16-820 Advanced Computer Vision
Homework 2: Lucas-Kanade Tracking

Abhinandan Vellanki

1 Lucas-Kanade Tracking

1.1

1.1.1

$\frac{\partial W(x:p)}{\partial p^T}$ is the Jacobian matrix which is a matrix of partial derivatives of the image coordinates with respect to the warp parameters. The values of this matrix depend on the coordinates of the pixels and the warp parameters. For this translation warp, the Jacobian matrix is:

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

$$\frac{\partial W(x:p)}{\partial p^T} = \begin{bmatrix} \frac{\partial W_x}{\partial p_x} & \frac{\partial W_x}{\partial p_y} \\ \frac{\partial W_y}{\partial p_x} & \frac{\partial W_y}{\partial p_y} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

where 'x' represents the image coordinates (x, y). Hence, the Jacobian is computed for every pixel. Here, it is the same for all pixels since it is independent of the image coordinates.

1.1.2

A here is the matrix of steepest descents in the least squares problem. It is computed by multiplying the image gradients with the Jacobian matrix:

$$A = \nabla I \frac{\partial W(x:p)}{\partial p^T}$$

b here is the residual error to be minimized, i.e., the difference between the template being tracked and the warped image.

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

where 'x' represents the image coordinates (x, y). The A matrix and b is also computed for every pixel.

1.1.3

For a unique solution for Δp to be found, $A^T A$ must be invertible, i.e., $Det(A^T A) \neq 0$

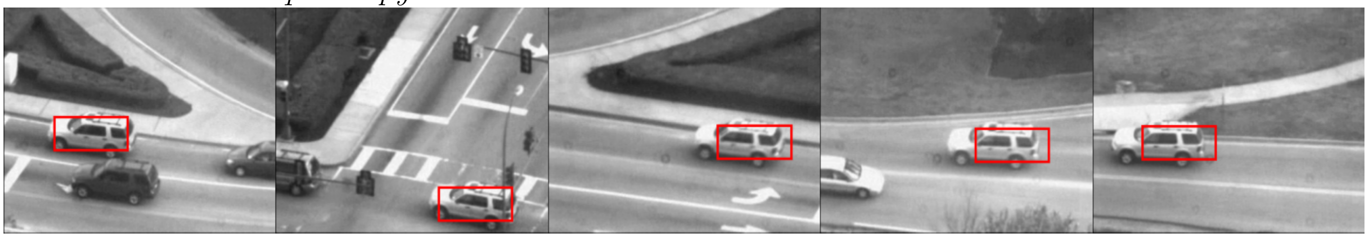
1.2

LucasKanade.py attached in **abhinanv_hw2.zip**

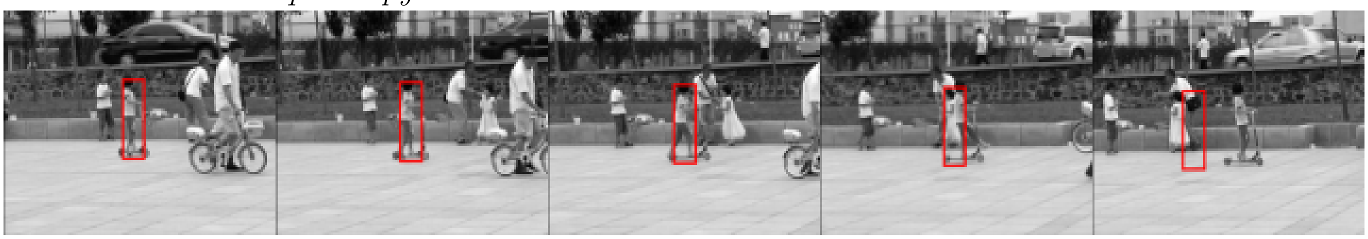
1.3

Attached *testCarSequence.py, testGirlSequence.py, carseqrects.npy, girlseqrects.npy* in **abhinanv_hw2.zip**.

Results of *testCarSequence.py*:

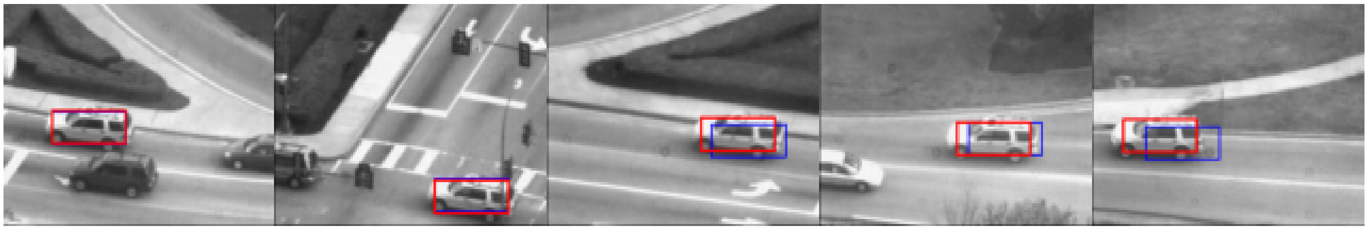


Results of *testGirlSequence.py*:

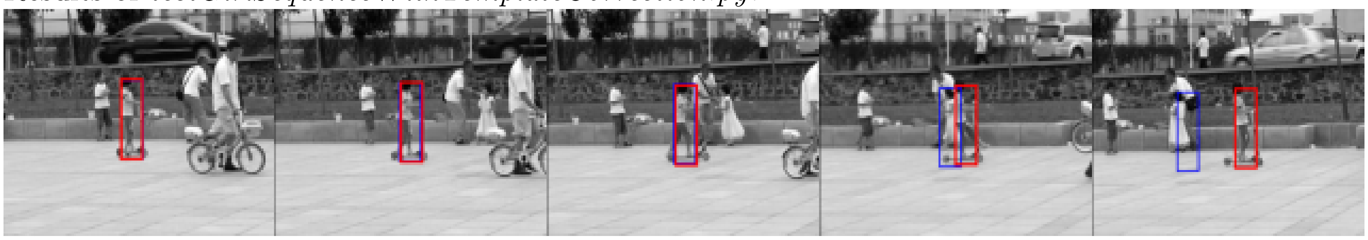


1.4

Attached *testCarSequenceWithTemplateCorrection.py, testGirlSequenceWithTemplateCorrection.py, carseqrects-wcrt.npy, girlseqrects-wcrt.npy* in **abhinanv_hw2.zip**.
Results of *testCarSequenceWithTemplateCorrection.py*:



Results of *testGirlSequenceWithTemplateCorrection.py*:



2 Affine Motion Subtraction

2.1 Dominant Motion Estimation

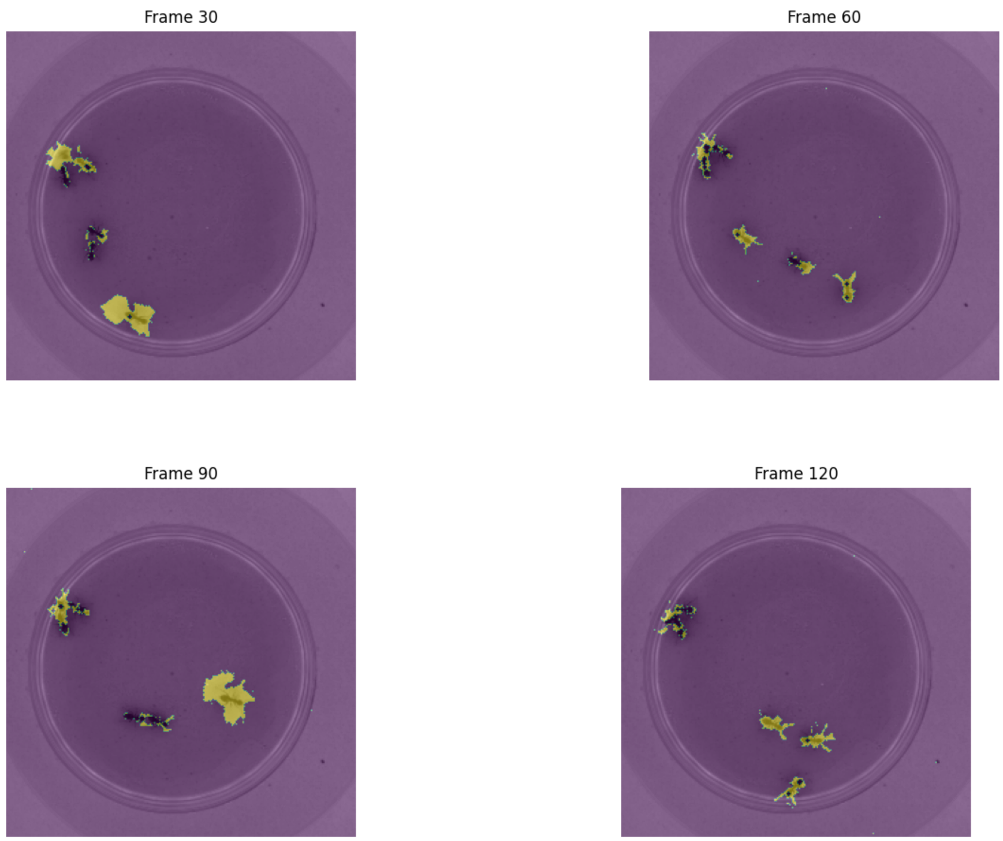
LucasKanadeAffine.py attached in **abhinanv_hw2.zip**

2.2 Moving Object Detection

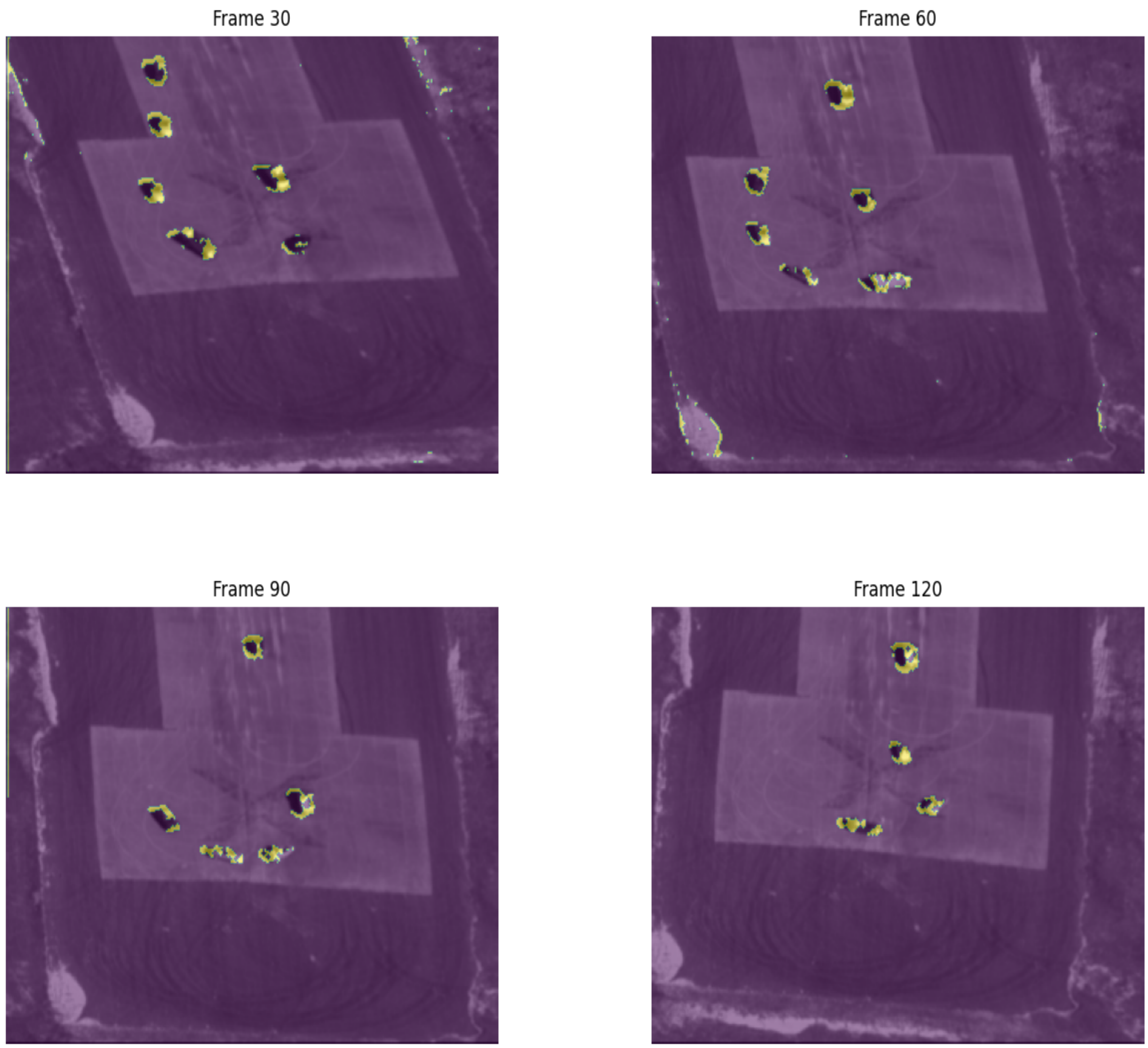
SubtractDominantMotion.py attached in **abhinanv_hw2.zip**

2.3 Moving Object Detection

testAerialSequence.py and *testAntSequence.py* attached in **abhinanv_hw2.zip**
Results of *testAntsSequence.py*:



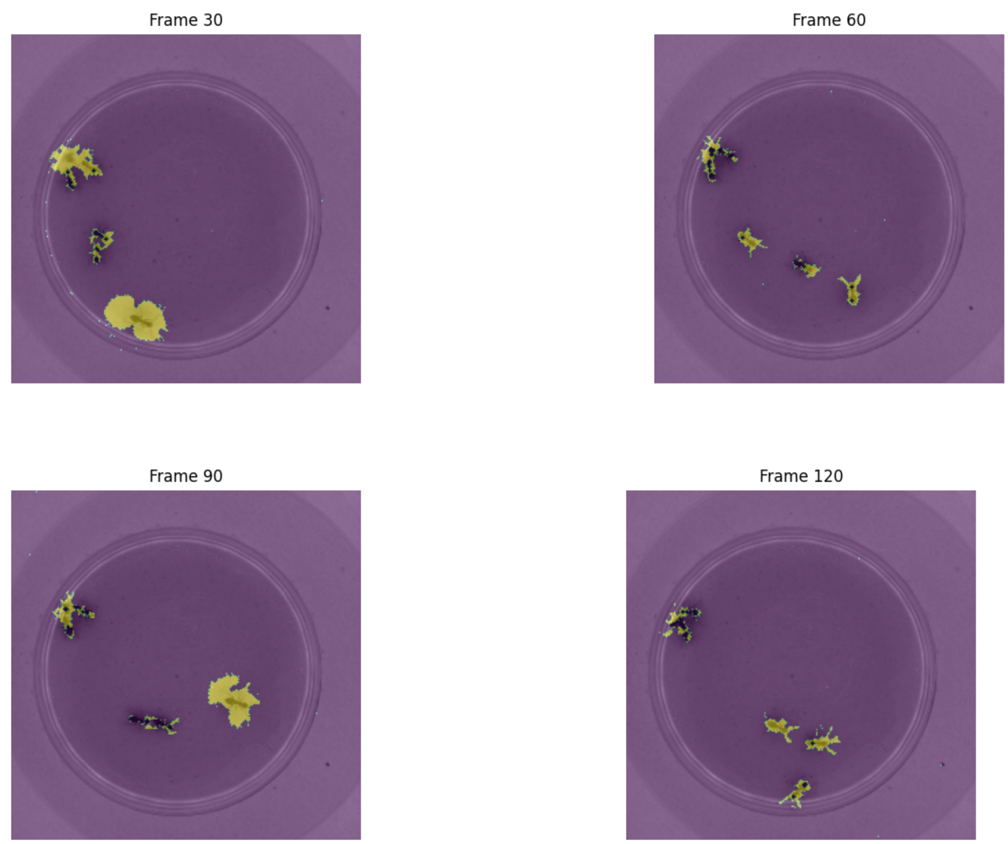
Results of *testAerialSequence.py*:



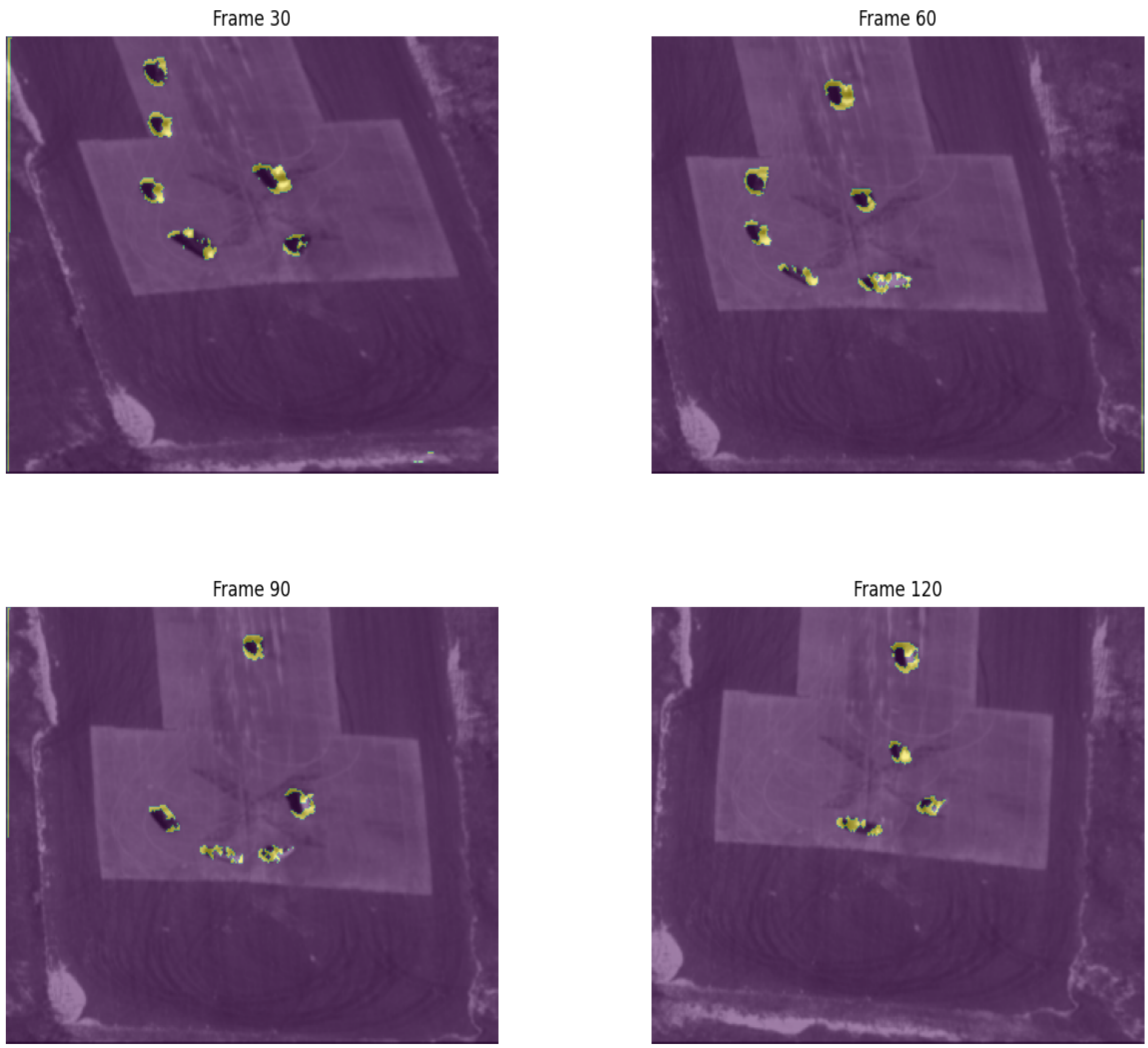
3 Efficient Tracking

3.1 Inverse Composition

InverseCompositionAffine.py attached in **abhinanv_hw2.zip**
Results of *testAntsSequence.py*:



Results of *testAerialSequence.py*:



3.2

The inverse composition alignment algorithm is more efficient than the additive alignment algorithm because of the lower number of computations performed in every gradient descent iteration.
In this algorithm, by warping the template and computing its gradient before the iterative process, the Hessian can be computed before the iteration as well, leaving only the computation of the residual error in each gradient descent iteration.

4 References

4.1 Collaborators:

- Ayush Fadia
- Abhishek Iyer
- Yu Jin Goh
- Yang Wang
- Parth Gupta
- Ranai Srivastav

4.2 Code

Attached in **abhinanv_hw2.zip**

—THE END—