CIS 581 Project 3B: Optical Flow

-Arnav Dhamija and Saumya Shah

1. **Running the code**

   Run the following commands

   ```
   python3 getFirstFrame.py vids/Easy.mp4 # generate first frame PNG
   labelImg first.png # create bounding box and save points
   python3 main.py # generate the video's frames in intermediate/
   ./create_video.sh # create an AVI video of optical flow
   ```

2. **Initial Bounding box**
   The bounding box was created using the labelImg tool and the coordinates from the XML file were extracted using the getBoundingBoxes.py code.

3. **Feature Detection**
   2 feature detectors were used for the purpose:

   a. corner_shi_tomashi (cv2.goodFeaturesToTrack)
      This was used to get the initial features inside the bounding box, since the features given by this were of superior quality.

   b. corner_harris (cv2.cornerHarris)
      This was used to detect new features when we lost the features. This was used as we wanted to retain the previous features which were not lost and get replacements only for the lost features. Thus, the corner response matrix from the corner_harris matrix helped us customize the feature selection so as to retain the old features and add new ones which were still far apart using anms.

4. **Feature Tracking**
   We had used the KLT tracker for finding the translation of the feature points.
   The iterative refinement was performed until the u and v converged or the number of iterations reached the maximum number of iterations.

   The features were discarded if the translational values were found to be more than a set threshold which was set differently for fast moving and slow moving videos.

5. **Object Tracking and Bounding box**
   The objects are tracked using a bounding box which surrounds the original object. This bounding box is calculated from the original bounding box which is given by the user in the first part. Using the motion of the points we can obtain a homography using RANSAC to find a transformation of the corners of the bounding box to the new image. However,

these corners may not be aligned with the axes. To correct this, we take the four corner points and find an axis aligned rectangle which touched all the edge of the quadrilateral.

As each box in the image has its own tracked features and coordinates, we use a key-value pair to store the coordinates of its corners, the X and Y coordinates of its features and the currently valid features in the image.

6. Results

Easy Video

1st Frame

50th frame

100th frame

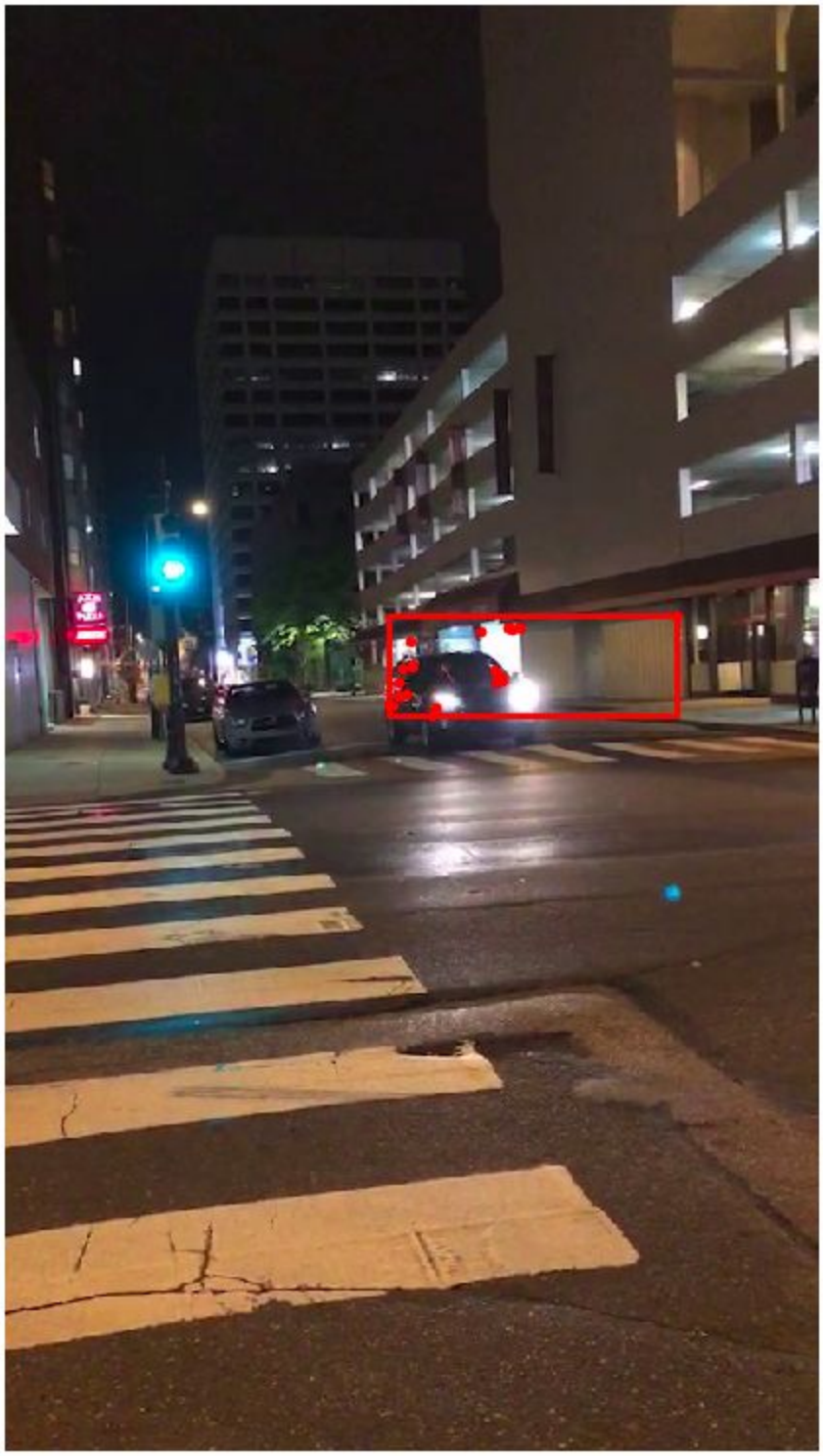
150th frame

250th frame

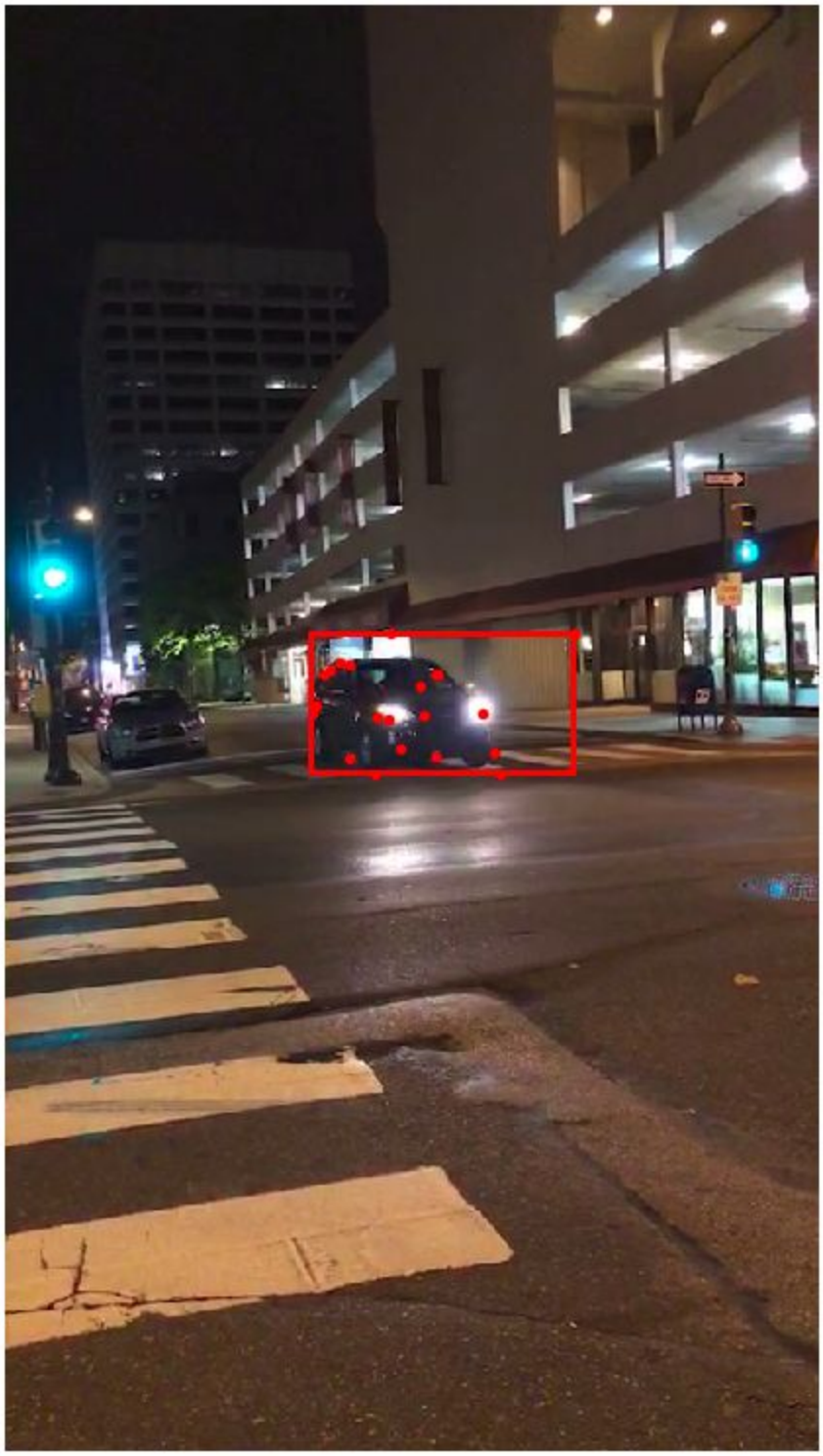
309th frame

Medium Video

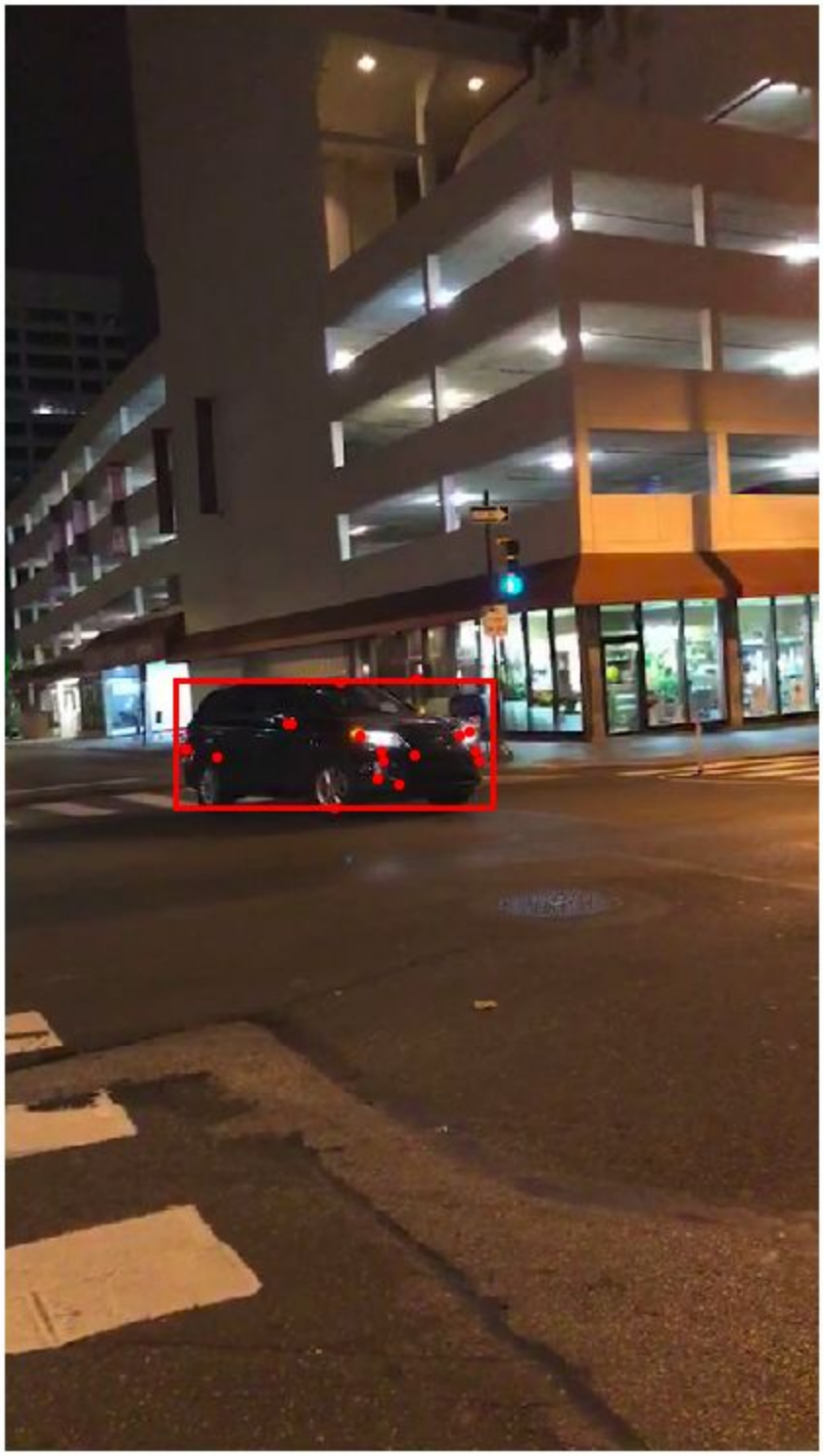
1st frame



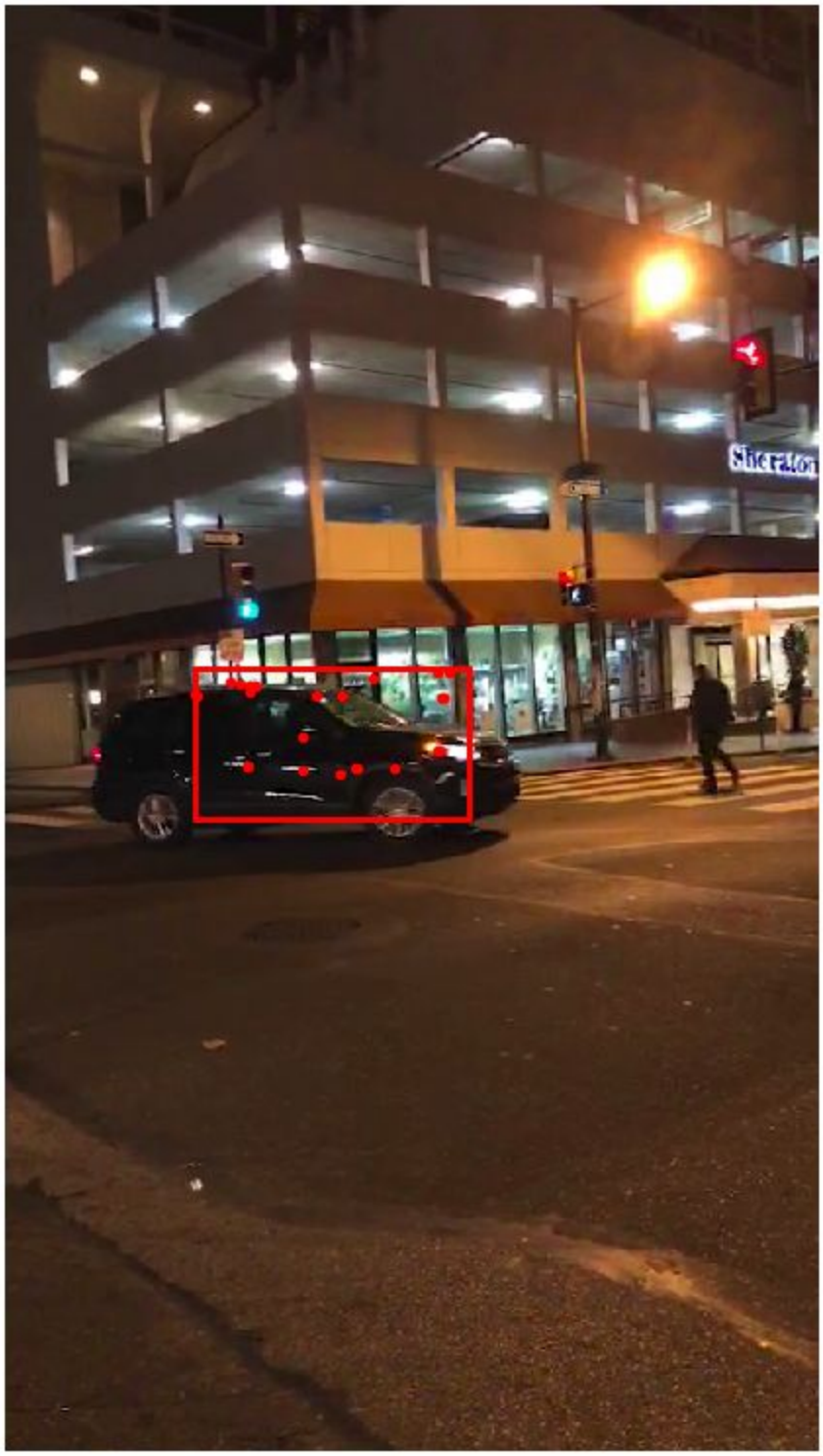
90th Frame



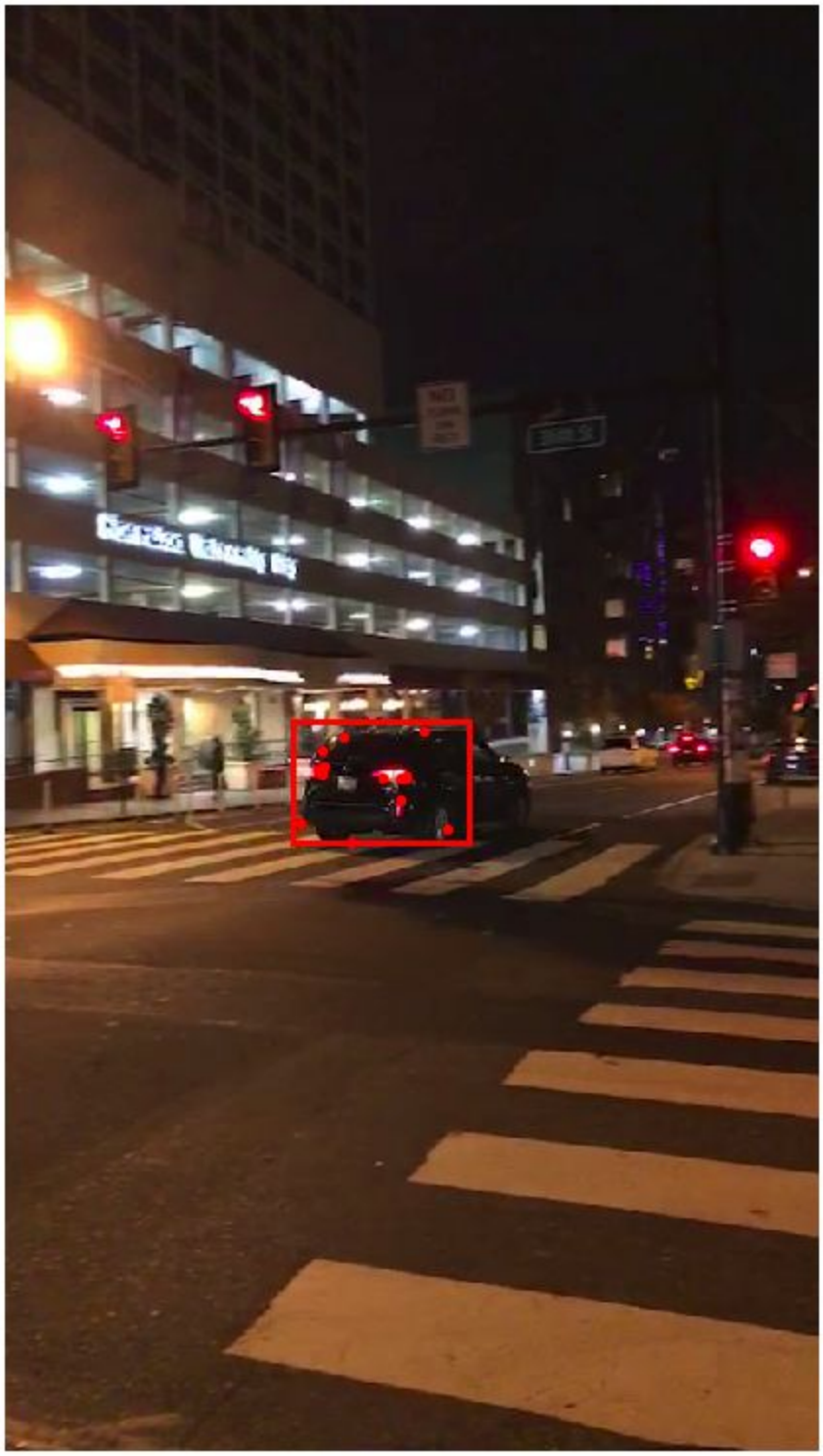
128th Frame



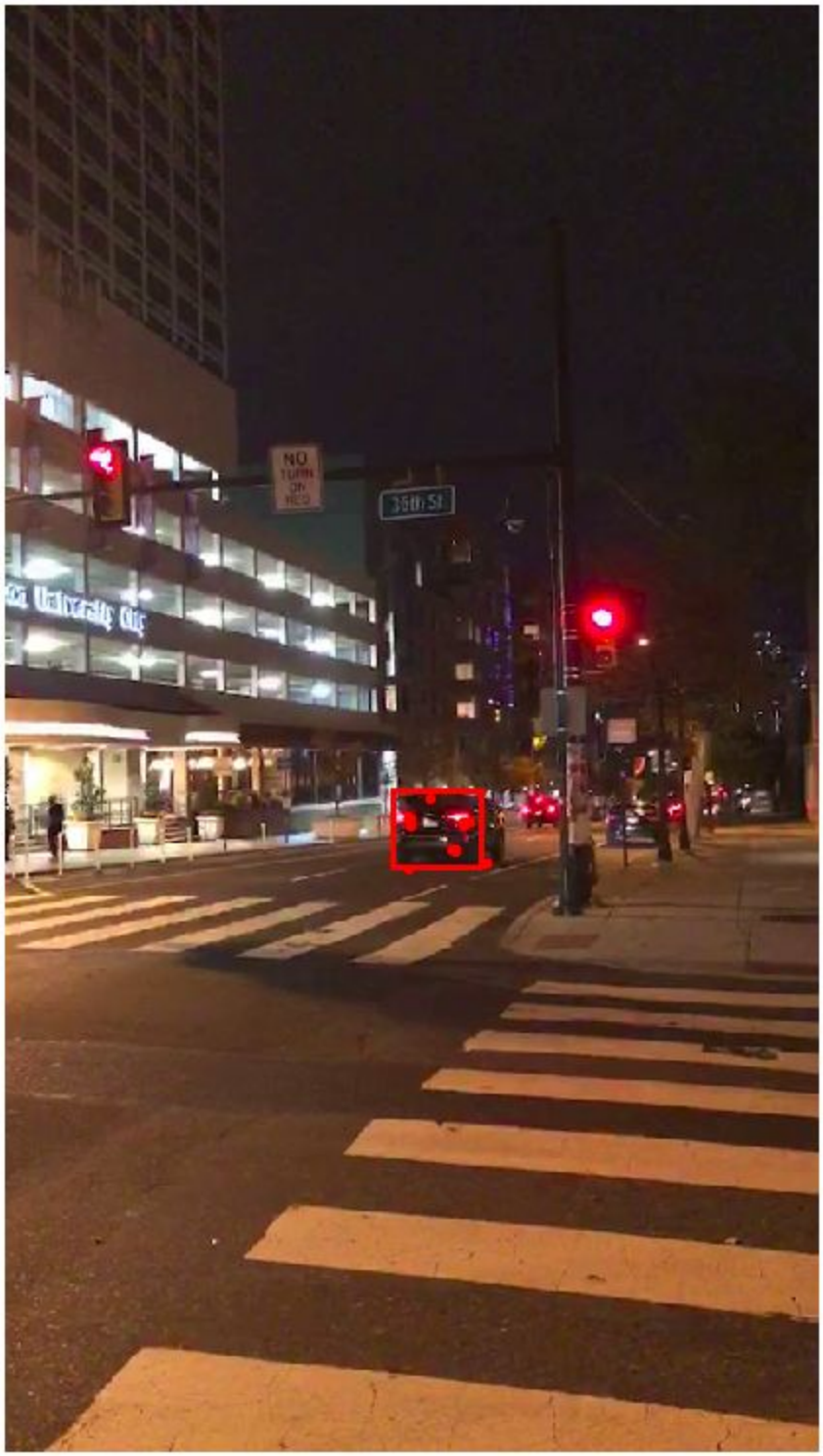
200th Frame (Middle Frame)



250th Frame



317th Frame



369th Frame

Hacks and Tricks:

1. The larger window size was used for larger context as in the case of the bigger objects like the features on the first car in the easy video and after different trials, the best values for the Gaussian derivative and its size were found out to be:

   For Medium Video:
   - Window size: 3
   - Gaussian kernel size: 5
   - Distance threshold < 4 (point suppressed if drifted a lot)
   - Number of features: 25
   - Shi-Tomashi feature detector parameter(n,0.01,10)
   - Refreshed when the feature count fell below 20

   For Easy Video:
   - Window size=5
   - Gaussian kernel size: 5
   - Distance threshold<4 (point suppressed if drifted a lot)
   - Number of features: 10
   - Shi-Tomashi feature detector parameter (n,0.01,10)
   - No refreshing of features required

2. Iterative Refinement
   The iterative refinement was performed under 2 constraints: i.e. if the u and v converged or if the maximum number of iterations was reached.
3. Refreshing the features:
   - The refreshFeatures.py does the refreshing of the features.
   - The criteria for refreshing varies from case to case. For the easy video, we did not need any refreshing as the points were not lost. For the medium video, starting with 25 points, the criteria was to refresh whenever the points went below 20 and add the required number of points to make it 25. The numbers were various obtained by trials.
   - Refresh features uses the corner harris detector to find the new feature points in the bounding box at that frame.
   - The corner response matrix is then passed through the anms_mod function along with the old feature points. The old feature points are given the priority and the new features are added thus, yielding the retained points along with the new points which are uniformly spread across the bounding box.
4. Bounding box:
   The bounding box is meant to show the motion of the tracked object. We tried doing this in two different ways - through using a homography obtained from the motion of the points and creating a box using the minimum and maximum X and Y coordinates of the feature points. We found that the former approach worked well for the easy video and the latter helped us for the medium video. This bounding box is used deciding the region

of updating the points. We also found that refreshing the bounding box every frame worked well for the Easy video but refreshing it every ten frames for the Medium video gave us better results.

5. Medium video:
   The medium video was processed from the middle frame. The optical flow was measured from the middle frame to the first frame in reverse and from the middle frame to the last frame conventionally. This was done as it was easier to detect consistent features in the middle frame which had an angled view (2-Point perspective view) of the car (unlike just the front as in the first frame or the back as in the last frame).