Computer Vision - 16720-A

Kinjal Jain
Homework 3

March 7, 2020

1 Lucas-Kanade Tracking

***Q*1.1** What is $\frac{\partial W(x;p)}{\partial p^T}$?

$$x^{'} = W(x;p) = x + p$$

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

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

$$W(x;p) = \begin{bmatrix} W_x(x;p) \\ W_y(x;p) \end{bmatrix}$$

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

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

What is $\mathbf{A}$ and $\mathbf{b}$?

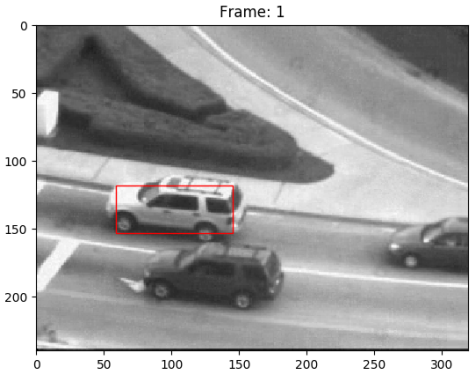
$$\mathbf{A} = \begin{bmatrix} \frac{\partial I_{t+1}\left(\mathbf{x}'_1\right)}{\partial \mathbf{x}'^T_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{\partial I_{t+1}\left(X'_N\right)}{\partial \mathbf{x}'_N} \end{bmatrix} \begin{bmatrix} \frac{\partial W(\mathbf{x}_1;\mathbf{p})}{\partial \mathbf{p}^T} \\ \vdots \\ \frac{\partial W(\mathbf{x}_N;\mathbf{P})}{\partial \mathbf{p}^T} \end{bmatrix}$$

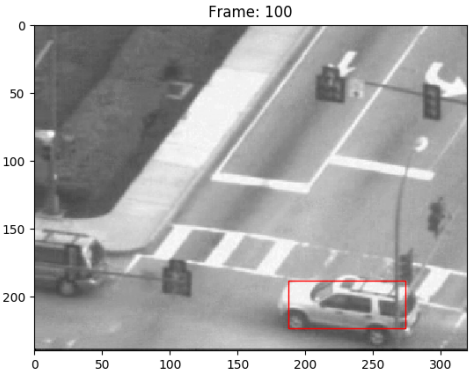
$$\mathbf{b} = \begin{bmatrix} I_t\left(\mathbf{x}_1\right) - I_{t+1}\left(\mathbf{x}'_1\right) \\ \vdots \\ I_t\left(\mathbf{x}_N\right) - I_{t+1}\left(\mathbf{x}'_N\right) \end{bmatrix}$$

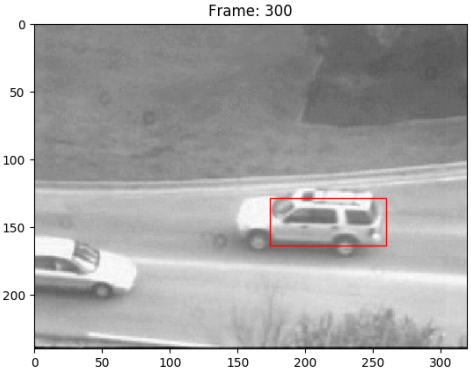
What conditions must $A^T A$ meet so that a unique solution to Δ p can be found?

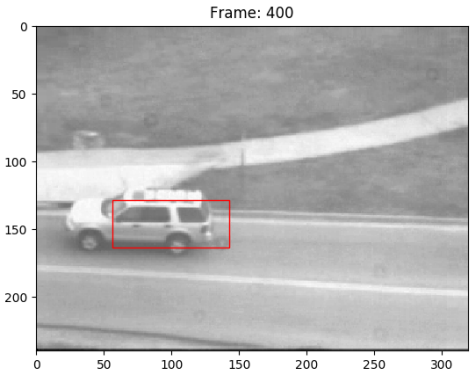
The required condition for $A^T A$ so that a unique solution to Δ p can be found is that the determinant of $A^T A$ should be non-zero.

***Q*1.3** Frames from *testCarSequence.py*

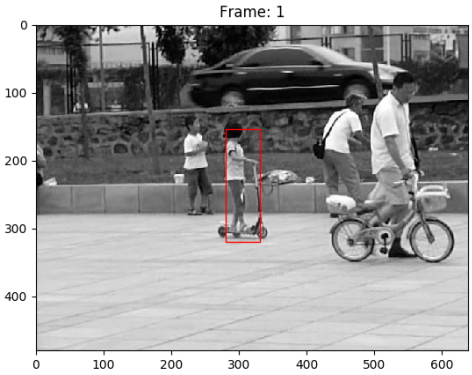


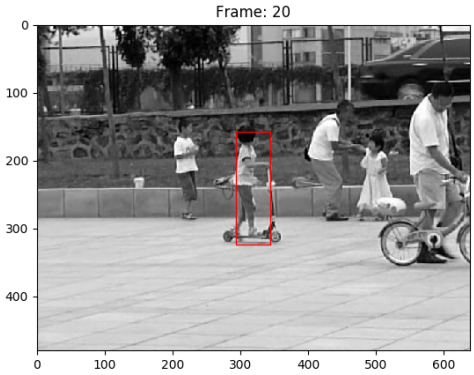


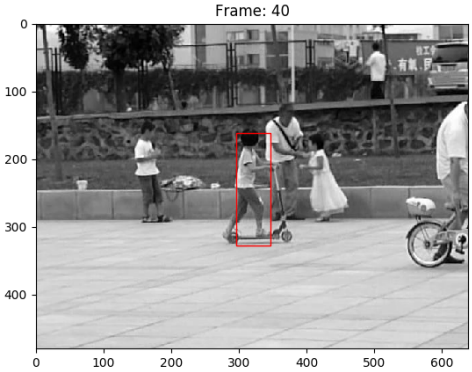


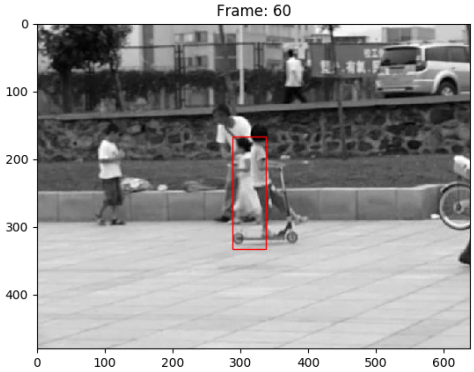


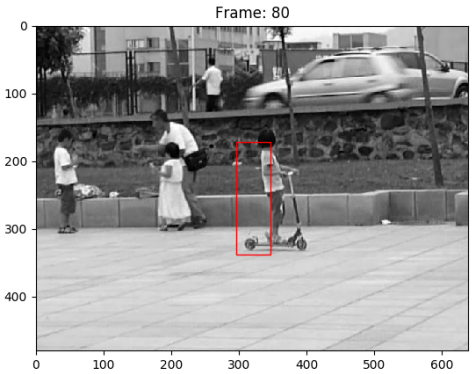
Frames from *testGirlSequence.py*



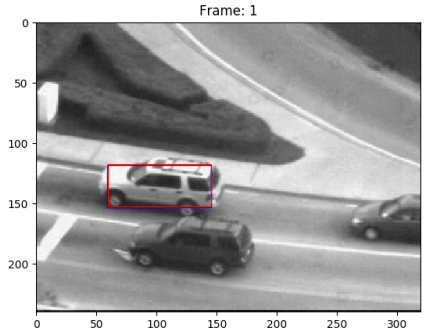


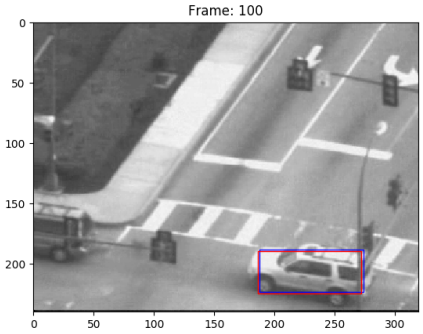


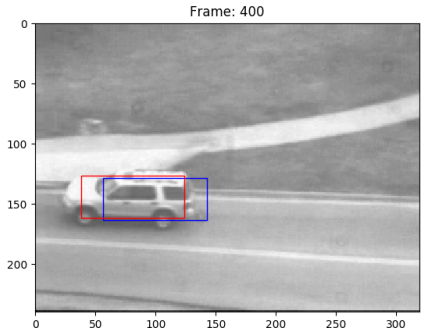




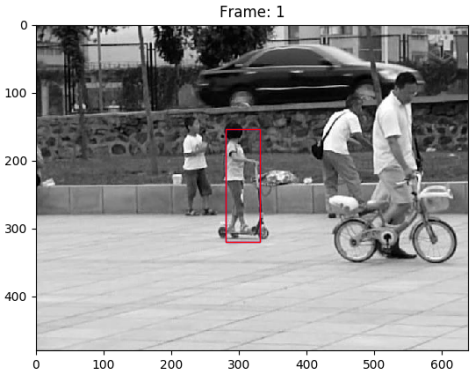
Q1.4 Red box depicts corrected bounded boxes and blue denotes the ones as generated for Q1.3
Frames from *testCarSequenceWithTemplateCorrection.py*

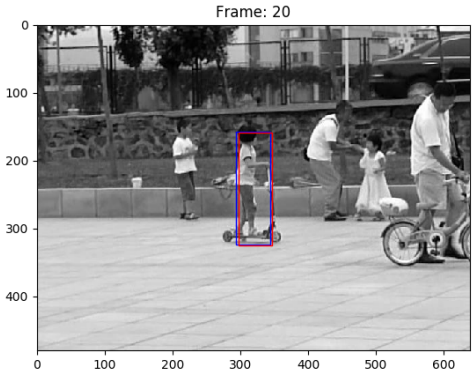


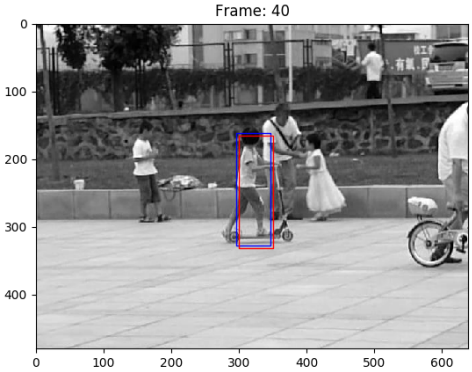


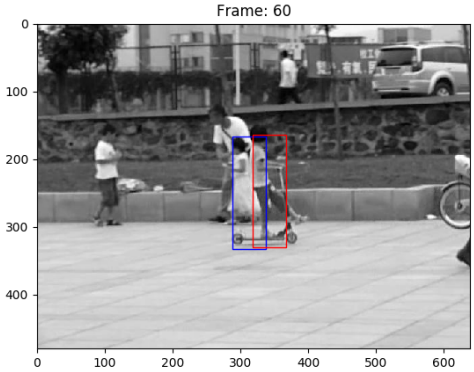


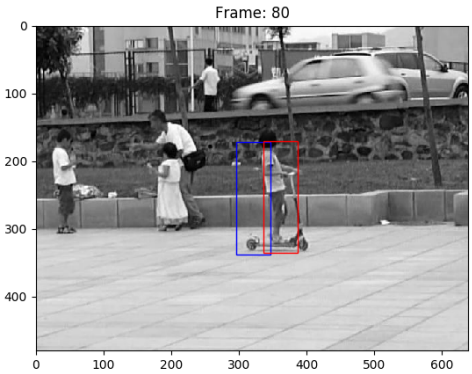
Frames from *testGirlSequenceWithTemplateCorrection.py*





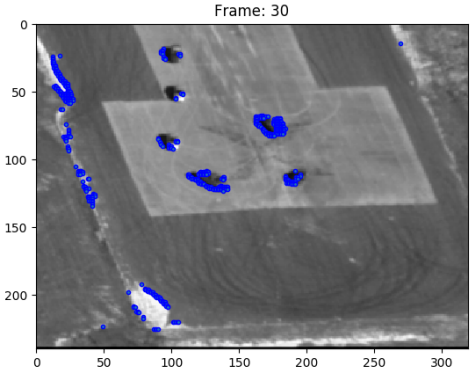


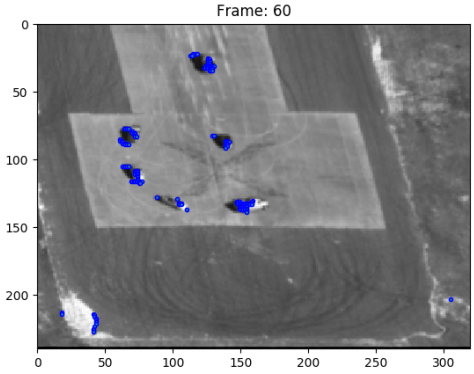


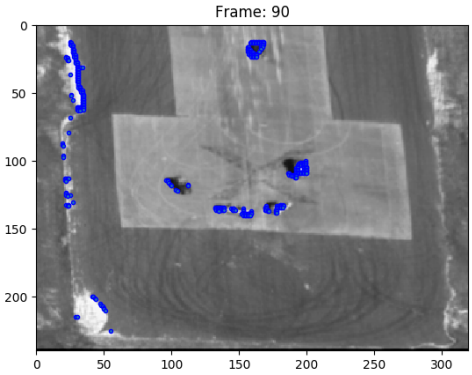


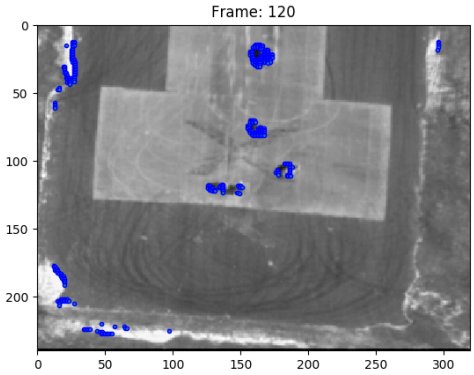
2 Affine Motion Subtraction

Q2.3 Frames for *testAerialSequence.py*:

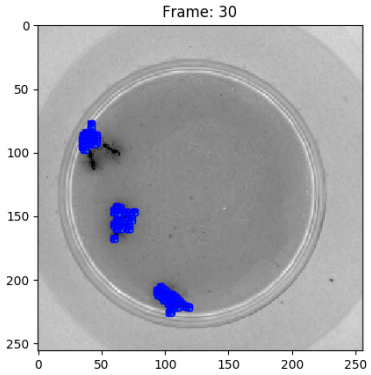


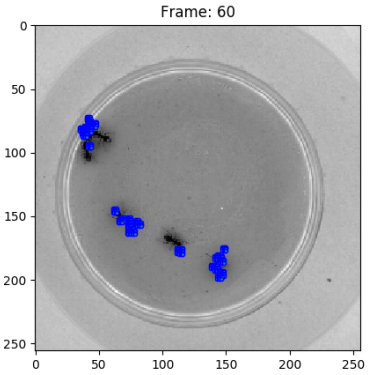


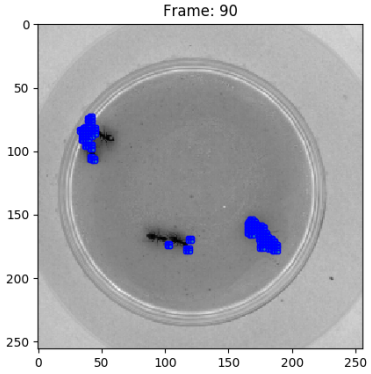


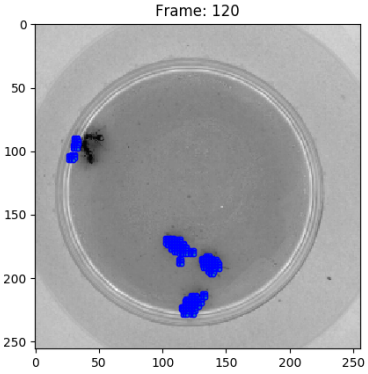


Frames for *testAntSequence.py*:





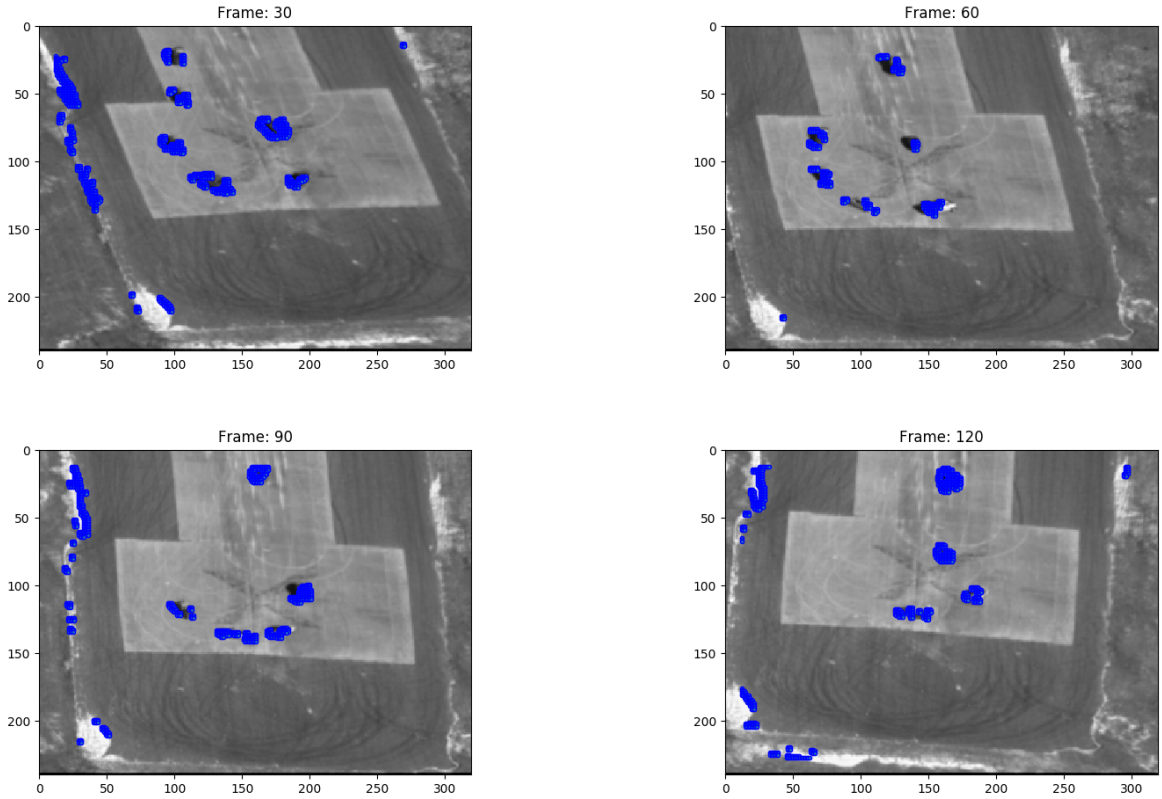




3 Efficient Tracking

***Q*3.1** Describe why the inverse compositional approach is more computationally efficient than the classical approach?
Inverse compositional approach is computationally more efficient than the classical approach because inverse composition method computes A and inverse of A only once as opposed to being calculated again in every iteration in classical approaches like Lukas Kanade Tracking. The only thing that is computed in the loop for the Inverse compositional algorithm is the error image.

Frames using inverse compositional approach for *testAerialSequence.py*:



Frames using inverse compositional approach for *testAntSequence.py*: