1. **Lucas-Kanade Tracking**

Q 1.1.

* It is the Jacobian matrix which varies based on the type of warping (Translation, affine, etc).

J for translation is . for affine, it is .p is the warping parameter and x is the pixel in the image.

* b =

where is the template and is the image being warped

A =

* should be Invertible for a unique solution of can be found.

**Q1.2 LK Script**

**Q 1.3 Execution of the LK Algorithm.**

The following images are the outputs for 1, 100, 200,300 and 400th frame respectively.











**Q 1.4 Template drift Correction**

The following is the Template drifting corrected execution of the LK algorithm











**Q 2.1**

Express w as a function of the image and the template and bases.

**Given:**

**The bases are orthogonal which makes the dot product with another basis 0 and an inner dot product with itself is a constant (Not 1, since they are not orthonormal)**

**= 0 (1)**

**(2)**

Bringing to the other side, we get

By multiplying Bi on both sides, we get

Taking only the RHS for now, we get

By multiplying Bi to the elements inside, we get

By given 1 and 2,

)

Thus,

**) /**

If the bases are orthonormal,

**)**

**Q 2.2 LucasKanadeBasis - script**

**Q 2.3 LucasKanadeBasis execution and output**

1st Frame



200th Frame



300th Frame



350th Frame



400thFrame

GREEN – LK with Appearance Basis RED – classic LK tracker

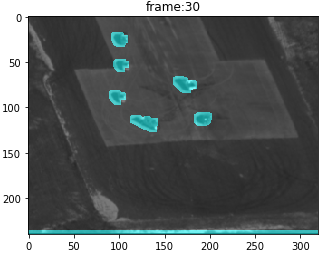
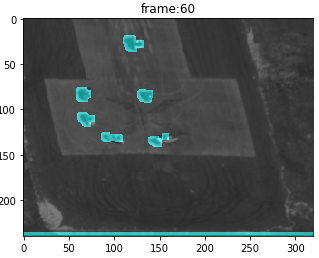
The difference in the tracking between the classicLucasKanade Tracker and the one with appearance basis is not so different from one another.

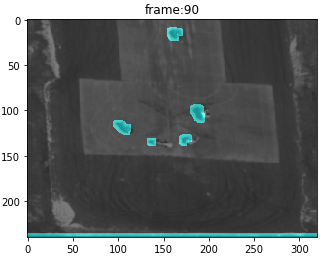
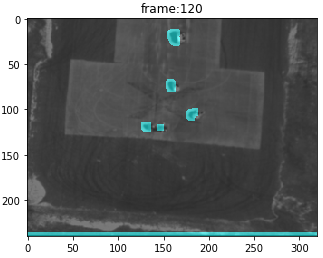
**3.1**

**3.2**

**3.3 Moving Object detection**:

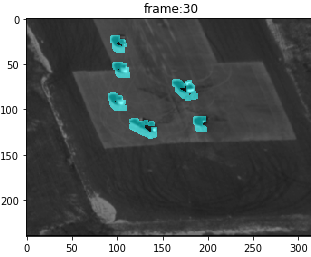
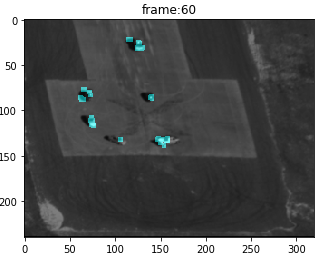
The following are the 30, 60, 90 and 120th frame of the ‘Aerialseq.npy’

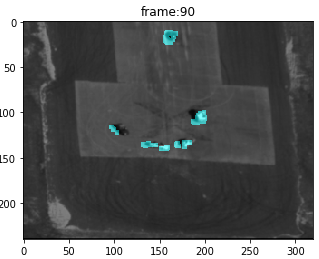
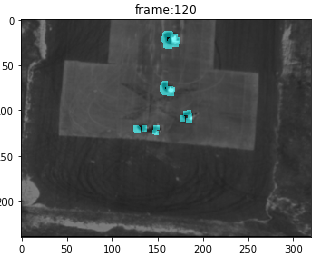




**4. Efficient Tracking**

**4.1 Inverse composition**

The difference between the forward and the inverse method is that in the forward, the Hessian is calculated for every iteration (i.e.) The Matrix A is calculated for every iteration whereas in the Inverse composition the A matrix is calculated outside the loop, only once. This is made possible by inverting the roles of the template and the image. By doing that, the Hessian Matrix is calculated only one per frame and the computational efficiency is increased.



**4.2 Correlation Filters.**

E(g) =

In order to minimize ‘g’, we take the derivative of E with respect to ‘g’

By equating the derivative to zero, we get

Rearranging,

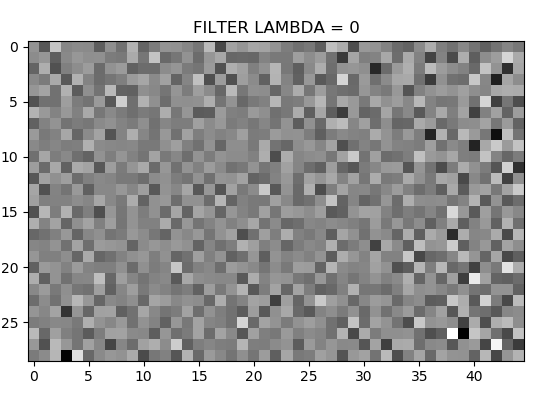
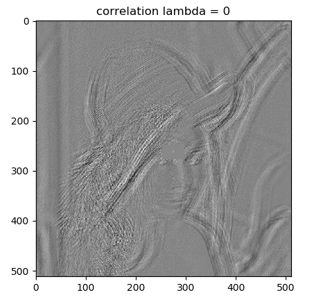
-) = 0

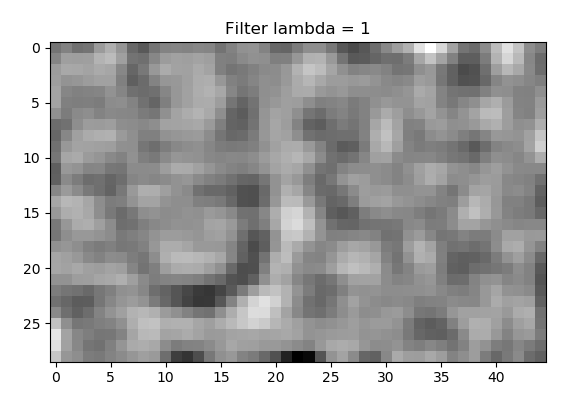
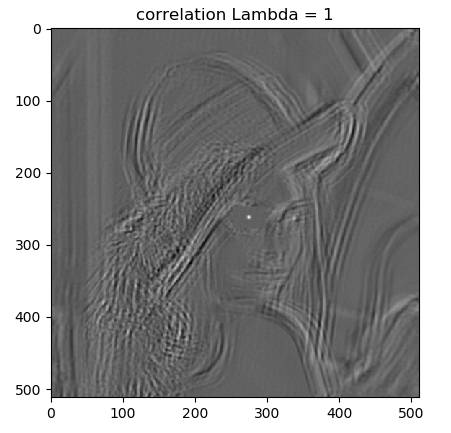
Taking g to the other side,

g =

The correlation filter in terms of S, X and y is shown below

Since,

**4.3 Correlation for different lambda values, 1 and 0**:



The above figures are the output from the correlation filter for lambda values of 0 and 1 respectively.

The value of lambda when it is 1 performs better. It is due to the fact that it prevents overfitting, whereas the regularising parameter is not in the picture when it is 0 which is obvious and leads to overfitting.

**4.4 Convolution for different values of Lambda, 1 and 0**

The Kernels are flipped while doing convolution which is the difference between the computation of Correlation and Convolution.

The following are the difference between the output with and without flipping of kernels.

Flipping of kernels is doing by flipping the NumPy array values using ‘np.flip’

