Q 1.1

1) $\frac{\partial \mathcal{W}(\mathbf{x};\mathbf{p})}{\partial \mathbf{p}^T}$ is the Jacobian of matrix of the warp W(x;p), warped coordinates with parameter p. The resulting matrix will be calculated by deriving the warp, in x & y, with every parameters, each p. This will result in a matrix size of 2 by the number of parameters.

   For pure translation, we can simplify the matrix to be $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ for every pixel/coord

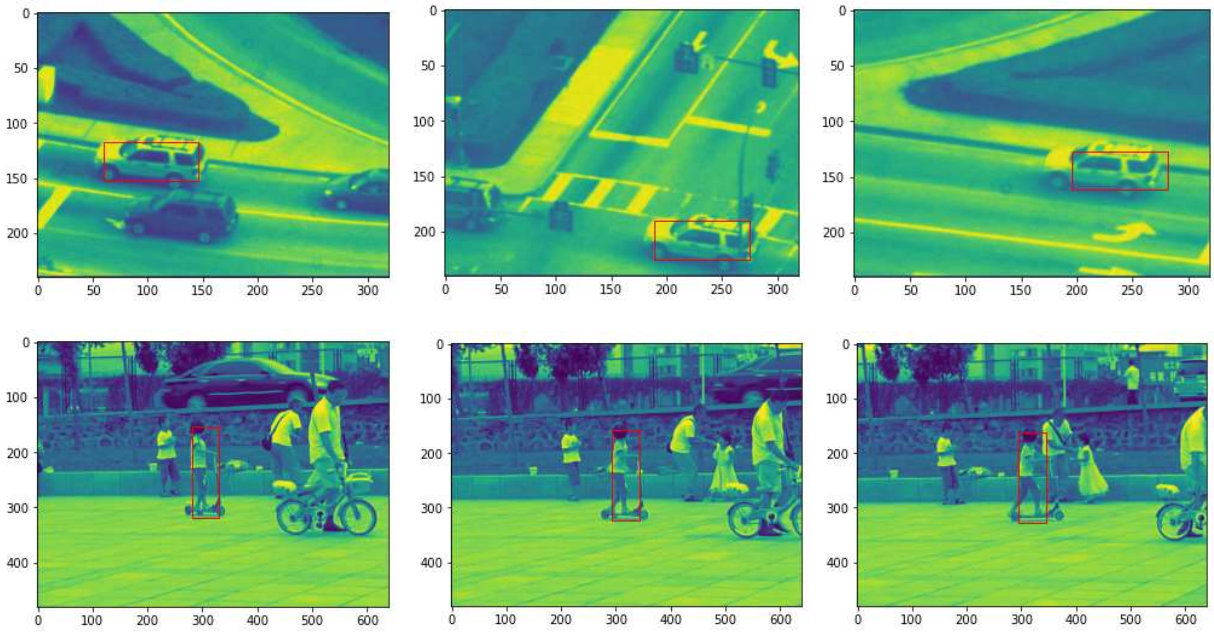
   For affine, $\frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \begin{pmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{pmatrix}$, unique for every coord

2) A is the template image that will be warped at the new coordinates with the parameter p (guess)
   B is the current image interpolated to the area.
   A and B are needed to find the error, which is used to find a suitable parameter p

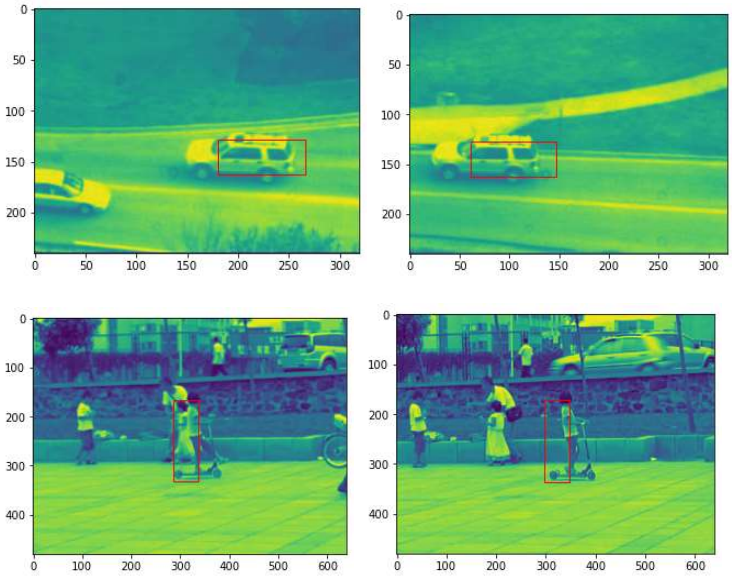
3) For unique delp to be found, delp = inv(A^TA) . A^TB.
   Therefore, A^TA should be 1 or an identity matrix

Q 1.3 LK Implementation

Num_iters do not affect the performance of the results as it only cap the amount of computational time/loop on finding the delta p. For both, num_iters is kept at 10,000. Threshold affects the results by allowing which delta p results in a 'close enough' match of the next image to the template image. The stricter the better, but it can get caught up with a different object of interest. Car uses 0.01 threshold while Girl uses 0.06 threshold. With 0.01 threshold, Girl tracking switches from the scooter boy to the girl in the white dress (background).



Frame 1 Frame 100 & 20 Frame 200 & 40



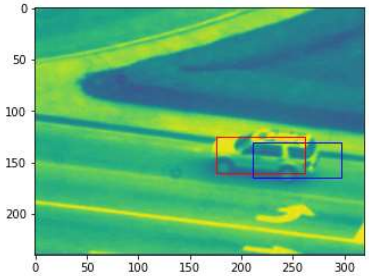
Frame 300 & 60 Frame 400 & 80

Q 1.4 LK Template Correction

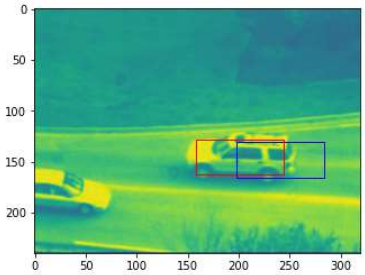
For template correction, a second lucas kanade to find P* is done and there is less chance to hit the threshold, which will go through the num_iters. The default 10000 requires too much computational time that I reduce to 300 for the Car as there is way more frames compared to Girl (400 vs 90). For threshold, I used the same value for Girl to get as accurate as possible but maintain the object to the scooter boy. On the other hand, threshold for the Car is increased to 0.68 as there is no del p that can achieve 0.01 (in Q 1.3); when del p could not converge to the threshold, it picks up the last calculated del p that is too random (very out of place). With 0.68, we can have the balance. Also, the blue rectangle (aka regular LK) performs worse than Q 1.3 with the higher threshold. All template threshold is set to 5, not too low to ignore the drift. It will be pointless to have much higher template threshold as it will try to correct frames that are minutely off.

Frame 1

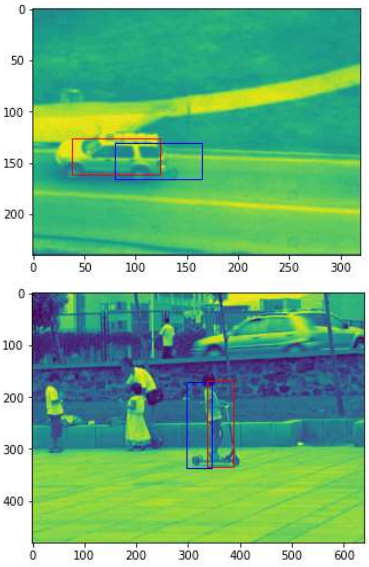
Frame 100 & 20



Frame 200 & 40



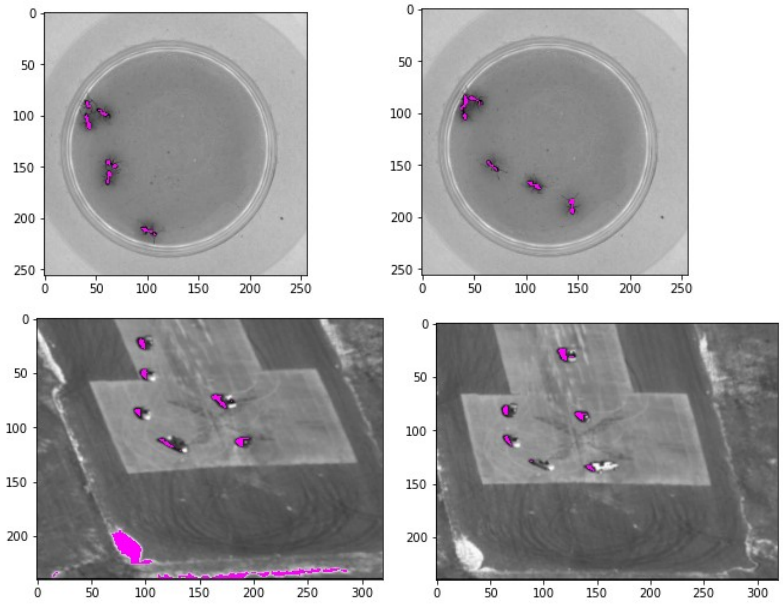
Frame 300 & 60



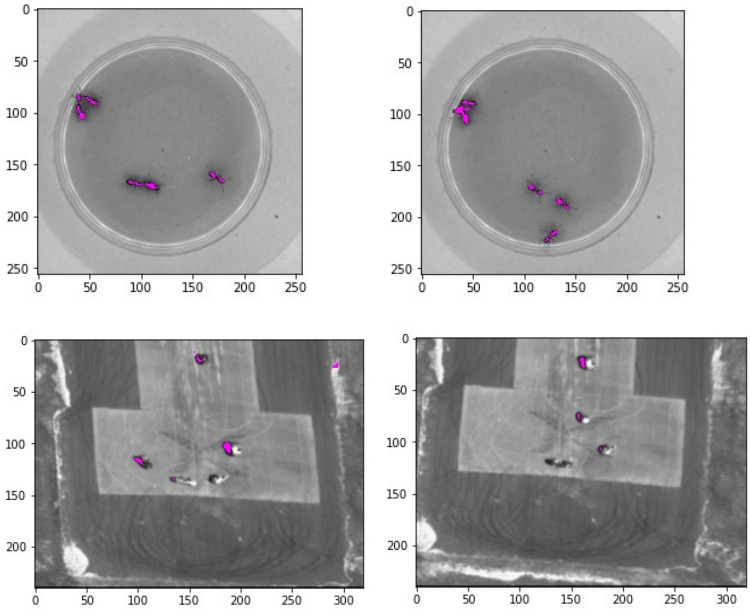
Frame 400 & 80

Q 2.3 Tracking LK Affine

For LK affine, num_iters values are 1000 and 100 while threshold values are 0.5 and 0.8 for Ant and Aerial respectively. Template threshold values are 0.73 for both. The template threshold is used to balance between showing the object of interests and hiding the moving surroundings. Ant's threshold value is kept at 0.5 to prevent highlighting its legs that gets jumbled up (blob of highlight) when lowering the value further; with 0.5, LK doesn't usually reach its full 1000 (keeping computational fast. On the other hand, Aerial is tougher to balance as beyond 0.8 threshold includes surrounding highlight while below 0.8 vanishes the highlight in cars. 100 iterations is picked due to the computational of one iteration to be 1.8 seconds, which stacks up after frame 60 (usually iterating until num_iters value).



Frame 30 Frame 60



Frame 90 Frame 120

Q 3.1 Inverse LK Affine

The inverse LK affine is more efficient & faster computationally because it precomputes the Hessian term only once compared to every loop computation in regular affine. The Hessian encompass calculating the derivatives (for x & y) & Jacobian terms (for each pixel). Having it to do once versus the num_iters variable will make substantial difference when having low threshold value. This is done in the expense of negligible inversion of the del H (from del p).

Time for LK affine takes roughly around 1.5 and 1.8 seconds for each del p iteration in Ant & Aerial respectively. Inverse LK affine takes roughly around 0.02 and 0.02 seconds for each del p iteration in Ant & Aerial. This opens up for tighter threshold tuning.