Q1.1

- What is $\frac{\partial \mathcal{W}(\mathbf{x};\mathbf{p})}{\partial \mathbf{p}^T}$?
  This is the Jacobian of the warp function. For the case of only translation, it can be written as:

$$\begin{bmatrix} \frac{d\mathcal{W}_x}{d\mathrm{p}_1} & \frac{d\mathcal{W}_x}{d\mathrm{p}_2} \\ \frac{d\mathcal{W}_y}{d\mathrm{p}_1} & \frac{d\mathcal{W}_y}{d\mathrm{p}_2} \end{bmatrix}$$

In this case, $\mathcal{W}(\mathbf{x};\mathbf{p})$ can be written as

$$\begin{bmatrix} \mathrm{x} \\ \mathrm{y} \end{bmatrix} + \begin{bmatrix} \mathrm{p}_1 \\ \mathrm{p}_2 \end{bmatrix}$$

So, we can evaluate the warp Jacobian as

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

- What is **A** and **b**?
  A is the matrix that represents the steepest descent images. A is calculated by multiplying the gradient of I_{t+1} by the warp Jacobian. It can be expressed as

$$\nabla I \, \frac{dW}{d\mathrm{p}}$$

  b is a vector representing the difference between all of the pixels in the template image and all the pixels in the current image. It can be expressed as

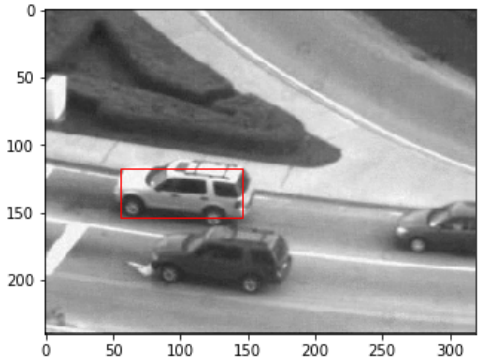
$$I_t(\mathbf{x}) - I_{t+1}(\mathbf{x})$$

- What conditions must $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ meet so that a unique solution to $\mathbf{\Delta p}$ can be found?
  The closed form solution of this least squares system is

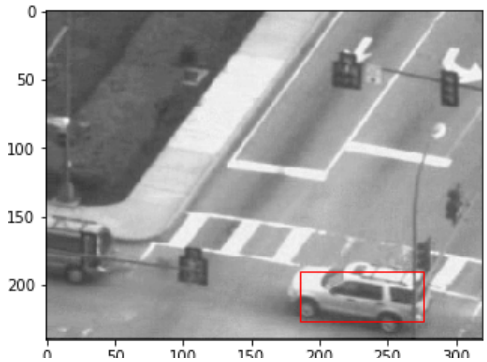
$$\Delta\mathbf{p} = (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b}$$

  This means that $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ must be invertible in order for a unique solution to be found.

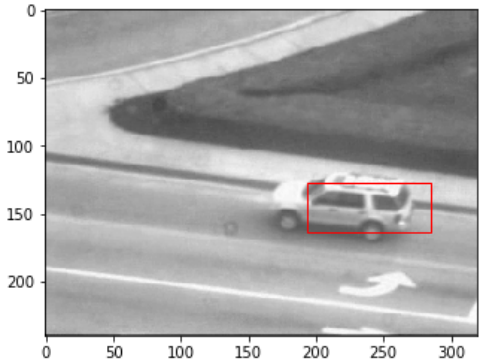
Q1.3 testCarSequence.py Tracking Results



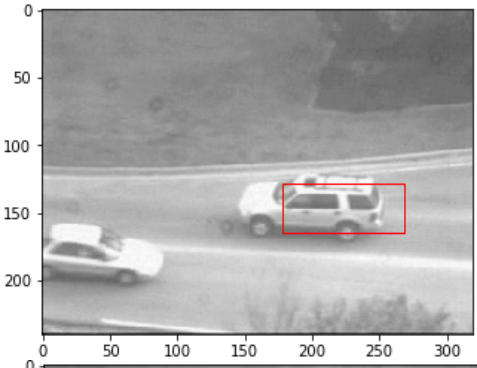
Frame 1



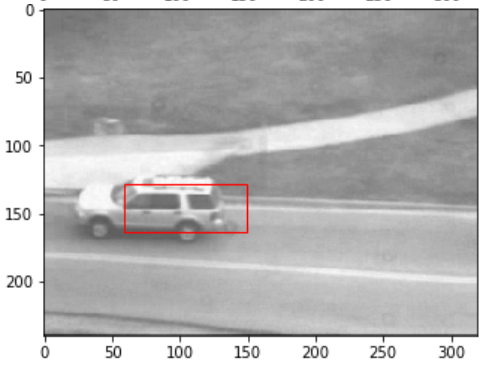
Frame 100



Frame 200

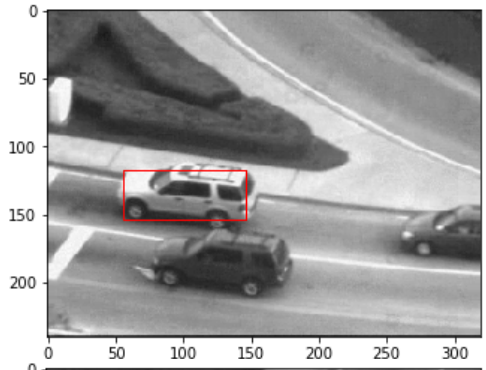


Frame 300

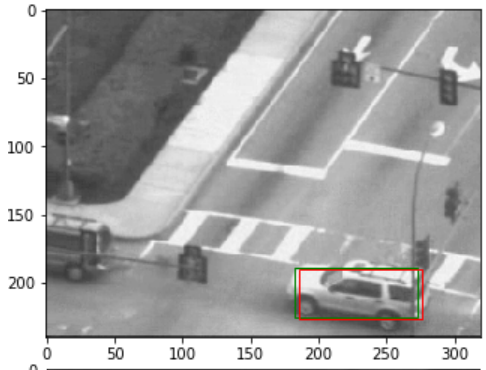


Frame 400

Q1.4 testCarSequenceWithTemplateCorrection.py Tracking Results (corrected results in green)



Frame 1



Frame 100

Frame 200

Frame 300

Frame 400

Q2.1

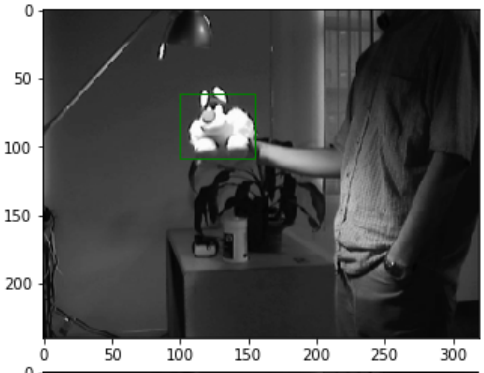
- Express **w** as a function of I_{t+1}, I_t, and B_k

  From the derivation in the Lucas Kanade 20 Years On paper, we see that the vector **w** can be expressed as

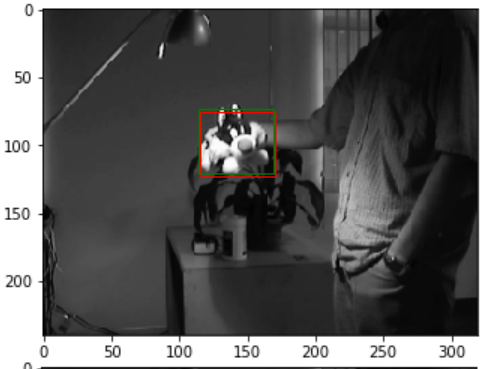
$$\mathbf{w} = \mathbf{B}(\mathbf{x}) * [I_{t+1}(\mathbf{x}) - I_t(\mathbf{x})]$$

  This assumes that each element of w corresponds a w_i, which is defined in equation (35) in the second LK paper.

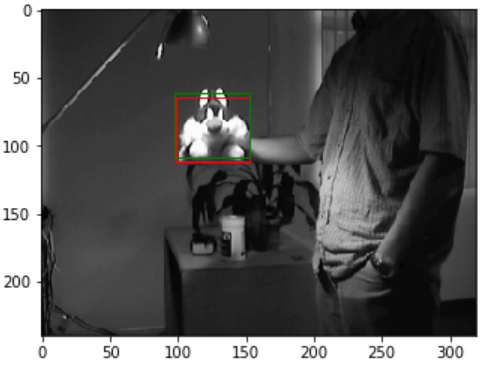
Q2.3 testSylvSequence.py Tracking Results (red is naïve LK and green is LK-Basis)



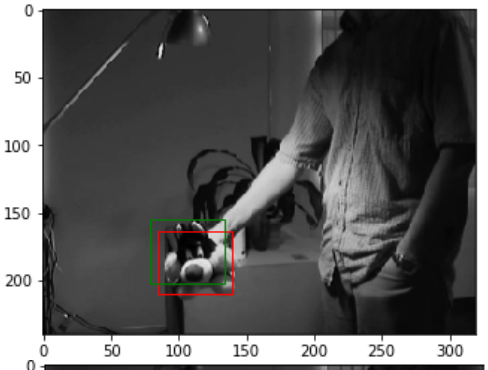
Frame 1



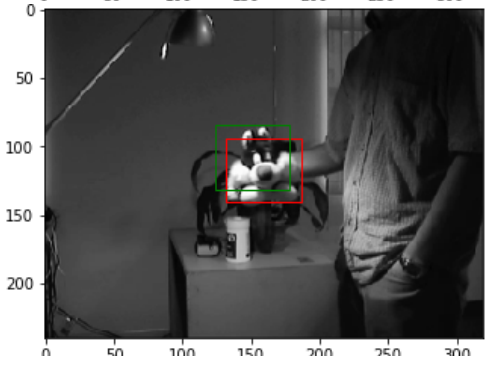
Frame 200



Frame 300

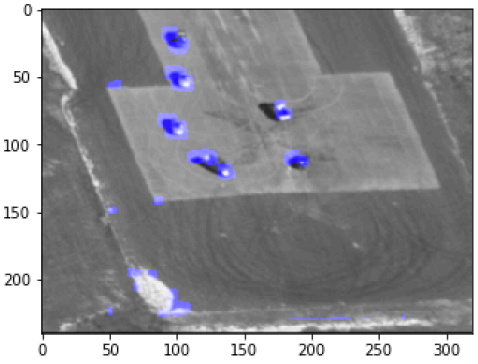


Frame 350

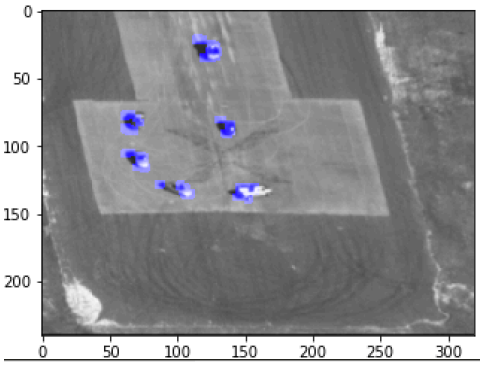


Frame 400

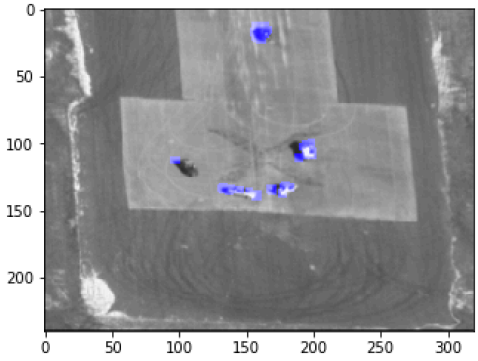
Q3.3 LucasKanadeAffine Results



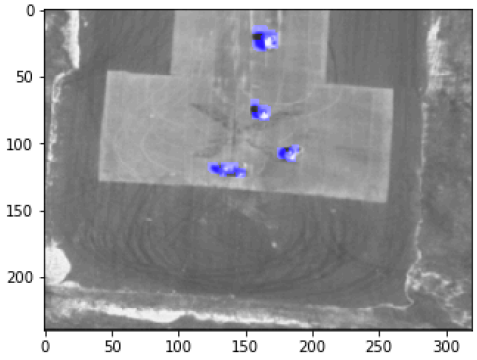
Frame 30



Frame 60



Frame 90



Frame 120

Q4.1 Benefits of Inverse Composition in Lucas Kanade

In inverse composition, we are able to pre-compute the steepest descent image matrix A and its transpose A^T. Using these two matrices we can subsequently pre-compute the hessian H and its inverse H^{-1}. This saves us from performing these costly matrix operations in every iteration of the loop, which eventually adds up to a lot of time and computing power saved.

Q4.2 Solution to Least Squares equation

$$\text{argmin}_g \frac{1}{2} \| \mathbf{y} - \mathrm{X}^{\mathrm{T}}\mathrm{g} \|_2^2 + \frac{\lambda}{2} \| \mathrm{g} \|_2^2$$

$$\text{argmin}_g \frac{1}{2}(\mathbf{y} - \mathbf{X}^{\mathrm{T}}\mathbf{g})^{\mathrm{T}}(\mathbf{y} - \mathbf{X}^{\mathrm{T}}\mathbf{g}) + \frac{\lambda}{2}\mathbf{g}^{\mathrm{T}}\mathbf{g}$$

$$\text{argmin}_g \frac{1}{2}(\mathbf{y}^{\mathrm{T}}\mathbf{y} - 2\mathbf{g}^{\mathrm{T}}\mathbf{X}\mathbf{y} - \mathbf{g}^{\mathrm{T}}\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{g}) + \frac{\lambda}{2}\mathbf{g}^{T}\mathbf{g}$$

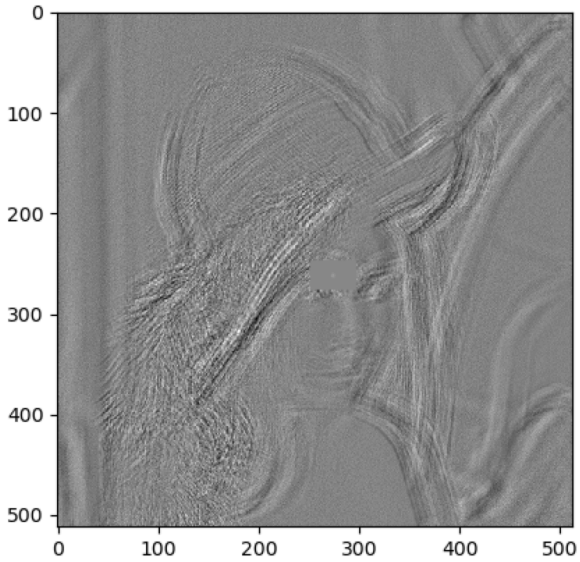
$$\frac{\partial}{\partial \boldsymbol{g}} = \mathbf{0} = -\mathbf{X}\mathbf{y} - \mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{g} + \lambda\mathbf{I}\mathbf{g}$$

$$\mathbf{g} = (\mathbf{X}\mathbf{X}^{\mathrm{T}} - \lambda\mathbf{I})^{-1}\mathbf{X}\mathbf{y}$$

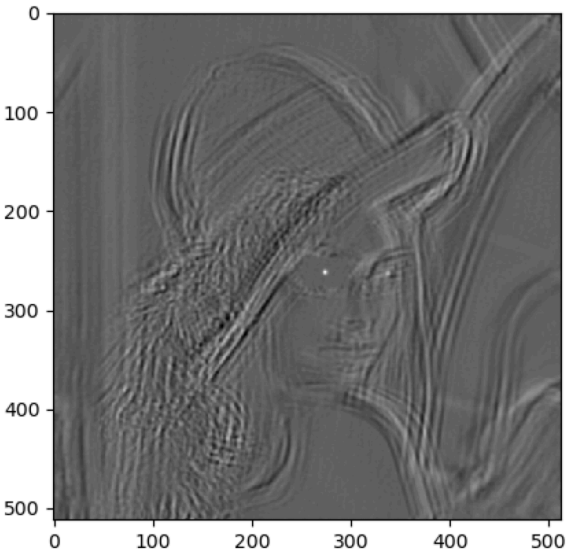
$$\mathbf{S} = \mathbf{X}\mathbf{X}^{\mathrm{T}}$$

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

Q4.3 Correlation of g on lena.npy



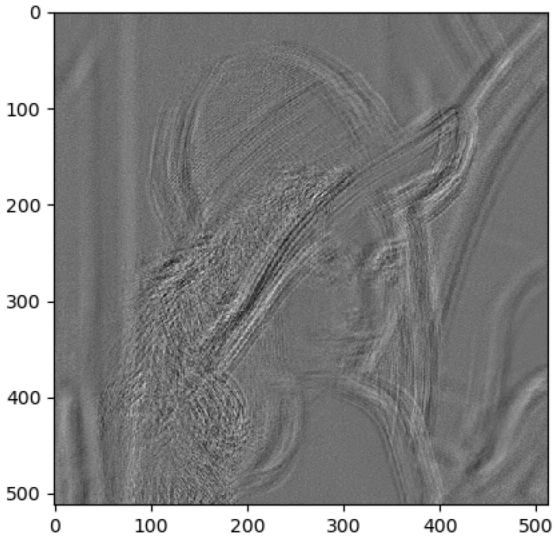
λ = 0



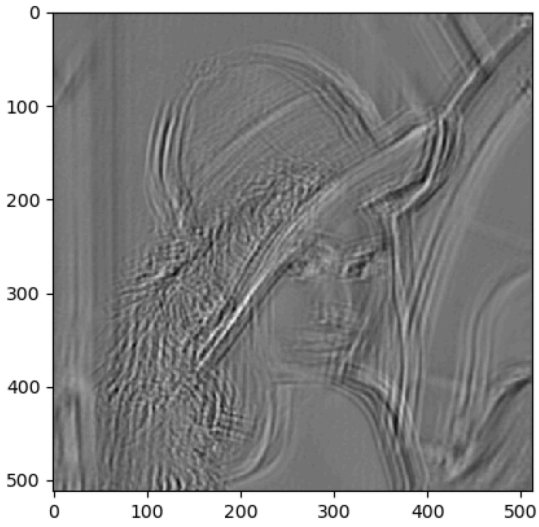
λ = 1

The filter for which λ = 1 performs best, as it has a very high (white) response at Lena's eyes, which makes sense because all of the subimages used in X were sampled around Lena's right eye. When λ = 0, we see a lot of noise near Lena's eyes as the result of overfitting. By adding the extra term in the minimization, we ensure that the g is forced to be smaller, which will prevent overfitting.

Q4.4 Convolution of g on lena.npy



λ = 0



λ = 1

We get a different response from correlation here because convolution performs correlation using a flipped version of the kernel. In order to get the same response as in 4.3, we would have to use np.flip(g,axis=0) and np.flig(g,axis=1) to flip the kernel along both axes. If we used that flipped g to perform convolution, we would get the same results as in 4.3.