$ is the Jacobian of the warp function with respect to the parameter p.
The pure translation warp function is given by:

$$W(x;p) = x + p = \begin{bmatrix} 1 & 0 & p_x \\ 0 & 1 & p_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + p_x \\ y + p_y \end{bmatrix}$$

We can derive $\frac{\partial W(x;p)}{\partial p^T}$ as follows:

$$\frac{\partial W(x;p)}{\partial p^T} = \begin{bmatrix} \frac{\partial(x+p_x)}{\partial p_x} & \frac{\partial(x+p_x)}{\partial p_y} \\ \frac{\partial(y+p_y)}{\partial p_x} & \frac{\partial(y+p_y)}{\partial p_y} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

2. What is A and b?

A is the image gradients at each pixel, which can be solved using the steepest descent method. It can be represented as follows:

$$A = \frac{\partial I_{t+1}(x')}{\partial x'^T} \quad \frac{\partial W(x;p)}{\partial p^T}$$

where (pure translation)

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

$$A = \begin{bmatrix} \frac{\partial I_{t+1}(W(x_1;p))}{\partial x} & \frac{\partial I_{t+1}(W(x_1;p))}{\partial y} \\ \frac{\partial I_{t+1}(W(x_2;p))}{\partial x} & \frac{\partial I_{t+1}(W(x_2;p))}{\partial y} \\ \vdots & \vdots \\ \frac{\partial I_{t+1}(W(x_D;p))}{\partial x} & \frac{\partial I_{t+1}(W(x_D;p))}{\partial y} \end{bmatrix}$$

b is the pixel intensity differences between the template and the warped image, which is the error we want to minimize. It can be represented as follows:

$$b = \begin{bmatrix} I_t(x_1) - I_{t+1}(W(x_1;p)) \\ I_t(x_2) - I_{t+1}(W(x_2;p)) \\ \vdots \\ I_t(x_D) - I_{t+1}(W(x_D;p)) \end{bmatrix}$$

3. What conditions must A^TA meet so that a unique solution to Δ_p can be found?
A^TA must be full ranked/invertible, non-singular, and positive definite so that a unique solution to Δ_p can be found.

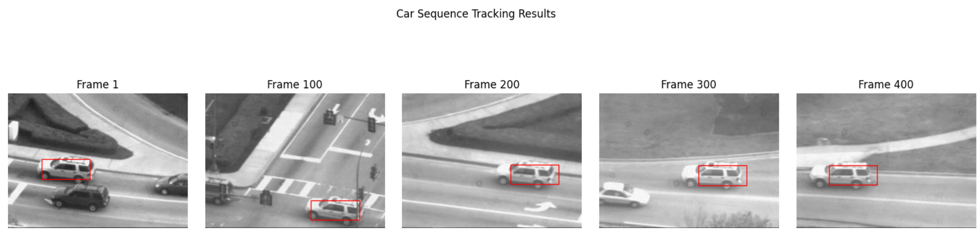
Q1.2

Implement a function with the following signature: LucasKanade(It, It1, rect, p0 = np.zeros(2))

Function implemented, results shown in the following questions.

Q1.3
Report your tracking performance (image + bounding rectangle) for the car sequence at frames 1, 100, 200, 300 and 400.

With default parameters, my tracking performance for the car sequence is shown below:



The output visualization of my Lucas Kanade tracking implementation on the car sequence.

Report your tracking performance (image + bounding rectangle) for the girl sequence at frames 1, 20, 40, 60 and 8.

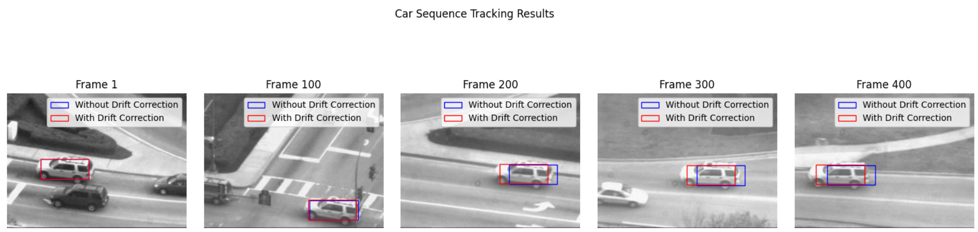
With default parameters, my tracking performance for the girl sequence is shown below:

The output visualization of my Lucas Kanade tracking implementation on the girl sequence.

Q1.4

Report your tracking performance with template correction (image + bounding rectangle) for the car sequence at frames 1, 100, 200, 300 and 400.

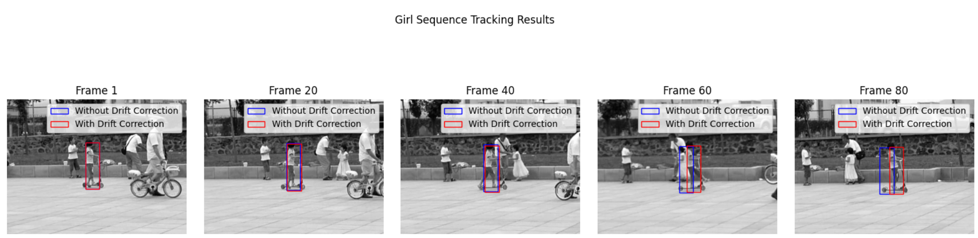
With default parameters, my tracking performance with template correction for the car sequence is shown below:



The output visualization of my Lucas Kanade tracking with template correction implementation on the car sequence.

Report your tracking performance with template correction (image + bounding rectangle) for the girl sequence at frames 1, 20, 40, 60 and 8.

With default parameters, my tracking performance with template correction for the girl sequence is shown below:



The output visualization of my Lucas Kanade tracking with template correction implementation the on girl sequence.

Q2.1
Write a function with the following signature: LucasKanadeAffine(It, It1)

Function implemented, results shown in the following questions.

Q2.2
Write a function with the following signature: SubtractDominantMotion(image1, image2)

Function implemented, results shown in the following questions.

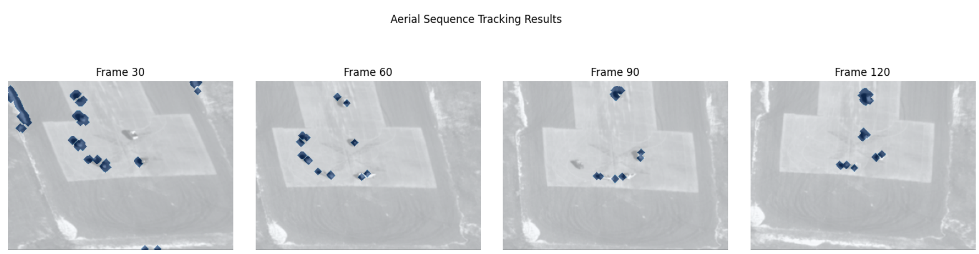
Q2.3

Report the motion detection performance at frames 30, 60, 90 and 120 with the corresponding binary masks superimposed for ant and aerial sequences.

With Lucas-Kanade iterations $= 1 \times 10^{10}$; dp threshold for Lucas-Kanade termination $= 5$; the binary threshold for mask intensity difference $= 0.016$; 1 erosion iteration; 5 dilation iterations, my motion detection performance for the ant sequence is shown below:

The output visualization of my Lucas Kanade motion tracking implementation on the ant sequence.

With Lucas-Kanade iterations $= 1 \times 10^{3}$; dp threshold for Lucas-Kanade termination $= 5$; the binary threshold for mask intensity difference $= 0.165$; 1 erosion iteration; 5 dilation iterations, my motion detection performance for the aerial sequence is shown below:



The output visualization of my Lucas Kanade motion tracking implementation on the aerial sequence.

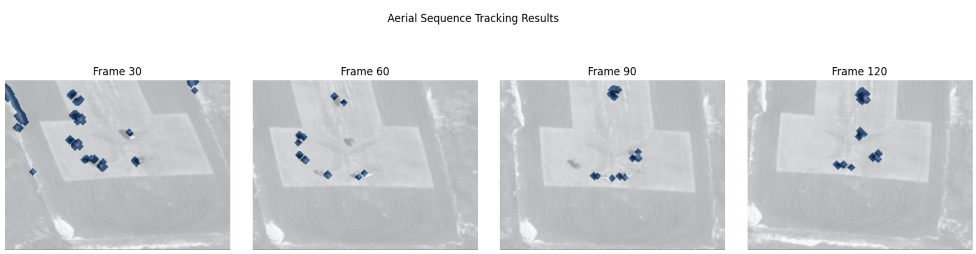
Q3.1

Reimplement the function LucasKanadeAffine(It,It1) as InverseCompositionAffine(It,It1) using the inverse compositional method. Similar to Q2.3, please visualize motion detection either from antseq.npy or from aerialseq.npy.

Using the inverse composition method With Lucas-Kanade iterations $= 1 \times 10^{10}$; dp threshold for Lucas-Kanade termination $= 5$; the binary threshold for mask intensity difference $= 0.016$; 1 erosion iteration; 5 dilation iterations, my motion detection performance for the ant sequence is shown below:

The output visualization of my Lucas Kanade motion tracking with inverse composition implementation on the ant sequence.

Using the inverse composition method with Lucas-Kanade iterations $= 1 \times 10^{3}$; dp threshold for Lucas-Kanade termination $= 5$; the binary threshold for mask intensity difference $= 0.165$; 1 erosion iteration; 5 dilation iterations, my motion detection performance for the aerial sequence is shown below:



The output visualization of my Lucas Kanade motion tracking with inverse composition implementation on the aerial sequence.

Q3.2

In your own words please describe why the inverse compositional approach is more computationally efficient than the classical approach.

The inverse compositional approach is more computationally efficient because it pre-computes the template image's gradient and the Jacobian matrix. By refactoring these heavy calculations outside the loop, the algorithm runs faster.

References

[1] L. Matthews, T. Ishikawa and S. Baker, "The template update problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 810-815, June 2004. https://doi.org/10.1109/TPAMI.2004.16

[2] Baker, S. (n.d.). Lucas-Kanade 20 Years on: A unifying framework part 1. Retrieved from https://www.ri.cmu.edu/pub_files/pub3/baker_simon_2004_1/baker_simon_2004_1.pdf

[3] Wikimedia Foundation. (2024, May 15). Lucas–Kanade Method. *Wikipedia*. Retrieved from https://en.wikipedia.org/wiki/Lucas%E2%80%93Kanade_method