Submitted by: Shahram Najam Syed (snysed) **Homework # 3 (16-720)**
Dated: 24th October 2023

Q1.1:

- **What is $\frac{\partial W(x;p)}{\partial p^T}$? Write out the mathematical expression.**

Ans: The transformation $W(x;p)$ maps a point x in the source image (or in our case frame of a video at time "t") to a point x' in the target image (in our case the template image) based on the parameters p. In the explanation provided in the writeup, $W(x;p)$ is a simple translation transformation described as:

$$x' = W(x;p) = x + p$$

Where x is the original point in the source image (i.e. 2D vector $[x, y]^T$) and p is the translation vector (i.e. 2D vector $[p_x, p_y]^T$).

The expression $\frac{\partial W(x;p)}{\partial p^T}$ represents the Jacobian matrix of the transformation W with respect to the parameter vector p (translation vector) at a specific point x. By calculating the Jacobian matrix of a function, we determine the rate of change of the function's output with respect to its input, i.e. it describes how small changes in the parameters p affect the transformed point x'.

For the translation transformation $W(x;p) = x + p$, the Jacobian $\frac{\partial W(x;p)}{\partial p^T}$ is computed as:

$$\frac{\partial W(x;p)}{\partial p^T} = \frac{\partial\ (x+p)}{\partial p^T}$$

And,

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

So,

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

Computing each derivative,

$\frac{\partial\ (x+p_x)}{\partial p_x}$ = 1, which indicates that a unit change in p_x will result in a unit change in the x-coordinate of the warp $W(x;p)$, i.e. x-coordinate of the warp is dependent on p_x and independent of p_y.

$\frac{\partial\ (x+p_x)}{\partial p_y}$ = 0, which indicates no relationship or influence of changing p_y to the x-coordinate

$\frac{\partial\ (x+p_y)}{\partial p_x}$ = 0, which indicates no relationship or influence of changing p_x to the y-coordinate

$\frac{\partial\ (x+p_y)}{\partial p_y}$ = 1, which indicates that a unit change in p_y will result in a unit change in the y-coordinate of the warp, $W(x;p)$, i.e. y-coordinate is dependent on p_y and independent of p_x.

So,

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

In essence, the above identity matrix represents a one-to-one correspondence between the warp parameters p and the resulting change in position. Intuitively this makes sense since the warp function is a pure translation of positions without scaling, rotation, or other complex transformations.

- **What is A and b? Write out the mathematical expression?**

Ans: From the linearized approximation:

$$It + 1(x' + \Delta p) \approx It + 1(x') + \frac{\partial It + 1(x')}{\partial x'^T} \frac{\partial W(x;p)}{\partial p^T} \Delta p$$

The error term e, which is the difference between the template (original image) and the warped version of the next image becomes:

$$e = It + 1(x') + \frac{\partial It + 1(x')}{\partial x'^T} \frac{\partial W(x;p)}{\partial p^T} \Delta p - It(x)$$

Given the linear approximation, the goal is to minimize this error, and this is represented in a least-squares sense as:

$$\arg min_{\Delta p} \|A \, \Delta p - b\|_2^2$$

The goal is to find the parameter update Δp that minimizes the squared error between a linear transformation of Δp and a vector b.

From the above equation,

a. A (steepest Descent Images) is the Jacobian of the image intensities with respect to the warp parameters. Since, $\frac{\partial W(x;p)}{\partial p^T}$ is the identity matrix (as shown in the previous question), A simplifies to the spatial gradient of the image at each pixel multiplied by the warp Jacobian:

$$A = \sum_{x \in N} \frac{\partial It + 1(x')}{\partial x'^T}$$

This means that for each pixel in the neighborhood N, we are calculating the image gradient of the next frame $It + 1$, the summation of these gradients yields the steepest descent direction, indicating how the image intensities change with respect to small changes in position.

A is matrix where each row corresponds to the spatial gradient (x and y gradients) of a pixel in the neighborhood. If the neighborhood consists of D pixels, A will be of size D X 2.

$$A = \begin{bmatrix} \frac{\partial I_t(x_1')}{\partial x_1'} & \frac{\partial It(x_1')}{\partial x_2'} \\ \frac{\partial It(x_2')}{\partial x_1'} & \frac{\partial It(x_2')}{\partial x_2'} \\ . & . \\ . & . \\ . & . \\ . & . \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Where, $\frac{\partial It(x_i')}{\partial x_j'}$ is the image gradient of the template image I_t at the ith pixel coordinate with respect to the jth pixel coordinate.

b. On the other hand, vector b represents the difference between the intensities of the template in the current frame It and target images, i.e. next frame $It + 1$, under the current transformation evaluated at specific pixel coordinates. Mathematically, b can be expressed as:

$$b = It(x) - It + 1(x')$$

Which expands to:

$$b = \begin{bmatrix} I_{t+1}(x_1') - I_t(x_1') \\ I_t(x_2') - I_t(x_2') \\ . \\ . \\ . \\ . \\ . \end{bmatrix}$$

Where, $I_{t+1}(x_i')$ is the intensity of the target image at the transformed ith pixel coordinate, and $I_t(x_i')$ is the intensity of the template image at the same coordinate.

- **What conditions must $A^T A$ meet so that a unique solution to Δp can be found?**

Ans: To find a unique solution for Δp for the equation, $A^T A\ \Delta p = A^T b$, $A^T A$ must be non-singular i.e. invertible. And for $A^T A$ to be non-singular.

Q1.2:

Ans: Implemented as python code.

Q1.3:

Ans: Following is the result for Lucas Kanade without template correction on the car sequence (with default parameters num_iters=1x10^4 ; threshold=1x10^{-2}):

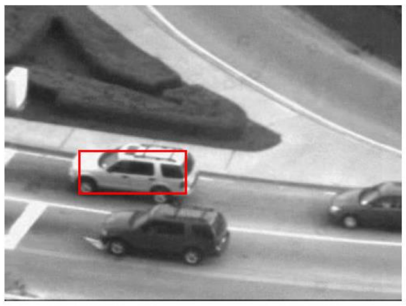
Figure: Frame 1

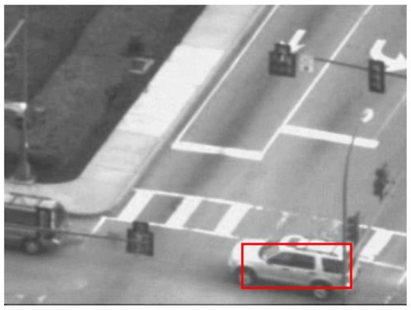
Figure: Frame 100

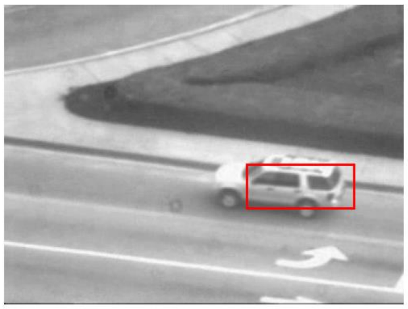
Figure: Frame 200

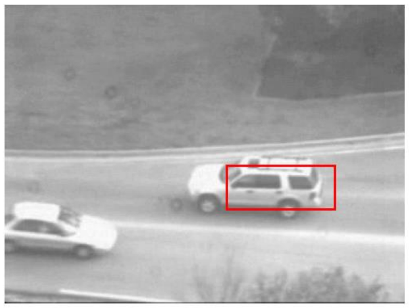
Figure: Frame 300

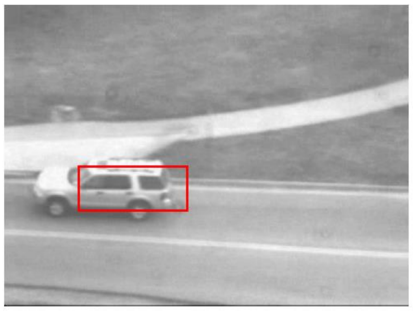
Figure: Frame 400

Following is the result for Lucas Kanade without template correction on the girl sequence (with default parameters num_iters=1x10^4 ; threshold=1x10^{-2}):

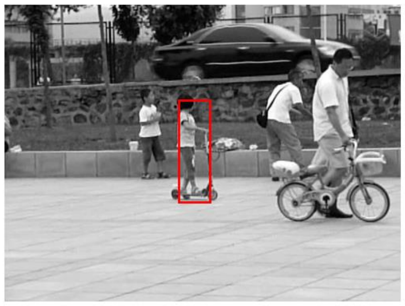
Figure: Frame 1

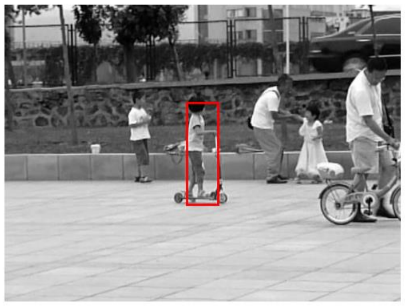
Figure: Frame 20

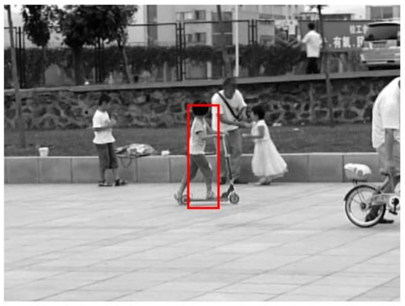
Figure: Frame 40

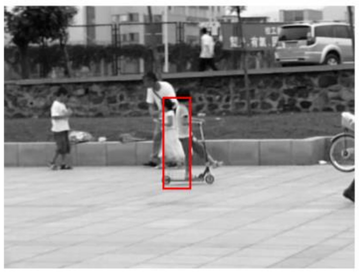
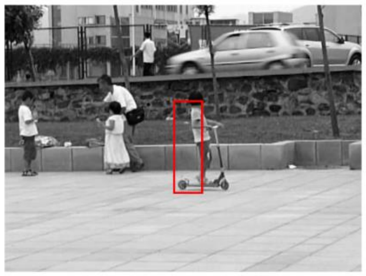

Figure: Frame 60 **Figure: Frame 80**

Over here, the tunable parameters available are:

a. num_iters: This is the number of iterations for the Lucas-Kanade algorithm used within the for the iterative optimization process. Increasing num_iters may lead to more refined and accurate results at the cost of computational time, while decreasing it would lead to run faster but at the expense of lesser accurate motion estimation (due to potential non-convergence).
b. threshold: This is the termination threshold for the Lucas-Kanade method. If the change between two consecutive iterations is less than this threshold, the algorithm will terminate prematurely considering it has converged. A good analogy to describe would be this being an "early stopping condition". A smaller threshold would mean that the algorithm would only stop when there is very little change between iterations, potentially leading to more accurate results but at the expense of runtime computational speed. Conversely, a larger threshold might speed up the process but could stop the optimization before it has truly converged, potentially leading to less accurate results.

Q1.4:

Ans: Following is the result for Lucas Kanade with template correction on the car sequence (with default parameters num_iters=1x10^4 ; threshold=1x10^{-2}):

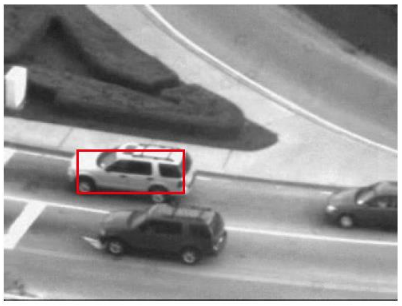
Figure: Frame 1*

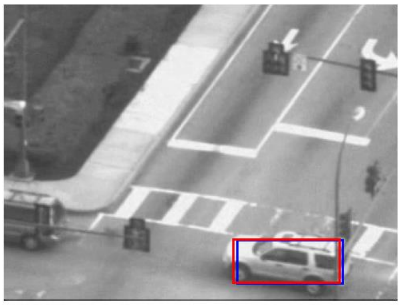
Figure: Frame 100

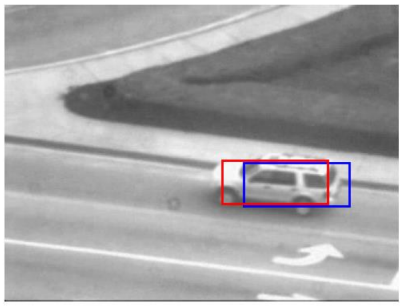
Figure: Frame 200

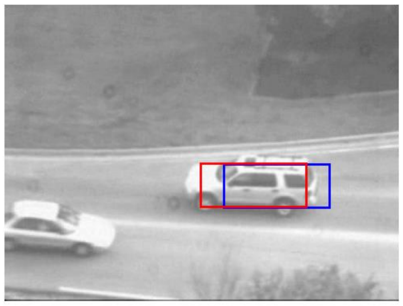
Figure: Frame 300

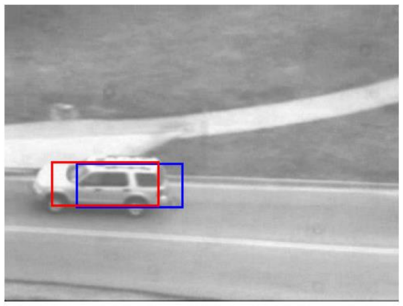
Figure: Frame 400

Following is the result for Lucas Kanade with template correction on the girl sequence (with default parameters num_iters=1x10^4 ; threshold=1x10^{-2}):

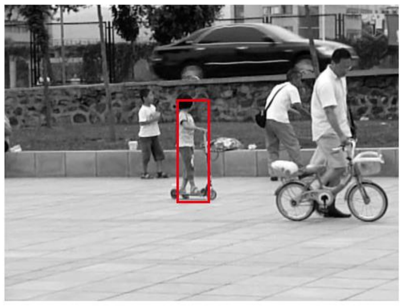
Figure: Frame 1

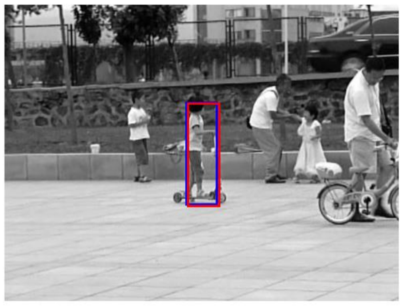
Figure: Frame 20

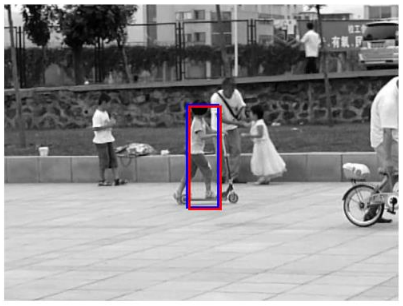
Figure: Frame 40

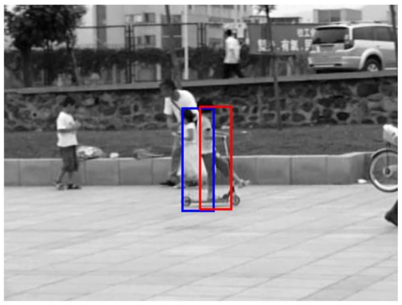
Figure: Frame 60

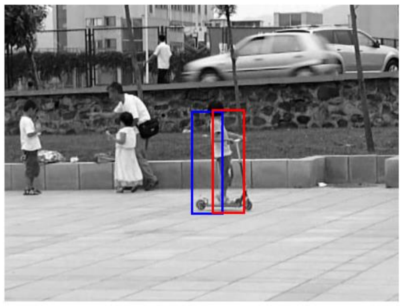
Figure: Frame 80

For this question, we introduced template correction and so in addition to the tunable parameters mentioned in the previous question (Q1.3), it has an additional parameter "template_threshold" which is used to determine when to update the template the tracking algorithm. Hence, a smaller template_threshold makes the algorithm more stricter about updating the template, meaning it would update less frequently and only when the change is very minor. Conversely, a larger template_threshold would make the algorithm more liberal in laxed in updating the template, potentially adapting to more significant changes in the appearance of the tracked object, but as a result might also become more prone to drifts.

Q2.1:

Ans: Implemented as python code.

Q2.2:

Ans: Implemented as python code.

Q2.3:

Ans: Following are the results for the Ant Sequence using the parameters (--num_iters= 5000, --theshold= 1e-4, --tolerance= 0.15):

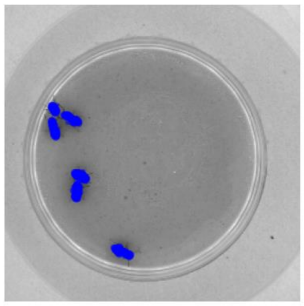
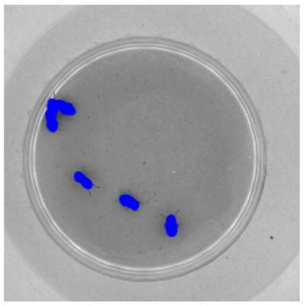
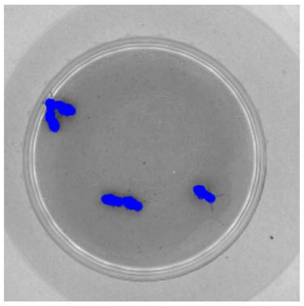
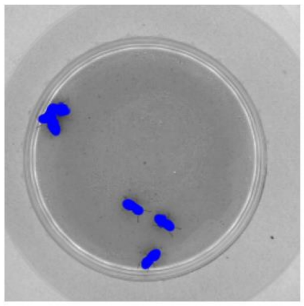

Figure: Frame 30 **Figure: Frame 60** **Figure: Frame 90** **Figure: Frame 120**

Following are the results for the Aerial Sequence using the parameters (--num_iters= 5000, --theshold= 1e-4, --tolerance= 0.15)

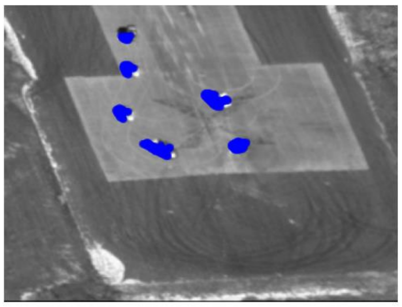
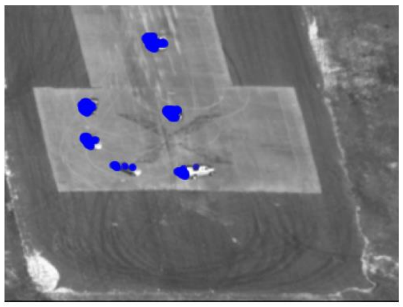
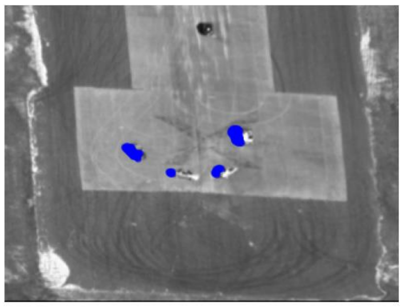

Figure: Frame 30 **Figure: Frame 60** **Figure: Frame 90**

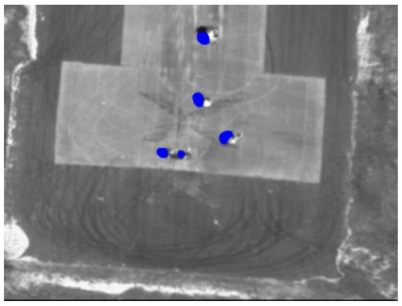

Figure: Frame 120

For this question, we introduced dominant motion subtraction and so in addition to the tunable parameters mentioned in the previous question (Q1.3), it has an additional parameter "tolerance" which is the binary threshold of intensity difference when computing the mask. In the context of motion subtraction, pixels in the difference image with intensity greater that this threshold are considered as belonging to moving objects, while those with lesser intensity are deemed static. A lower tolerance would classify more pixels as moving, potentially leading to noisier results where even small intensity differences (due to noise, slight illumination changes, etc.) might be flagged as motion.

Similarly, a higher tolerance would classify fewer pixels as moving potentially missing out on genuine subtle motions but providing "cleaner" results with fewer false positives.

Q3.1:

Ans: Inherently, "Inverse Compositional Tracking" is a variation of the standard Lucas-Kanade method for optical flow estimation, such that both methods seek to align a template image to a target image by estimating the warp between them, but inverse compositional method approaches this problem in a different manner, affording the advantages it has over vanilla Lucas-Kanade method, where instead of warping the template to align with the target, the goal is to find a warp that, when applied to the target aligns it to the template. The Jacobian and Hessian of the warp w.r.t. the template is precomputed and stored in memory, since the template remains fixed in the inverse compositional method, which is not the case of standard Lucas-Kanade Affine warping, where the Jacobian and Hessian need to be recalculated in each iteration as the warped template changes, thus allowing for a faster computational convergence and robustness. Following is the runtime speed (by runtime measurement) and memory usage (using memory-profiler) for both Lucas-Kanade Affine warp and inverse compositional tracking:

Algorithm Name	**Number of iterations**	**DP threshold**	**Binary threshold tolerance**	**Runtime speed**	**Memory usage**
Lucas-Kanade Affine Motion Subtraction	5000	10^{-4}	0.15	~51.5 seconds	~174 MiB
Inverse Compositional Tracking	5000	10^{-4}	0.15	~19.06 seconds	~171 MiB

Following are the results for the above mentioned parameters:

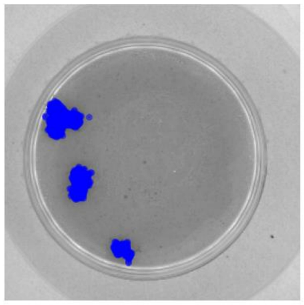
Figure: Frame 30

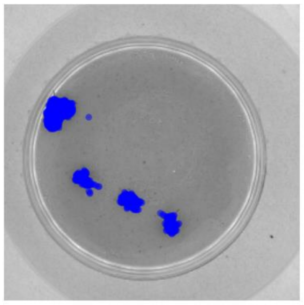
Figure: Frame 60

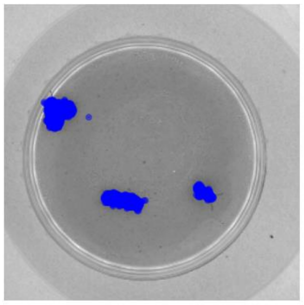
Figure: Frame 90

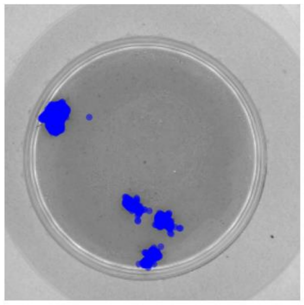
Figure: Frame 120

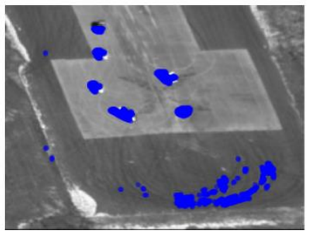
Figure: Frame 30

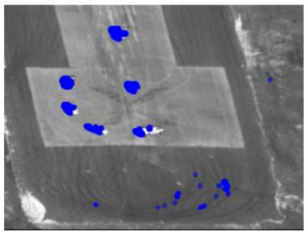
Figure: Frame 60

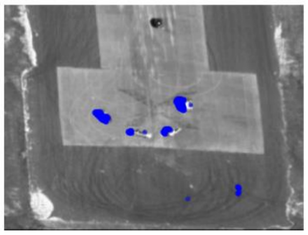
Figure: Frame 90

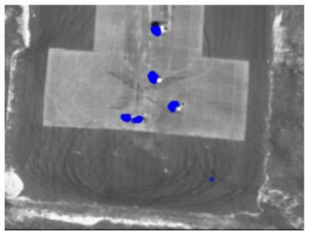
Figure: Frame 120

References:

[1] Hartley, Richard, and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[2] Szeliski, Richard. *Computer vision: algorithms and applications*. Springer Nature, 2022.

[3] Forsyth, David A., and Jean Ponce. *Computer vision: a modern approach*. prentice hall professional technical reference, 2002.

[4] Golub, Gene H., and Charles F. Van Loan. *Matrix computations*. JHU press, 2013.