Assignment 3

Problem 1

Q1.1

The term $\frac{\partial W(x;p)}{\partial p^T}$ is given as derivative of the warp of x using p wrt p. For a translational warp:

$$W(x;p) = \begin{bmatrix} 1 & 0 & p_1 \\ 0 & 1 & p_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Therefore, on differentiating:

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

A and b matrices are defined as follows:

$$A = \frac{\partial I(x')}{\partial x'^T} \frac{\partial W(x;p)}{\partial p^T}$$

where the second term is defined above and the first term is computed by taking the derivative of the image and then warping it to the template's coordinates.

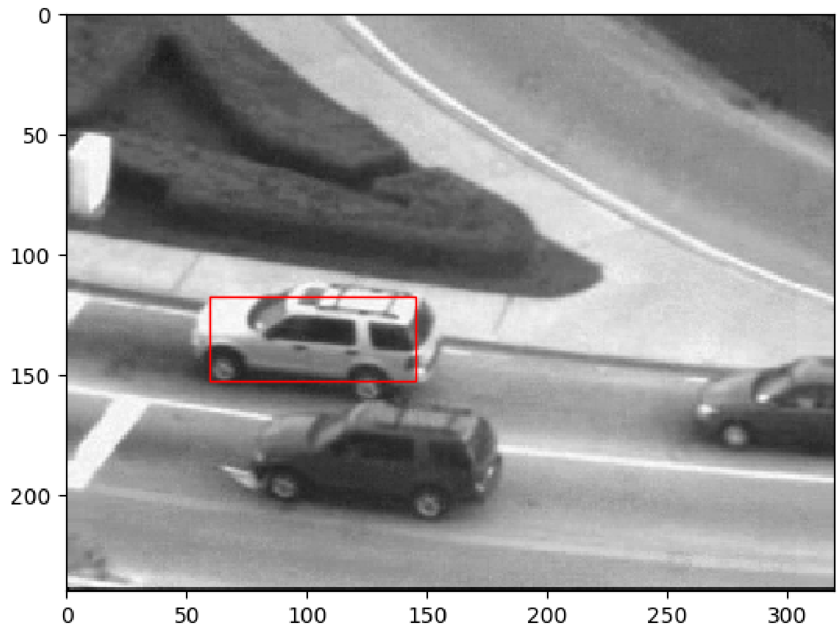
$$b = T(x) - I(W(x;p))$$

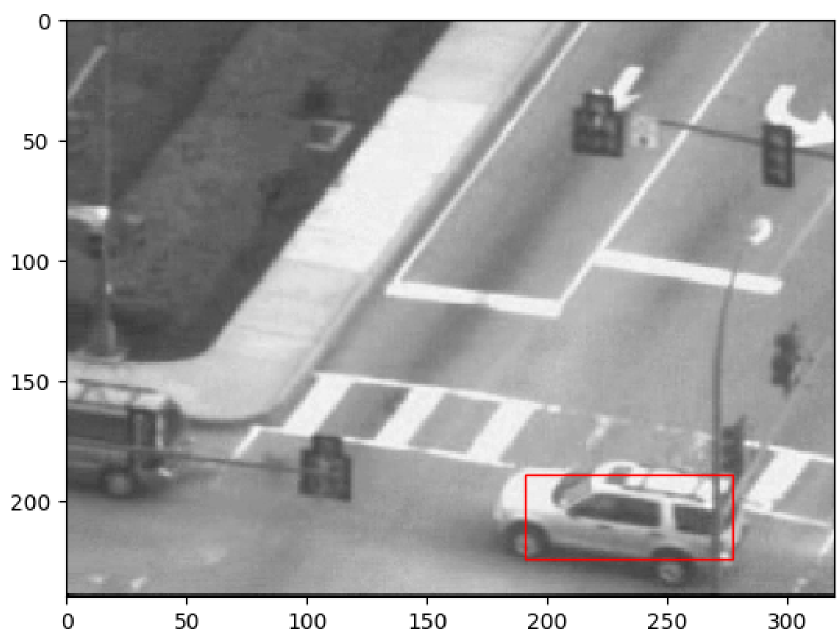
where T denotes the template and I the image.

Since the solution to the problem of solving for Δp is a least squares solution with A and b, $A^T A$ should be invertible.

Q1.3

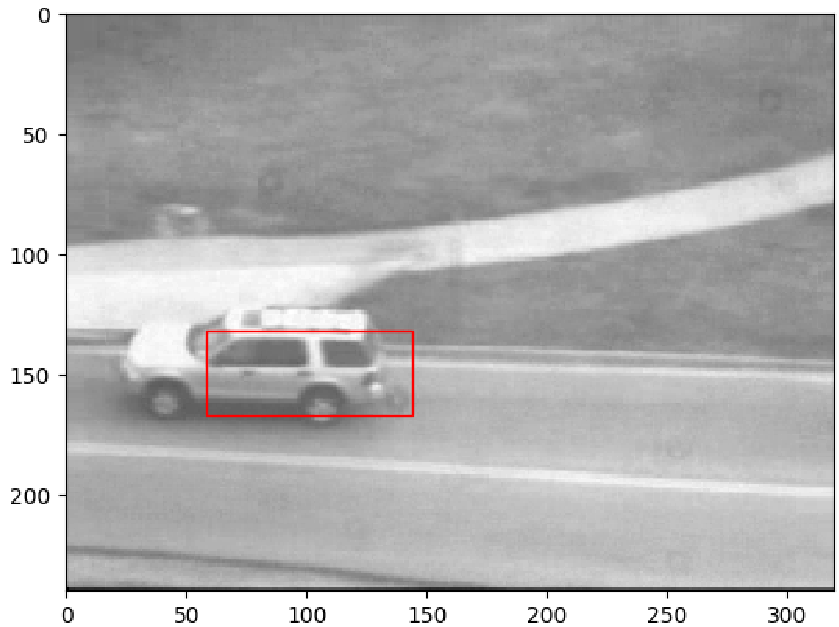
The following are results from the regular Lukas-Kanade algorithm at 1st, 100th, 200th, 300th and 400th frame. Running the script will run an animation and the part which generates these images has been commented out.


0
50
100
150
200
0
50
100
150
200
250
300


0
50
100
150
200
0
50
100
150
200
250
300

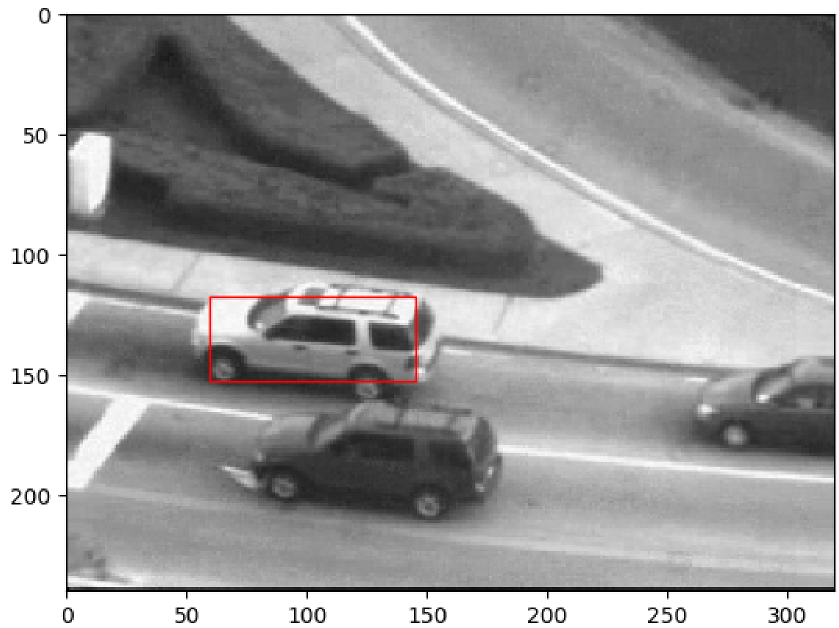

0
50
100
150
200
0
50
100
150
200
250
300

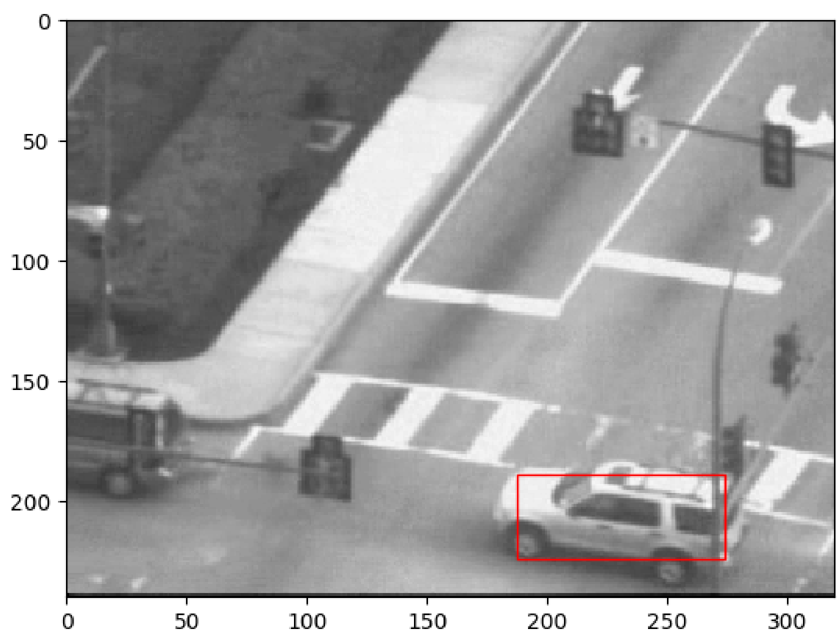

0
50
100
150
200
0
50
100
150
200
250
300



Q1.4

The following are results from the Lukas-Kanade algorithm with template correction scheme described in the referenced paper at 1st, 100th, 200th, 300th and 400th frame. Running the script will run an animation and the part which generates these images has been commented out.


0
50
100
150
200
0
50
100
150
200
250
300


0
50
100
150
200
0
50
100
150
200
250
300


0
50
100
150
200
0
50
100
150
200
250
300


0
50
100
150
200
0
50
100
150
200
250
300

Problem 2

Q2.1

$$\sum_{i=1}^{K} w_i B_i(x) = I_{t+1}(x) - I_t(x)$$

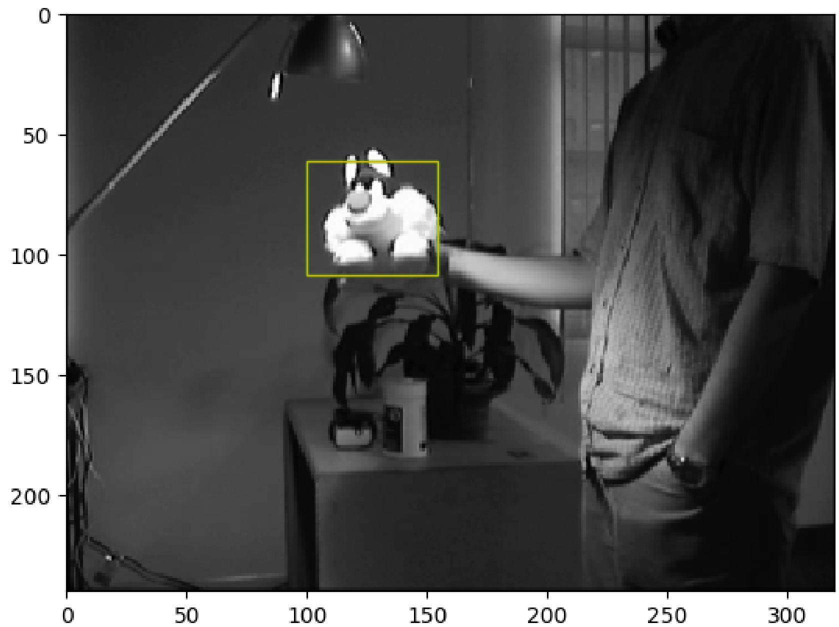
$$(\sum_{i=1}^{K} w_i B_i(x)).B_i(x) = (I_{t+1}(x) - I_t(x)).B_i(x)$$

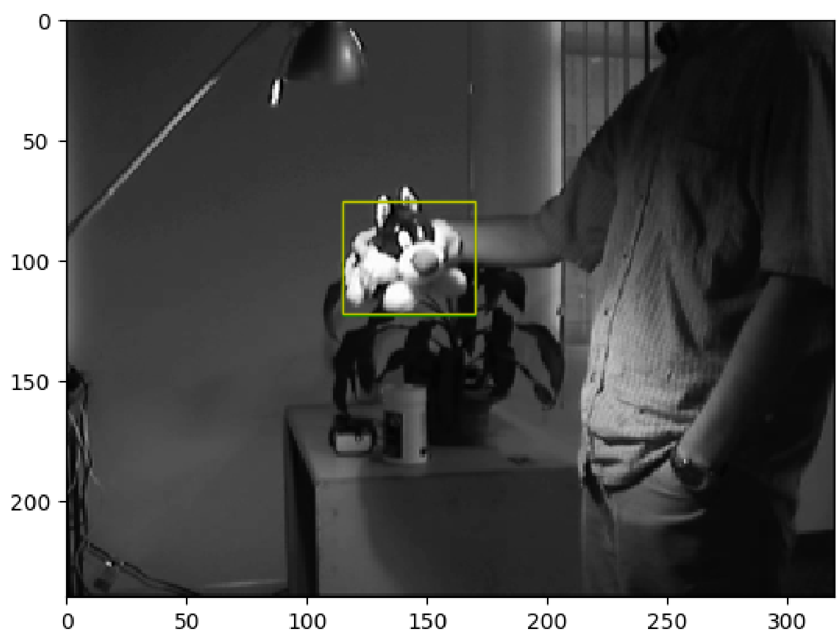
$$w_i = (I_{t+1}(x) - I_t(x)).B_i(x)$$

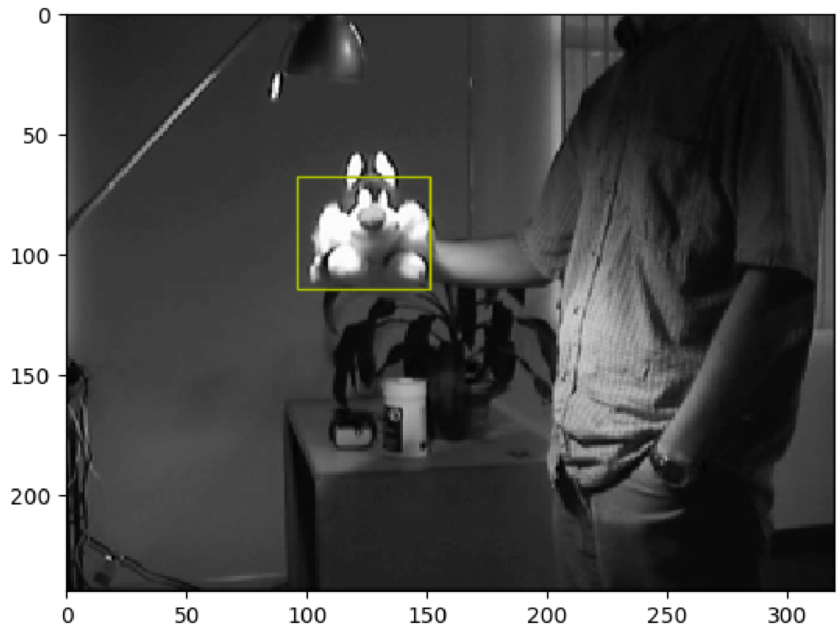
The above is written in matrix form. In reality, the w_i is actually a scalar written as the sum over all x when the dot product is expanded.

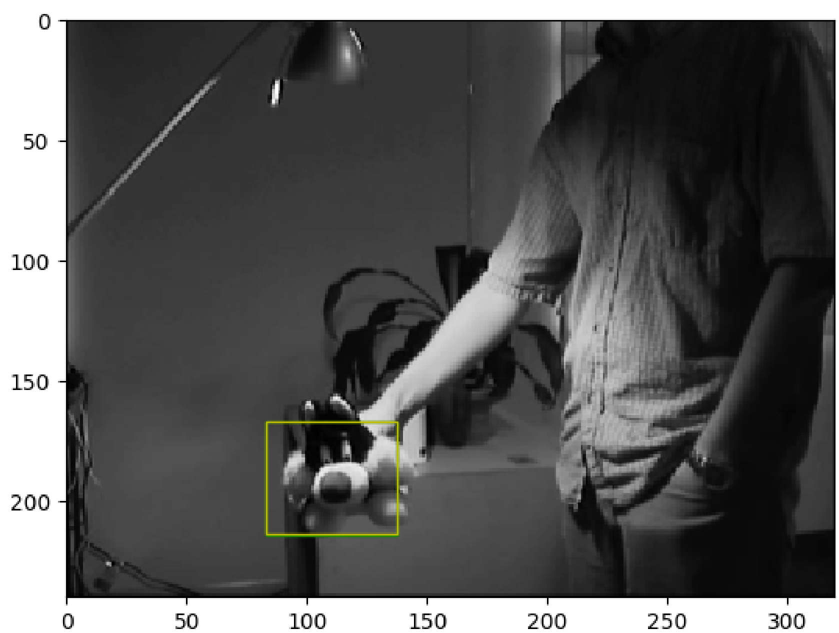
Q2.3

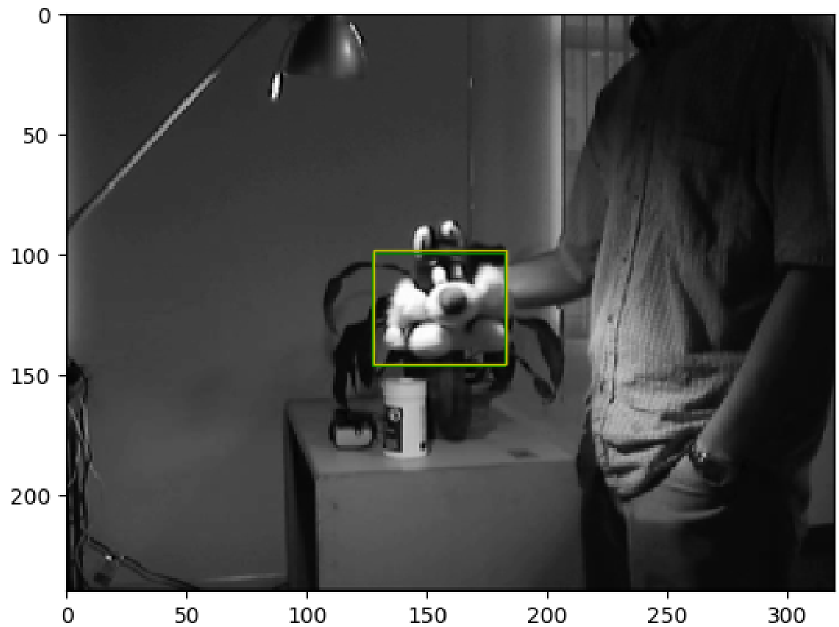
The following are results from the Lukas-Kanade algorithm with appearance basis (yellow bbox) and the regular Lucas-Kanade (green bbox) at 1st, 200th, 300th, 350th and 400th frame. Running the script will run an animation and the part which generates these images has been commented out. Please note that the regular LK algorithm performs equally well and in most of the frames, the two bboxes are not easily distinguishable (400th frame has both visible to some extent in the images below).


0
50
100
150
200
0
50
100
150
200
250
300


0
50
100
150
200
0
50
100
150
200
250
300


0
50
100
150
200
0
50
100
150
200
250
300

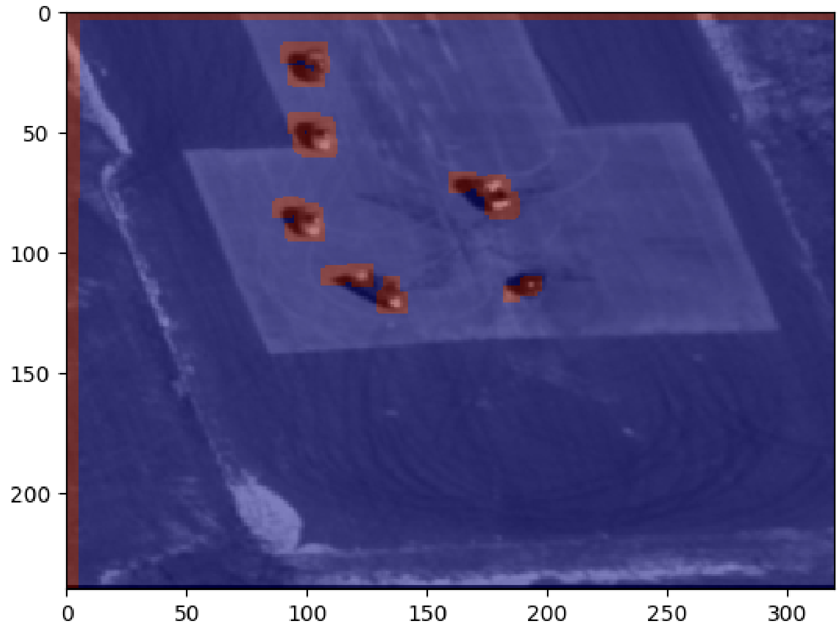

0
50
100
150
200
0
50
100
150
200
250
300



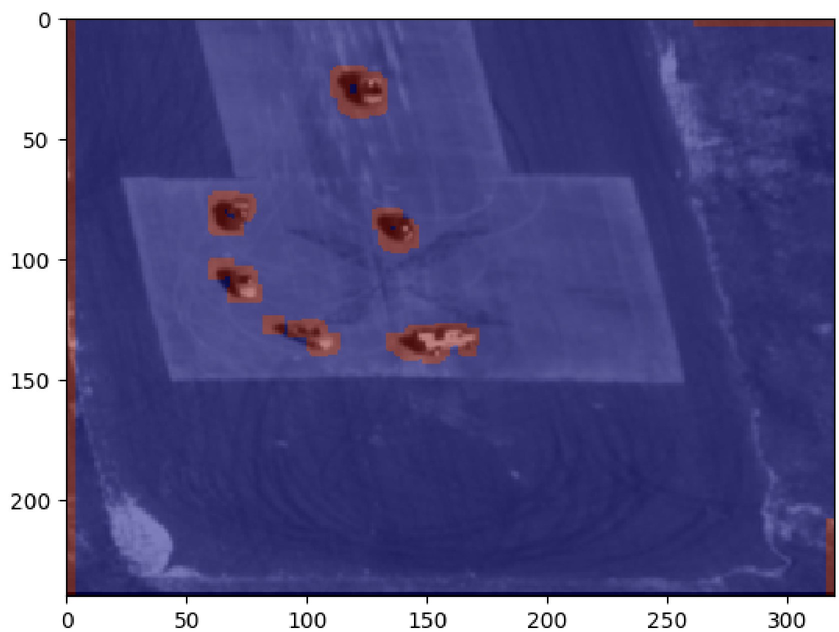
Problem 3

Q3.3

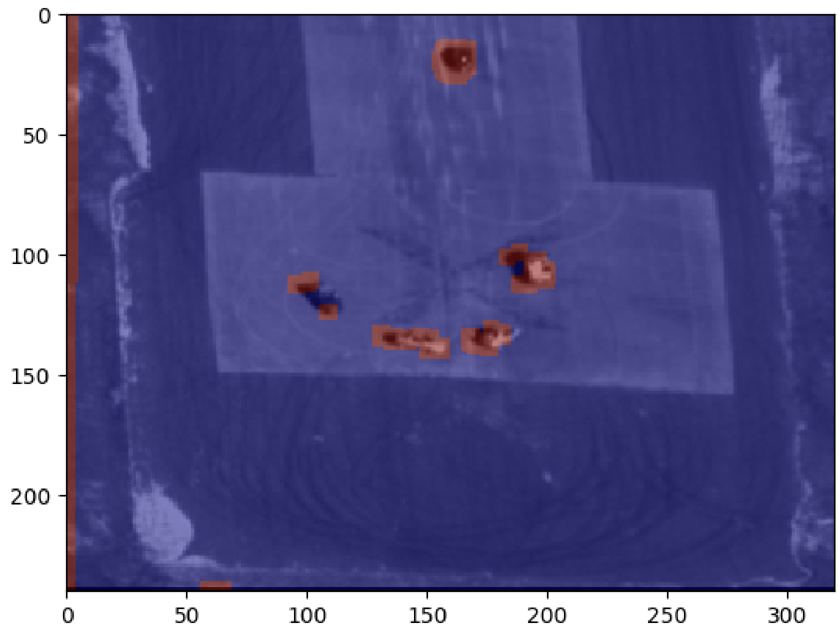
The following are results from the Lukas-Kanade algorithm with affine transformations at 30th, 60th, 90th and 120th frame. Running the script will run an animation and the part which generates these images has been commented out.

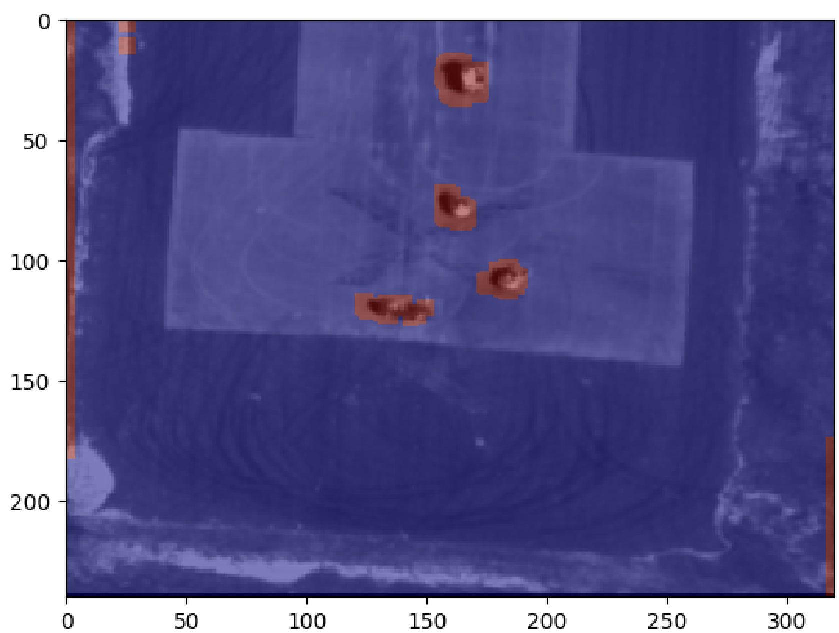


0
50
100
150
200
0
50
100
150
200
250
300



0
50
100
150
200
0
50
100
150
200
250
300


0
50
100
150
200
0
50
100
150
200
250
300


0
50
100
150
200
0
50
100
150
200
250
300

Problem 4

Q4.1

The inverse composition algorithm makes two changes to the original LK formulation. The first is to convert the additive calculations to being multiplicative and the second of changing the forward warping to that of inverse warping. Thus, this reformulation results in a minimization problem which has a linearized form like original LK, but out of the variables A and b in the least squares problem $A\Delta p - b$, only b is dependent on p unlike the forward additive LK where both A,b changes every iteration since they both depend on p. Therefore, the calculation of the A matrix can be done beforehand which makes the the inverse compositional LK much more faster and efficient.

Q4.2

We need to find the solution to the following problem:

$$\begin{aligned} f &= argmin_g 1/2||y - X^T g||_2^2 + \lambda/2||g||_2^2 \\ &= 1/2[(y - X^T g)^T (y - X^T g) + \lambda g^T g] \end{aligned}$$

Taking the derivative wrt g,

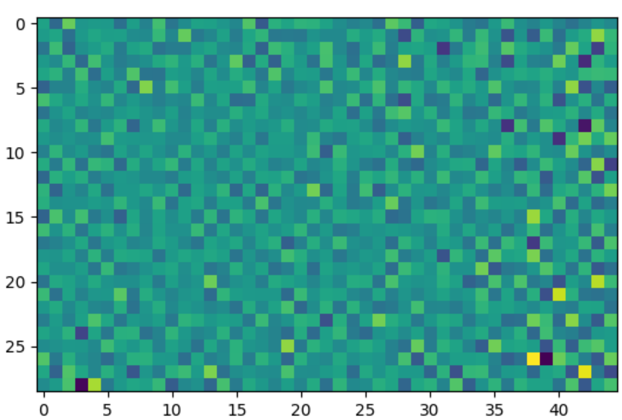
$$\begin{aligned} X^T y &= (X^T X + \lambda I)g \\ g &= (X^T X + \lambda I)^{-1} X^T y \end{aligned}$$

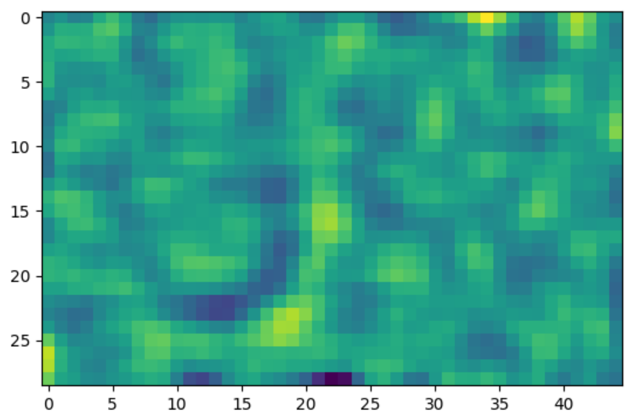
Taking $S = X^T X$,

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

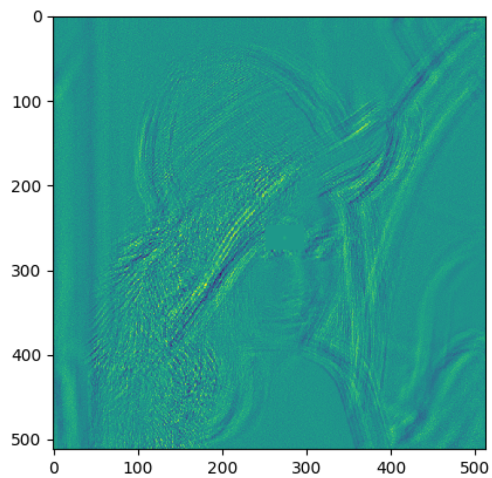
Q4.3

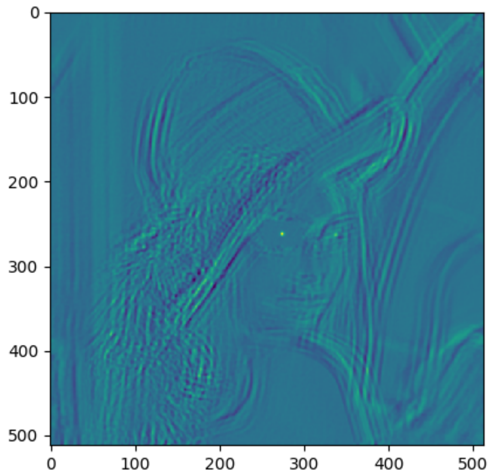
The following images are the weights visulization for $\lambda = 0$ and $\lambda = 1$ respectively.





The following are the results after running the correlation filter with the obtained weights for $\lambda = 0$ and $\lambda = 1$ respectively.

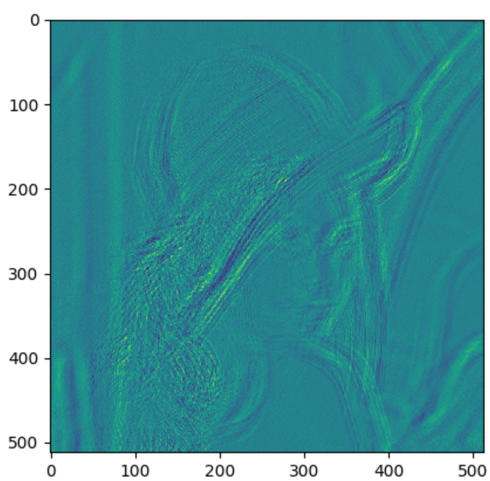


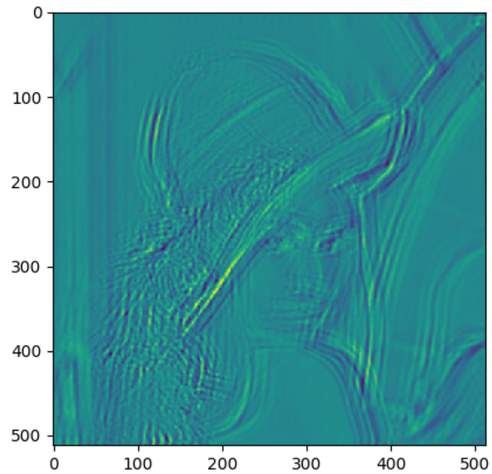


The value $\lambda = 1$ performs better because it constrains the least squares solution to a set of g's with the minimized 2-norm. In other words, by adding restrictions to the possible weights, it enforces a regularization of the minimization problem. This is specially useful if the matrices X and y are ill-conditioned and multiple analytical solutions exist for the least squares problem.

Q4.4

The following are the results after running the convolution filter with the obtained weights for $\lambda = 0$ and $\lambda = 1$ respectively.





The convolution operator doesn't give as good results as correlation does with regularization (see the sharp pixels at the eye location in the case of correlation). This is because the calculation of the filter was in terms of the correlation filter. We can use convolution operation to mimic correlation by flipping the weights upside down and left-to-right.