1. **Lucas Kanade Tracking**

* What is A and b?

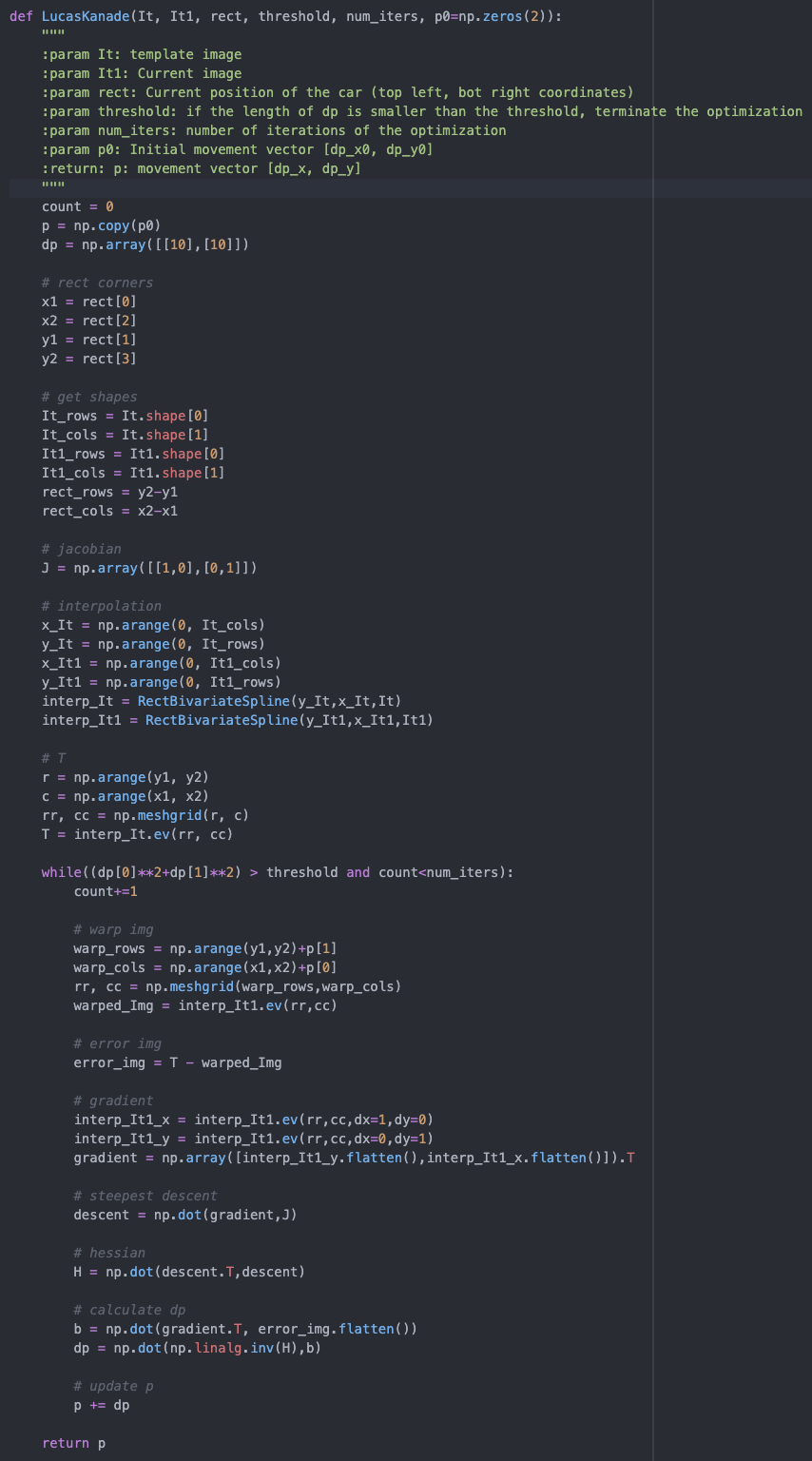
A is the steepest descent, which is the dot product of the gradient and the jacobian. B is the error image which is the image subtraction between the template and the warped image.

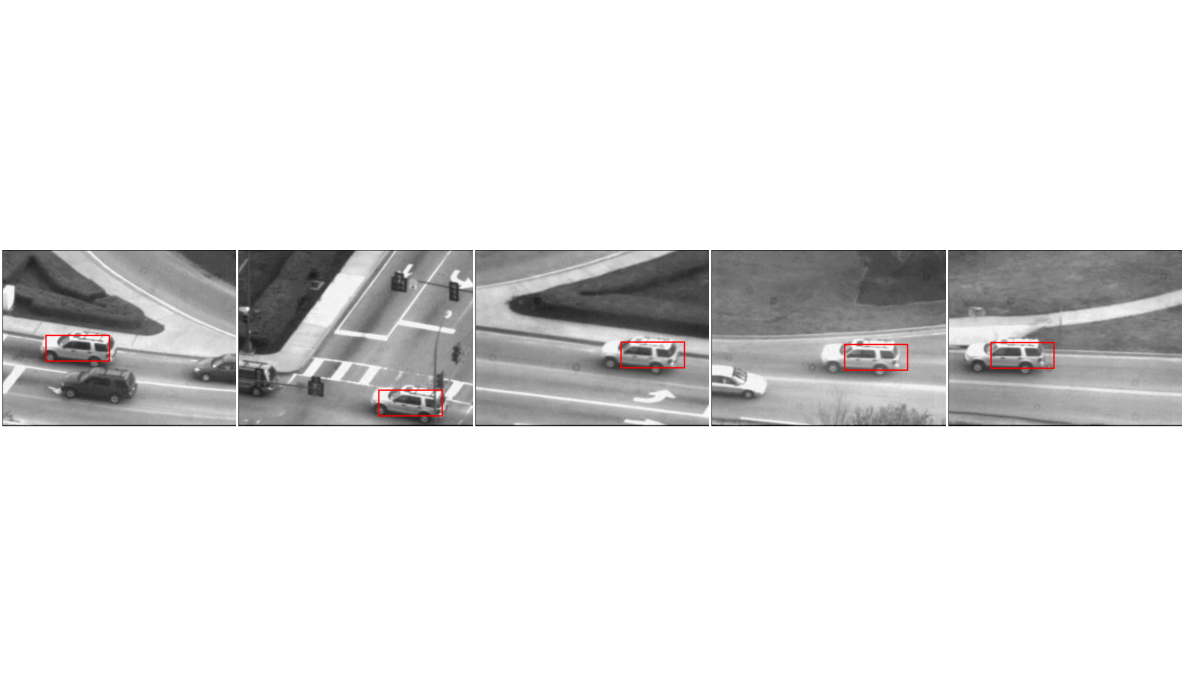
* What is dW/dp (should a 2x2 matrix)?

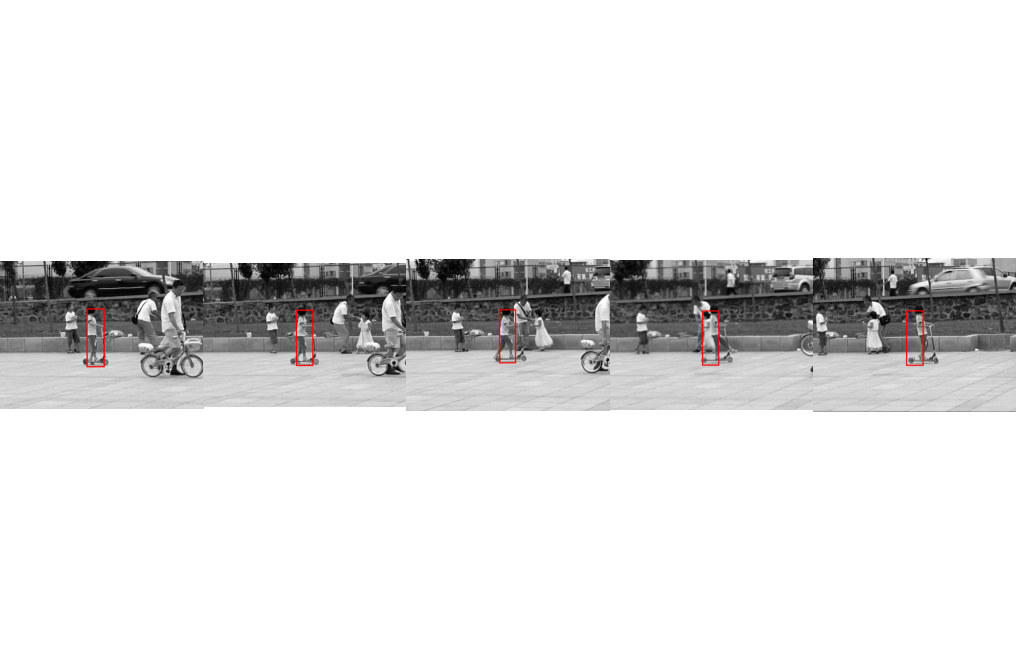
The jacobian, which is the partial derivative of our warp function. For translational warping it’s just [1 0; 0 1].

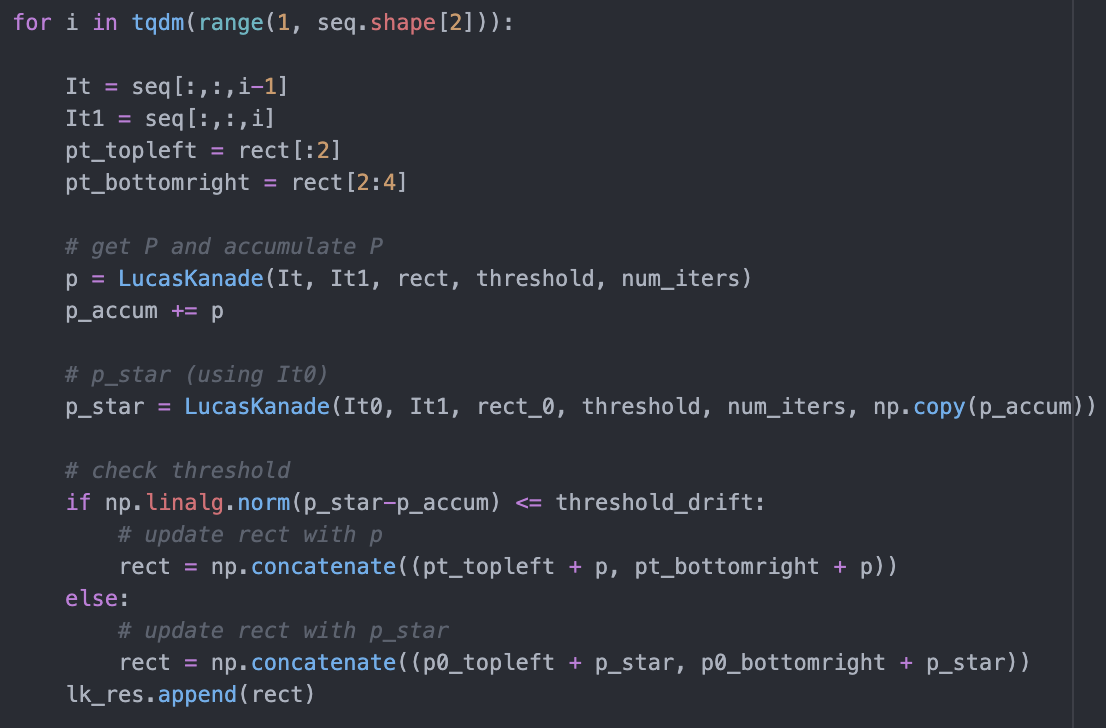
* What conditions must AT A meet so that a unique solution to ∆p can be found?

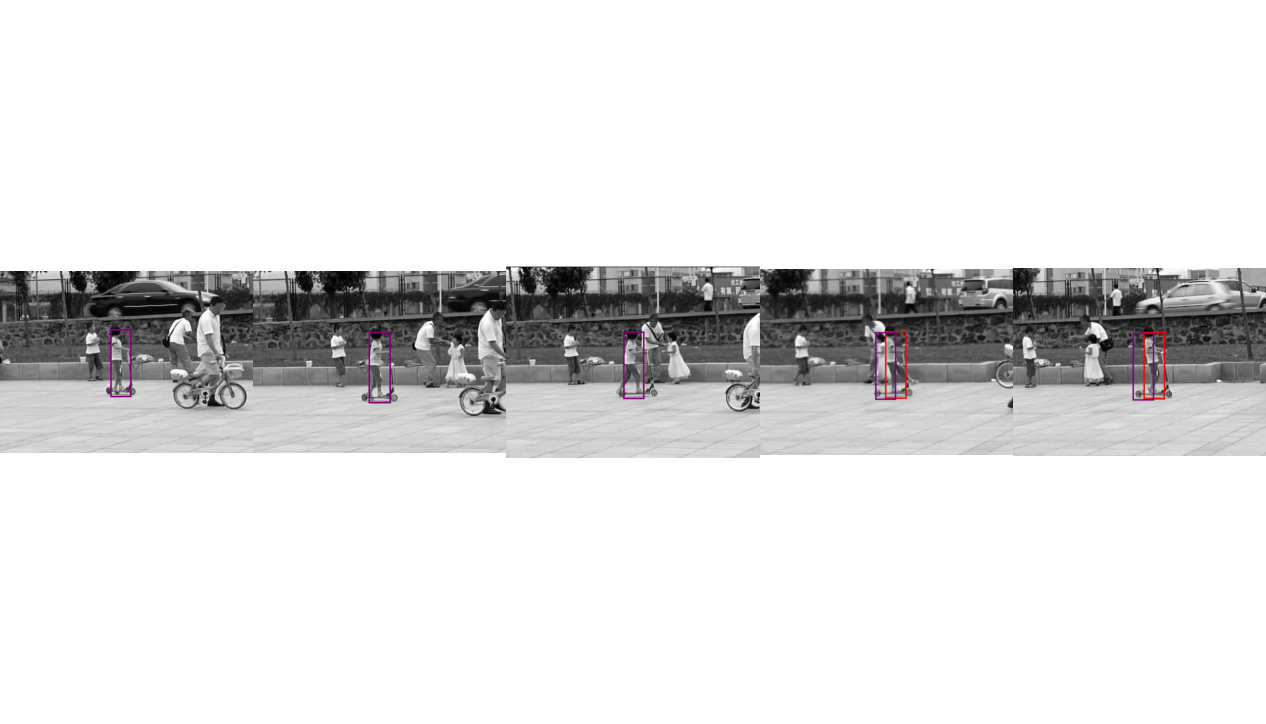
A must be invertible and well-conditioned. For it to be well-conditioned, the eigenvalues of A must be above a certain threshold, and one of the eigenvalues must be greater than the other but not significantly.

* 1. 





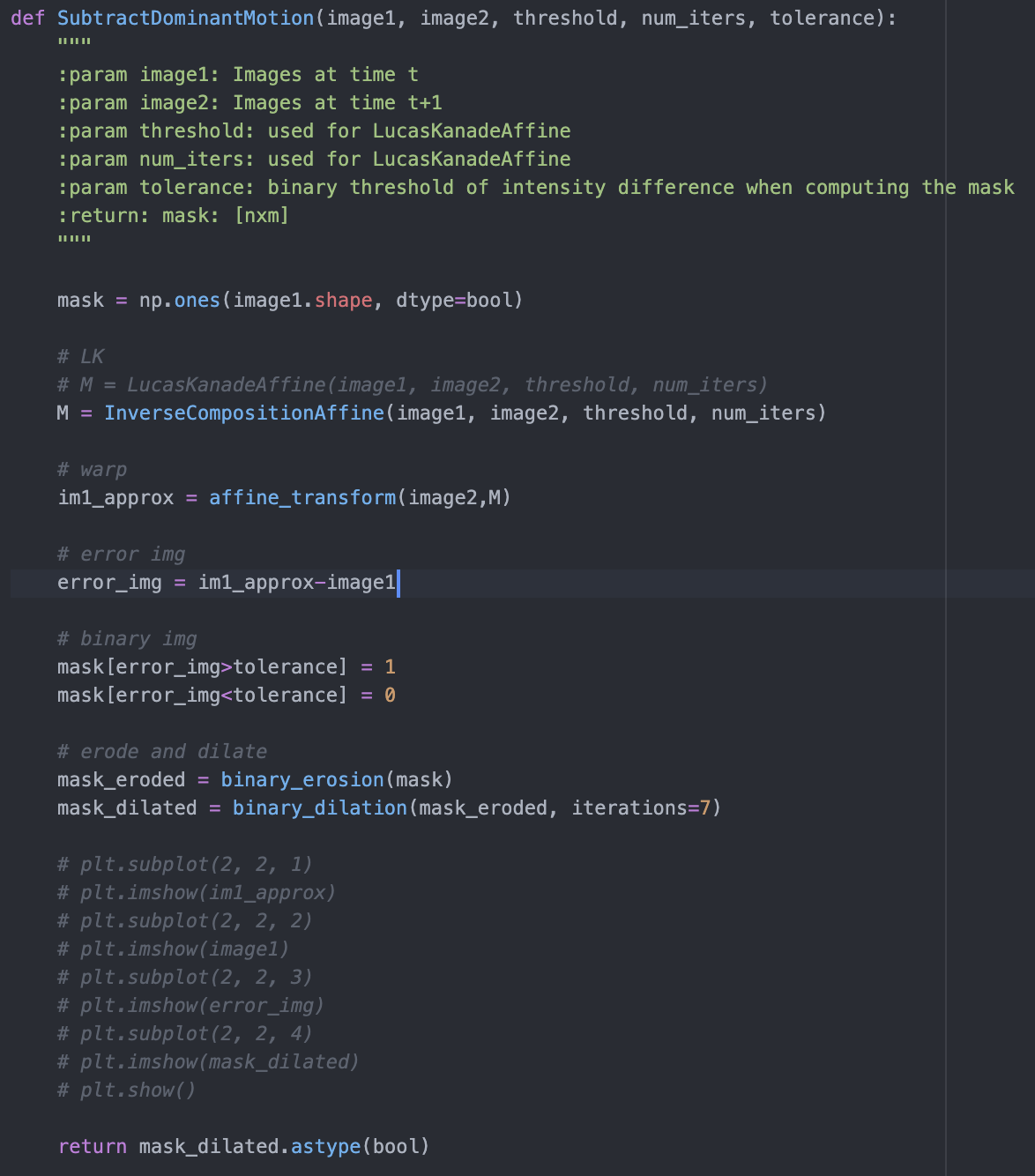


A picture containing tool

Description automatically generated

1. Affine Motion Subtraction
   1. Text

      Description automatically generated



A picture containing electronics, wall, loudspeaker, white

Description automatically generated

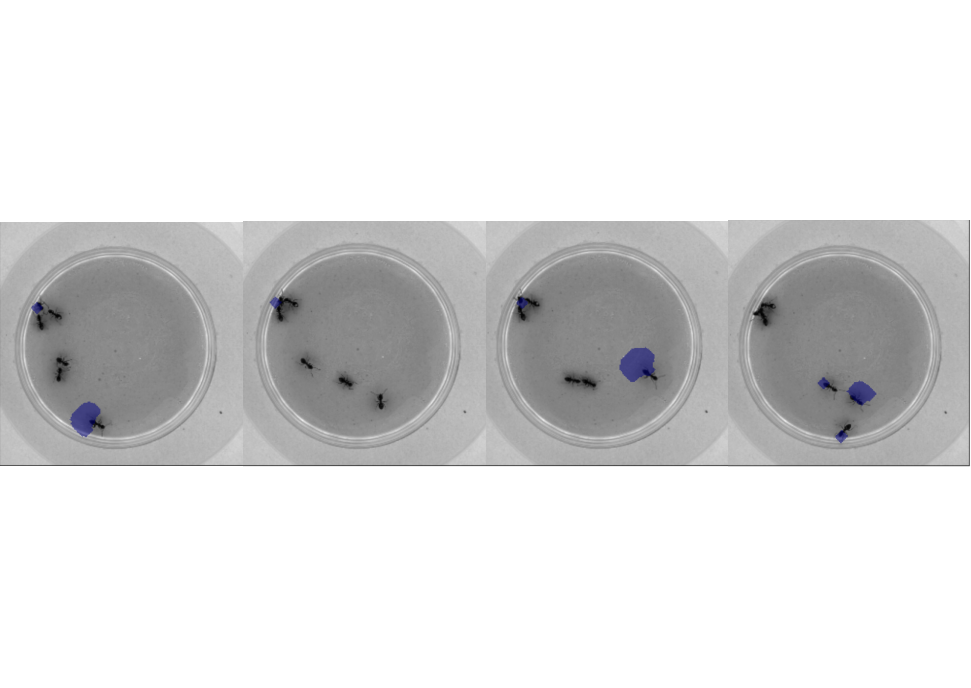
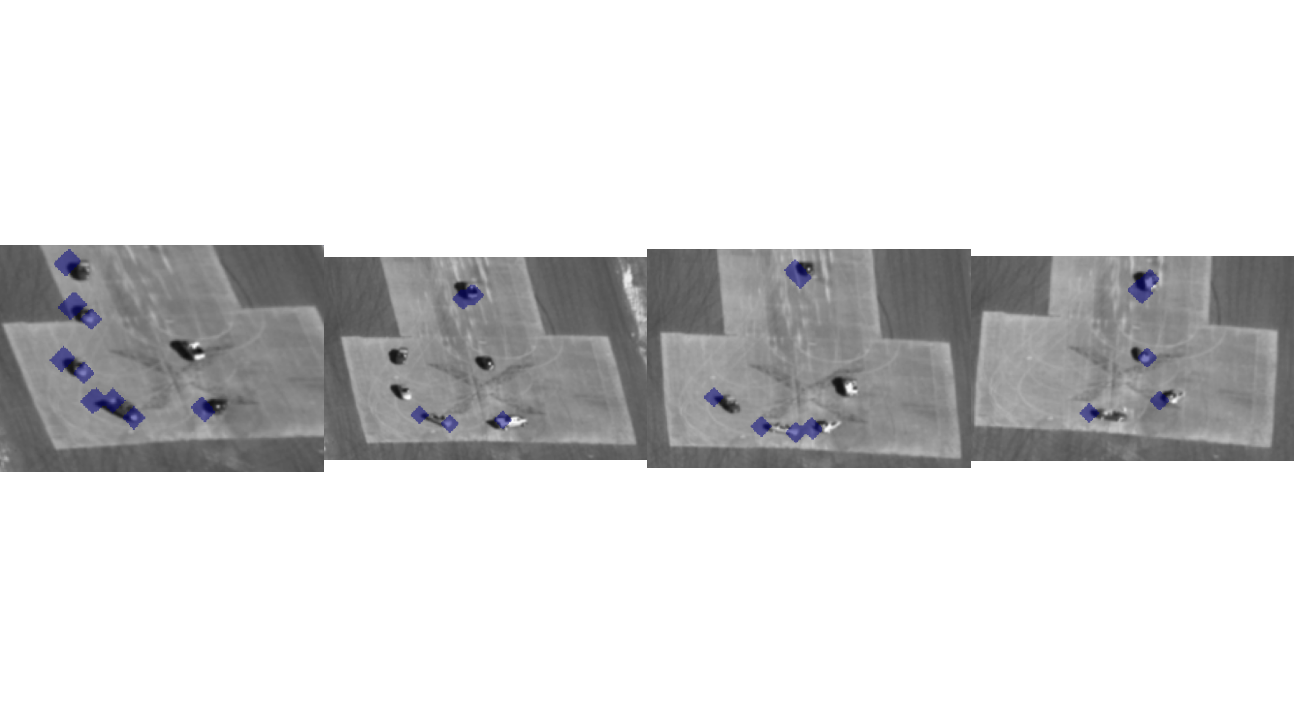
Graphical user interface, application

Description automatically generated with medium confidence

1. Efficient Tracking
   1. The runtime performance of the inverse compositional Lucas Kanade affine method is much better than the runtime performance of the original Lucas Kanade Affine transform method. In the inverse compositional method a change of variables is made to invert the roles of the template and the image. Through utilizing an inverted incremental warp, we are able to precompute the gradient, jacobian, steepest descent, and hessian. We are able to save a huge amount of computation by avoiding recalculating these values in every iteration.

Text

Description automatically generated



1. Extra Credit

I filmed a video with a salient foreground object undergoing a considerable amount of motion. I utilized my inverse compositional Lucas Kanade affine transform algorithm to detect the moving object. First I converted the video to a .npy file. Then, using the existing testAerialSequence.py script, I made a similar file (testVideoSequence.py) for the new video to capture the frames of the tracking. To achieve desirable results with the new video, I had to change the tolerance parameter used in the SubtractDominantMotion function. I lowered the tolerance and additionally had to increase dilation in order to capture the full moving object. Below are the results of my inverse compositional Lucas Kanade Affine Warp Alogirithm using the new video.

A group of people in a large room

Description automatically generated with low confidence