# ENPM 673 P4

SUBMITTED BY,

SAKET SESHADRI GUDIMETLA | CHAYAN PATODI | PRASANNA MARUDHU BALASUBRAMANIAN

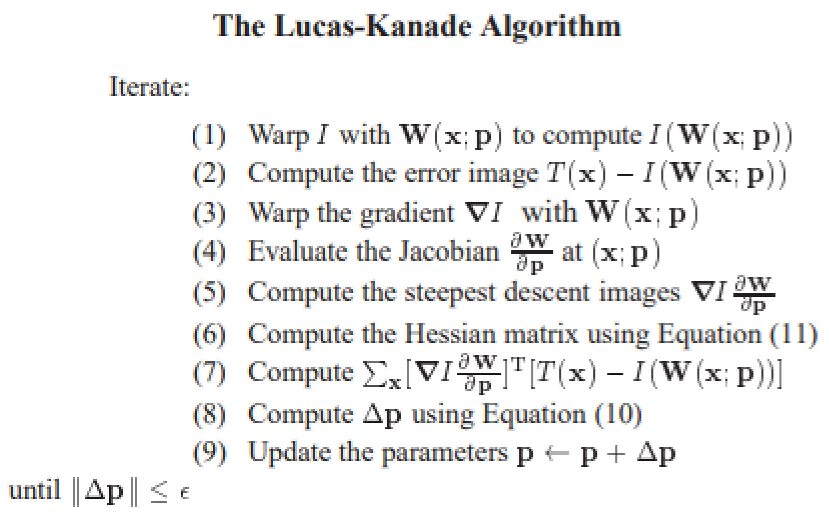
# Lucas Kanade Template tracker

## Introduction

The Lucas-Kanade tracking algorithm is a tracker built for the purpose of keeping a region/object of interest under constant surveillance by highlighting a bounding box drawn on its contours. We need to test our algorithm for three given datasets containing a car, human and a vase with the expected output that looks somewhat like follows –

## Procedure / Steps followed in this project:

1) The frames in each dataset is converted to the video featuring moving car on road, walking human on the pavement and a box on a table by using the '**videocreator.py**' program file.
2) The Lucas-Kanade template tracker is implemented as per the LK tracking algorithm described in the Section 2 of Simon and Matthew's paper.



**The Lucas-Kanade Algorithm**

```
Iterate:
    (1) Warp $I$ with $\mathbf{W}(\mathbf{x};\mathbf{p})$ to compute $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$
    (2) Compute the error image $T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))$
    (3) Warp the gradient $\nabla I$ with $\mathbf{W}(\mathbf{x};\mathbf{p})$
    (4) Evaluate the Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ at $(\mathbf{x};\mathbf{p})$
    (5) Compute the steepest descent images $\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$
    (6) Compute the Hessian matrix using Equation (11)
    (7) Compute $\sum_{\mathbf{x}} [\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}]^{\mathrm{T}} [T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]$
    (8) Compute $\Delta\mathbf{p}$ using Equation (10)
    (9) Update the parameters $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$
until $\|\Delta\mathbf{p}\| \leq \epsilon$
```

3) The Coordinates of the bounding box surrounding the object to be tracked is initialized.

4) The affineLKtracker function which was developed as core of the algorithm is given the inputs ‘current frame image, template frame image, previous bounding rectangle points and previous parameter values. It returns new parameter points and the new rectangle points. This “affineLKtracker function” iteratively computes the new warping parameter for each current frame until the constraint imposed on number of iterations/
5) The Tracker is implemented on three video sequences using the grayscale image of each frame.
6) The LK Tracker is made robust by employing the following strategies.

   - **Illumination** – The brightness of pixels in each frame is scaled to make the average brightness of pixels in tracked area to be same as the average brightness of the pixels in the template frame.
   - **M-Estimator** -- The weighted least square method is implemented using the minimization function along with the weighted Jacobian function

## AffineLKtracker Function/Algorithm

**Step 1:** The Image (current frame) is warped with the warp matrix that has been defined for affine transformations. The warp matrix is shown below –

$$\mathbf{W}(\mathbf{x};\mathbf{p}) = \begin{pmatrix} (1+p_1)\cdot x & + & p_3\cdot y & + & p_5 \\ p_2\cdot x & + & (1+p_4)\cdot y & + & p_6 \end{pmatrix} = \begin{pmatrix} 1+p_1 & p_3 & p_5 \\ p_2 & 1+p_4 & p_6 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

**Step 2:** The Error value is calculated by taking the difference between the template image and the warped image of the current frame. The template image is basically a cropped portion of the initial frame whose dimensions are that of the region of interest (which is a rectangle).

**Step 3:** The gradient of the Image is obtained using the Sobel function. Sobel operator is a joint Gaussian smoothing plus differentiation operation, so it is more resistant to noise. The x-direction and the y-direction derivatives are obtained individually as ‘Ix’ and ‘Iy’.

**Step 4:** The Jacobian (dw/dp) is calculated in this step which is shown below –

$$\frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \begin{pmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{pmatrix}.$$

**Step 5:** The steepest descent is calculated multiplying the gradient of the current image with jacobian computed in the previous step.

**Step 6:** The hessian matrix is computed which is given by the following equation –

$$H = \sum_{\mathbf{x}} \left[ \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^{\mathrm{T}} \left[ \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right].$$

**Step 7:** The steepest error is then obtained by multiplying the steepest descent with the error and then summed over x and y. It is shown below as follows –

$$\sum_{\mathbf{x}} \left[ \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^{\mathrm{T}} [T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]$$

**Step 8:** delta p is computed in this step by multiplying the inverted hessian matrix and steep error value obtained in the step 7 as follows –

$$\Delta \mathbf{p} = H^{-1} \sum_{\mathbf{x}} \left[ \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^{\mathrm{T}} [T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]$$

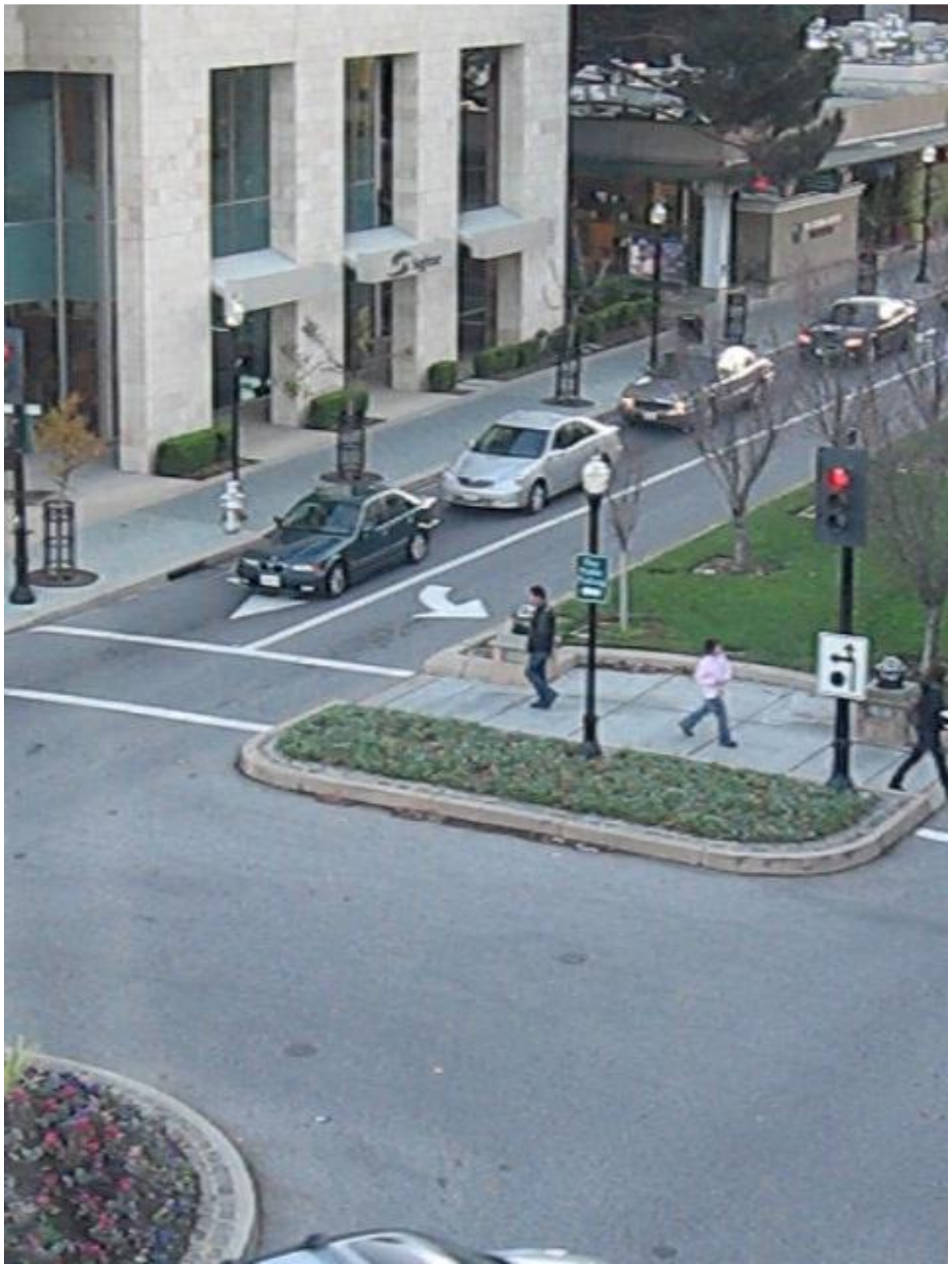
**Step 9:** The previous parameter values are updated in this step by adding the delta p value.

The function is iterated until it converges when the norm of delta p becomes smaller than the user defined threshold value (epsilon).

## Implementation of Object Tracking algorithm for Video sequence:

- The Video is obtained from the given frames dataset.
- The video is given as input to the python file “Frame_Creation”.  The initial frame of the video is captured and stored as the template file.

**Initial Car video frame**



**Initial Human video frame**

**Initial Vase video frame**

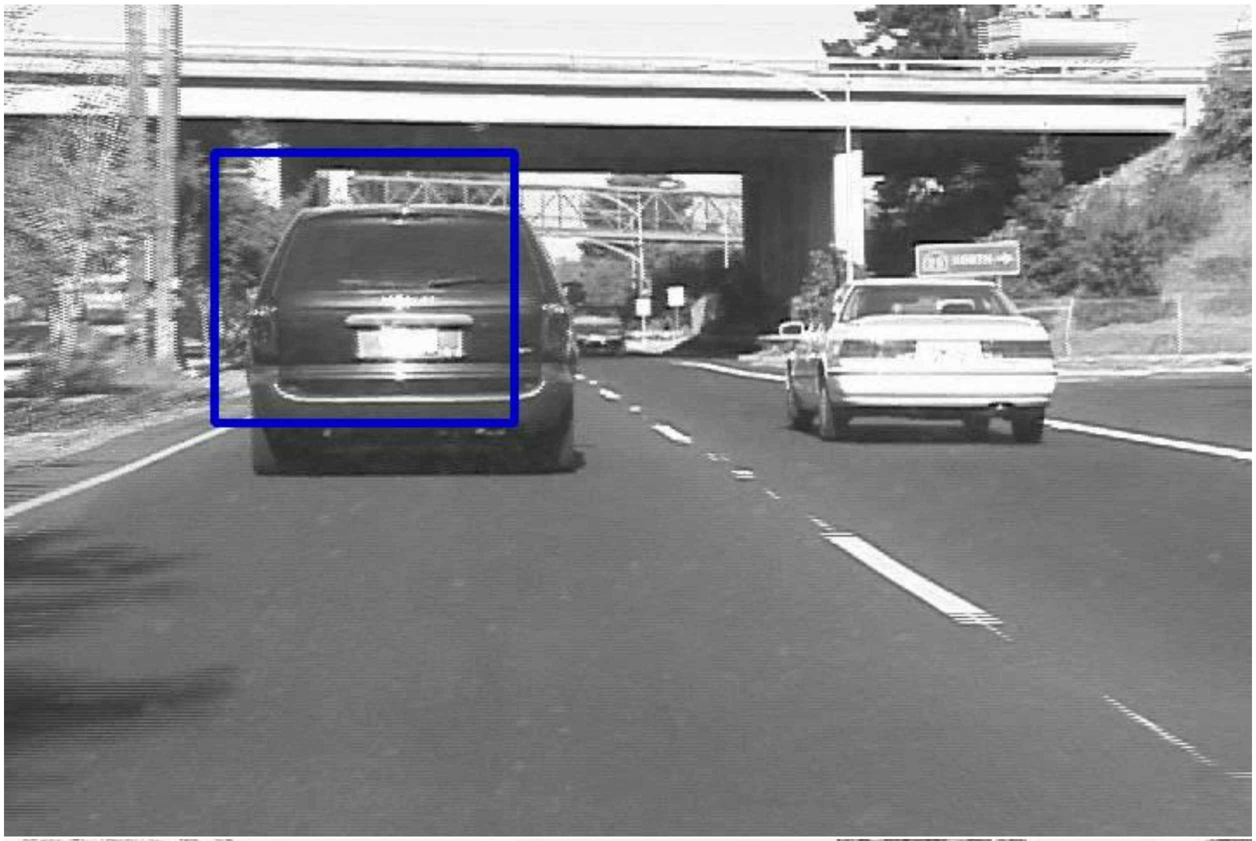
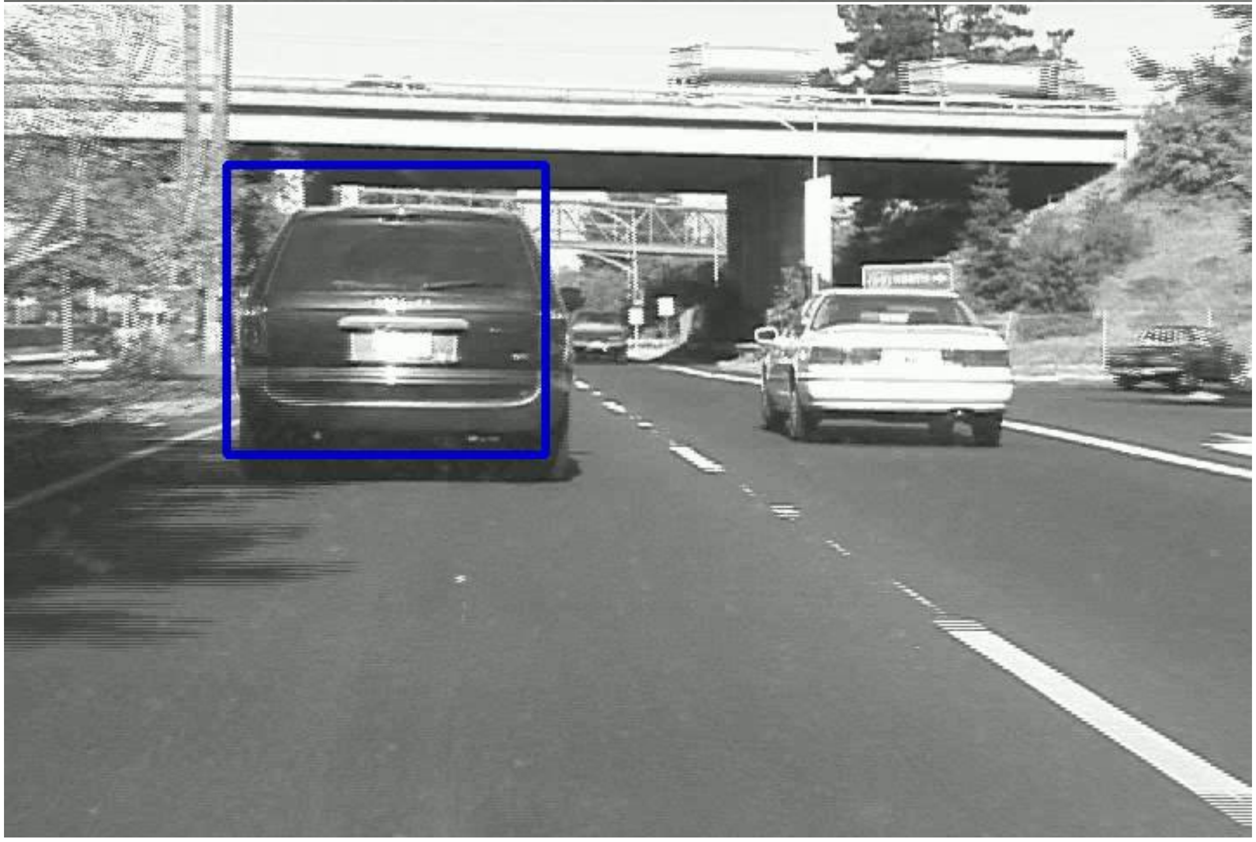
- The initial corner points of the rectangle that track the object in the video and the initial parameter value is initialized.
- Then the **affinLKtracker** function is called with Current Frame, Template frame, Previous rectangle points and Previous parameter points as the input arguments.
  The affineLKtracker function is explained in detail above.
- The new parameter and new rectangle co-ordinate points are obtained from the affineLKtracker function as the output after converging the iteration value in the function to get the new parameter values.
- The updated rectangle co-ordinate values are used to draw the tracking rectangle around the object in the video.

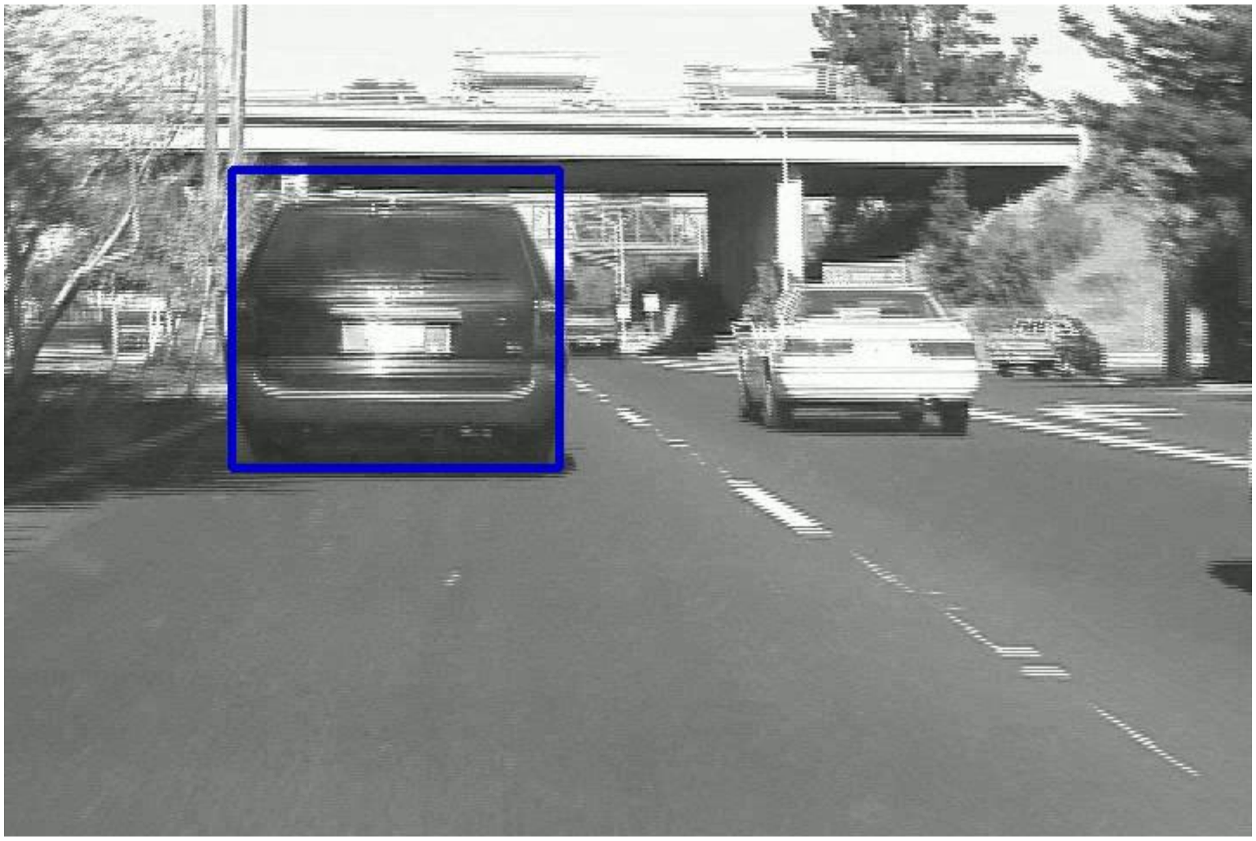
## Tracking_visualization :

### 1) Car Tracking :

The output video for car tracking is available in the following G-Drive link:

Some of the output tracking frames are mentioned below.


NORTH

NORTH

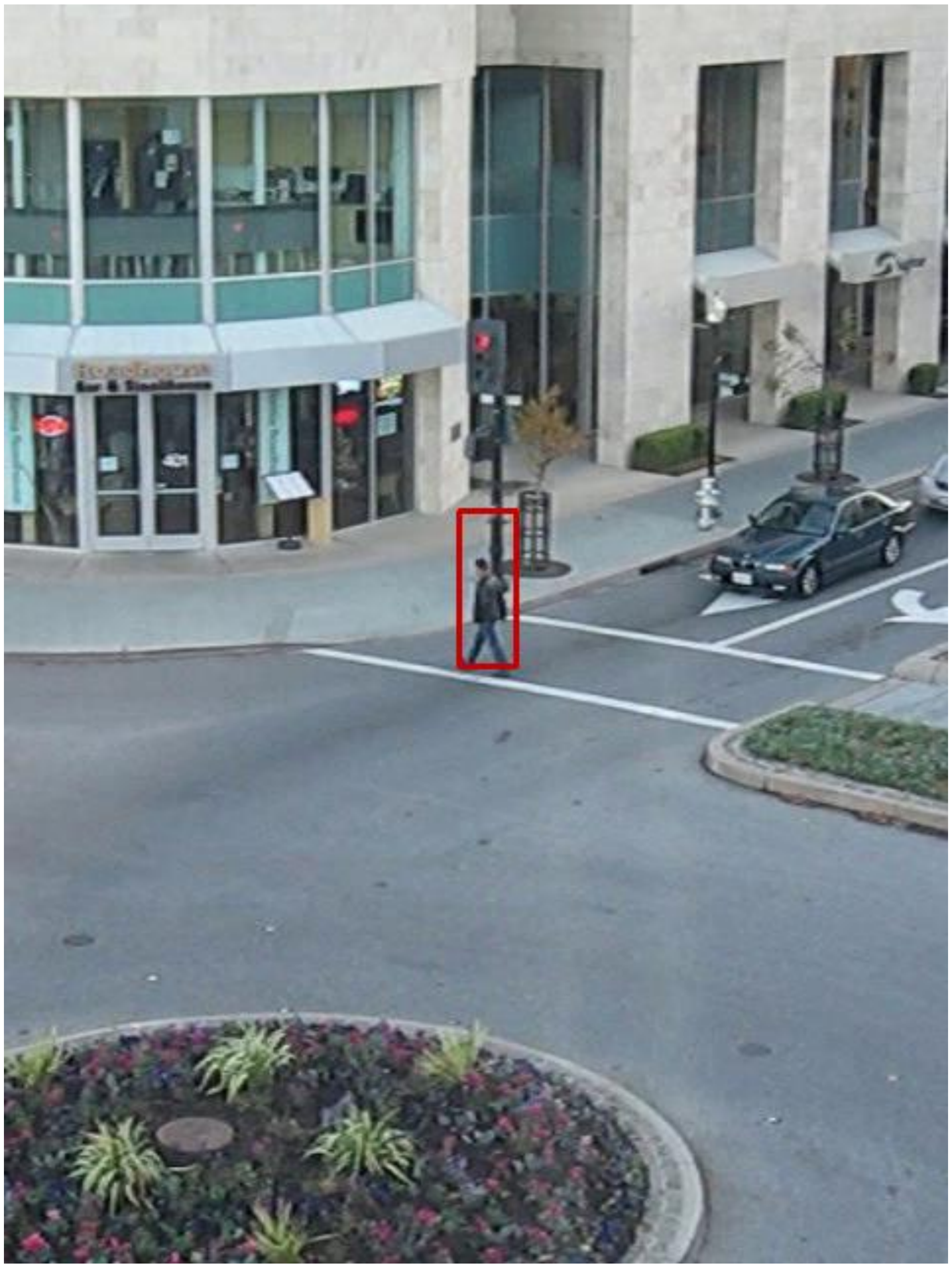


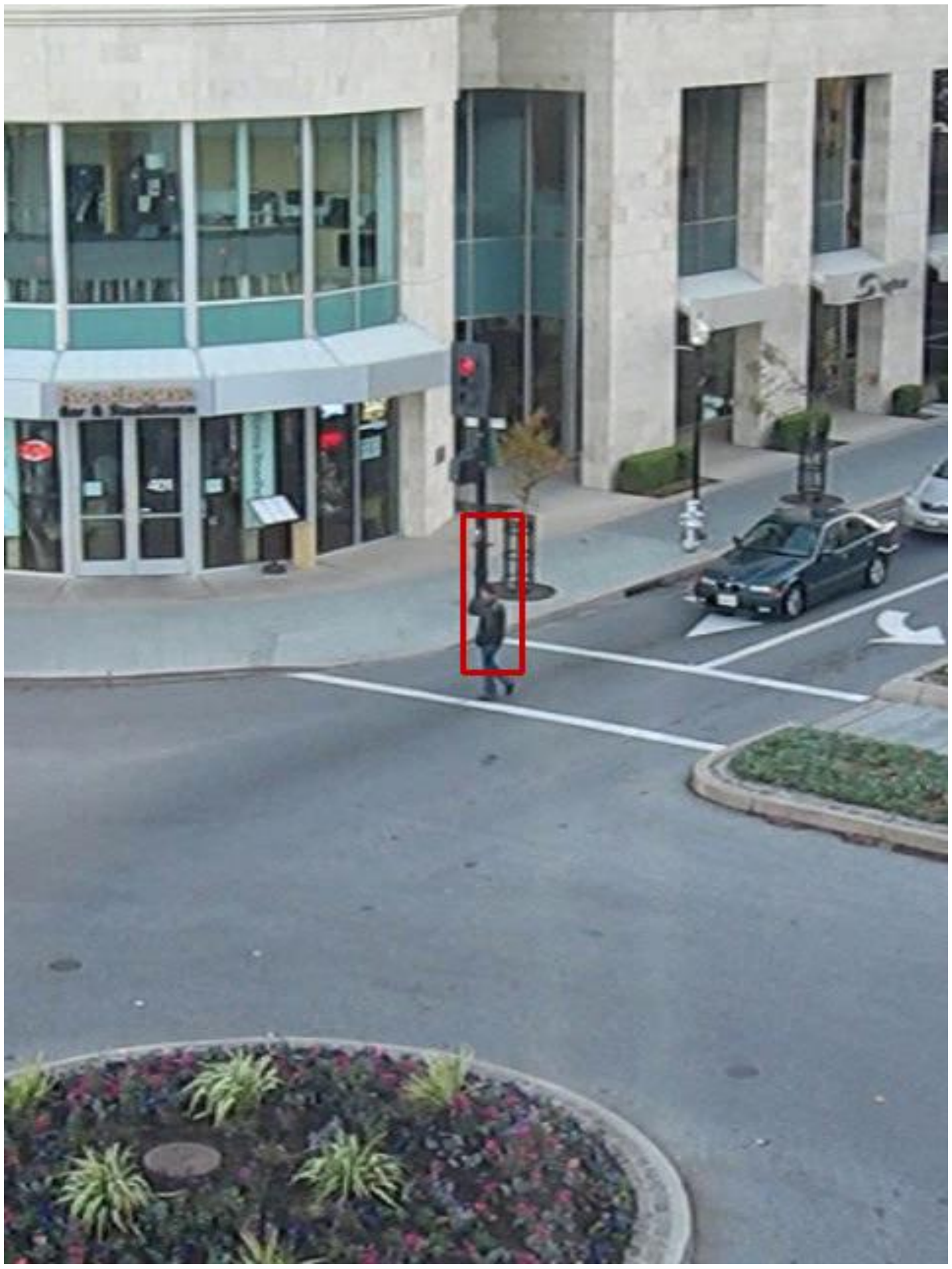
It observed that when the car goes under the bridge and moves to the rightmost lane nearly at the end of the video, the tracking rectangle deviates slightly from the object.

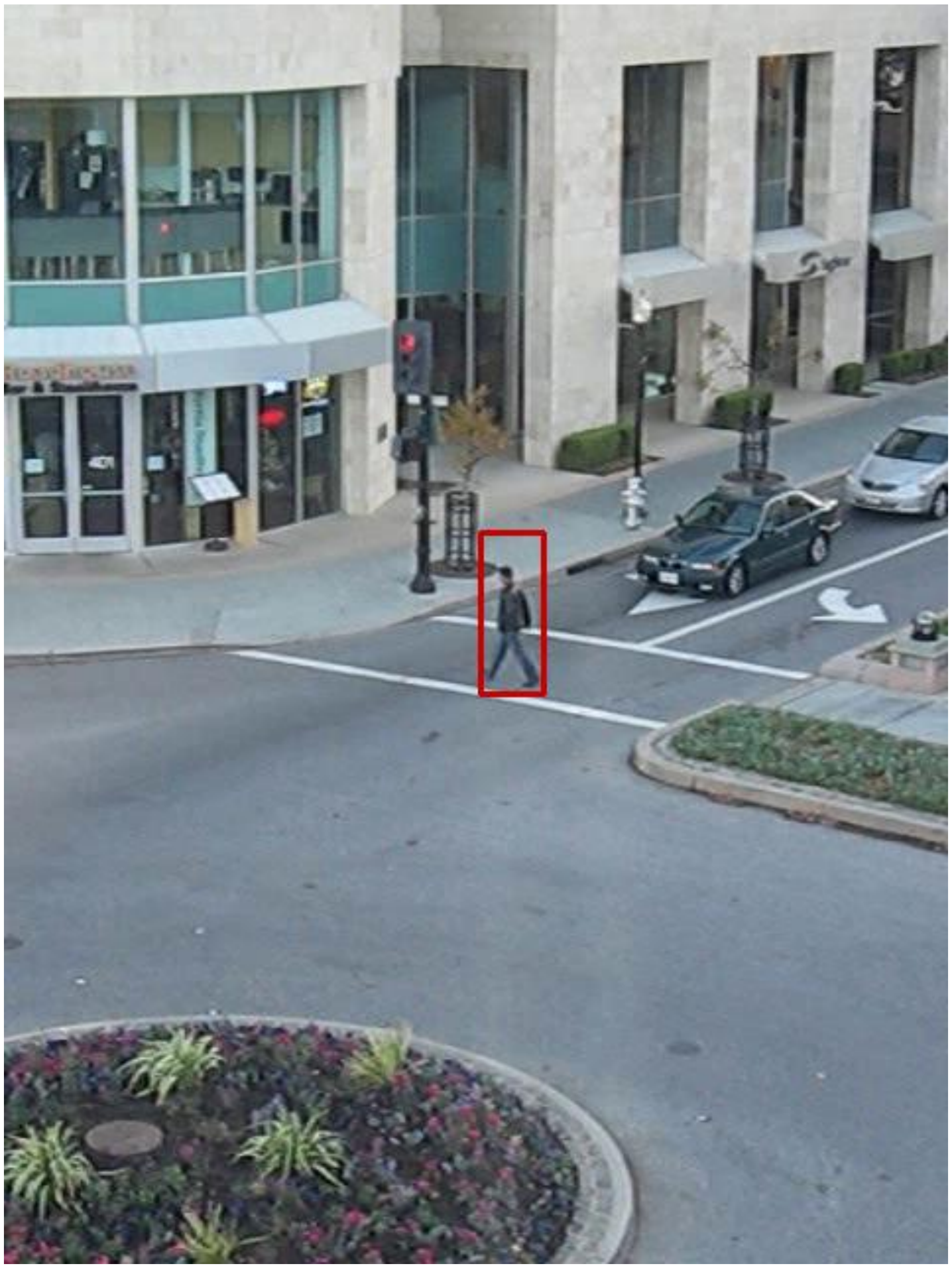
## 2) Human Tracking :

The output video for human tracking is available in the following G-Drive link:

Some of the output tracking frames are mentioned below.





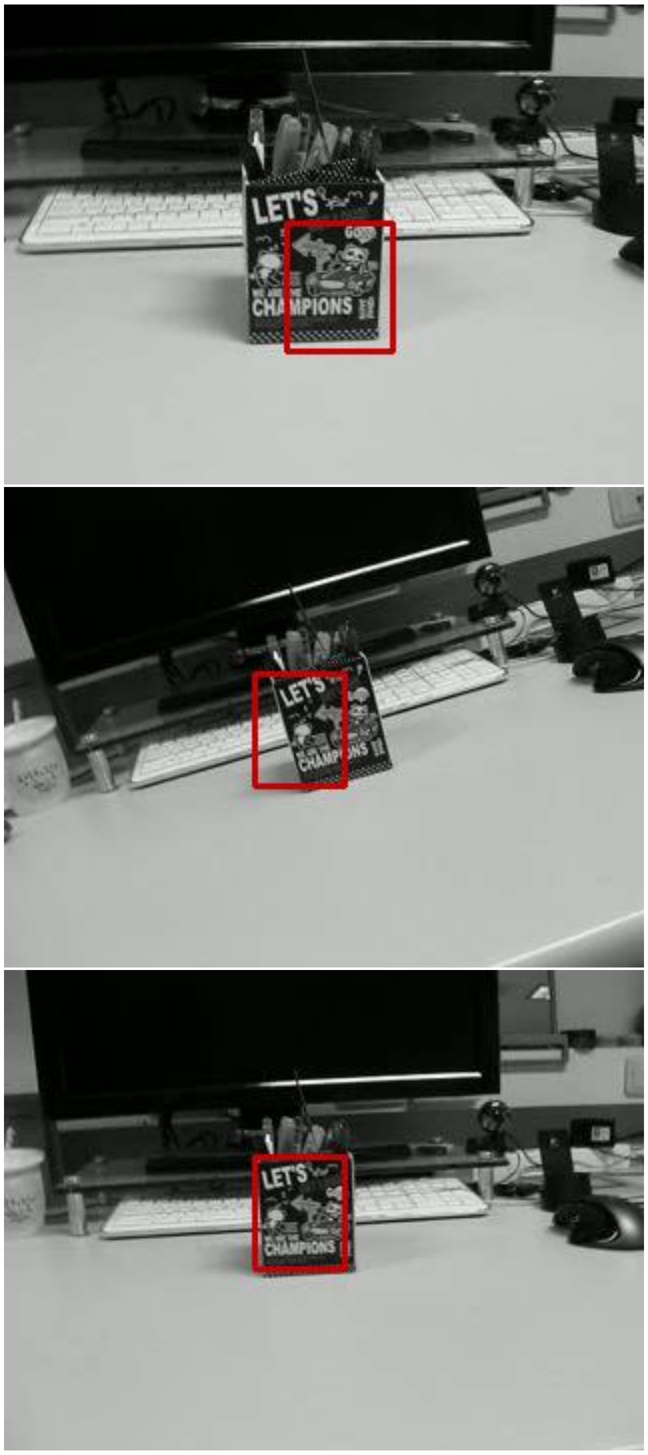


It observed that the tracking rectangle deviates slightly from the moving object at some point but the human is able to be tracked throughout the entire video frames.

## 3) Vase Tracking :

The output video for vase tracking is available in the following G-Drive link:

Some of the output tracking frames are mentioned below.



It observed that the tracking rectangle is not able to track the corner points of the vase but it could track the position of the object. As the video angle is keep rotating, the tracking rectangle is only able to track the position of points in each consecutive frames but the corner points of the box/vase on the table  is not obtainable to be tracked in consecutive frame hence the rotation of the object is not able to be tracked accurately.

# Robustness to Illumination

The algorithm is made robust to the changes in the illumination by implementing two of the following methods including method of scaling the brightness of each frames in the video and method of implementing the M-estimator.

## Method 1: Illumination of Video frames:

The brightness of pixels in each frame is scaled to obtain the average brightness of pixels in the tracked region to be the same as the average brightness of pixels in the template. The brightness of each frame is the video is increased by using the function 'increase_brightness'.

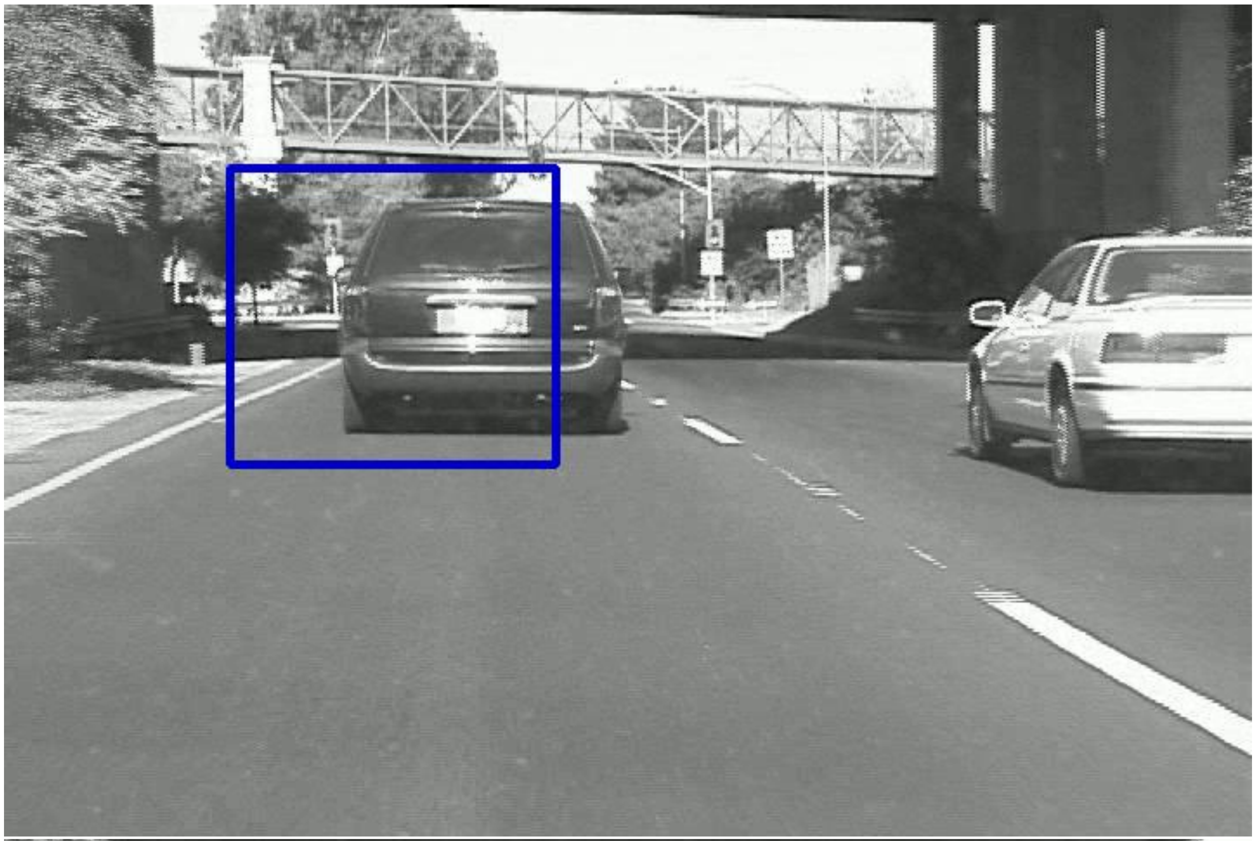
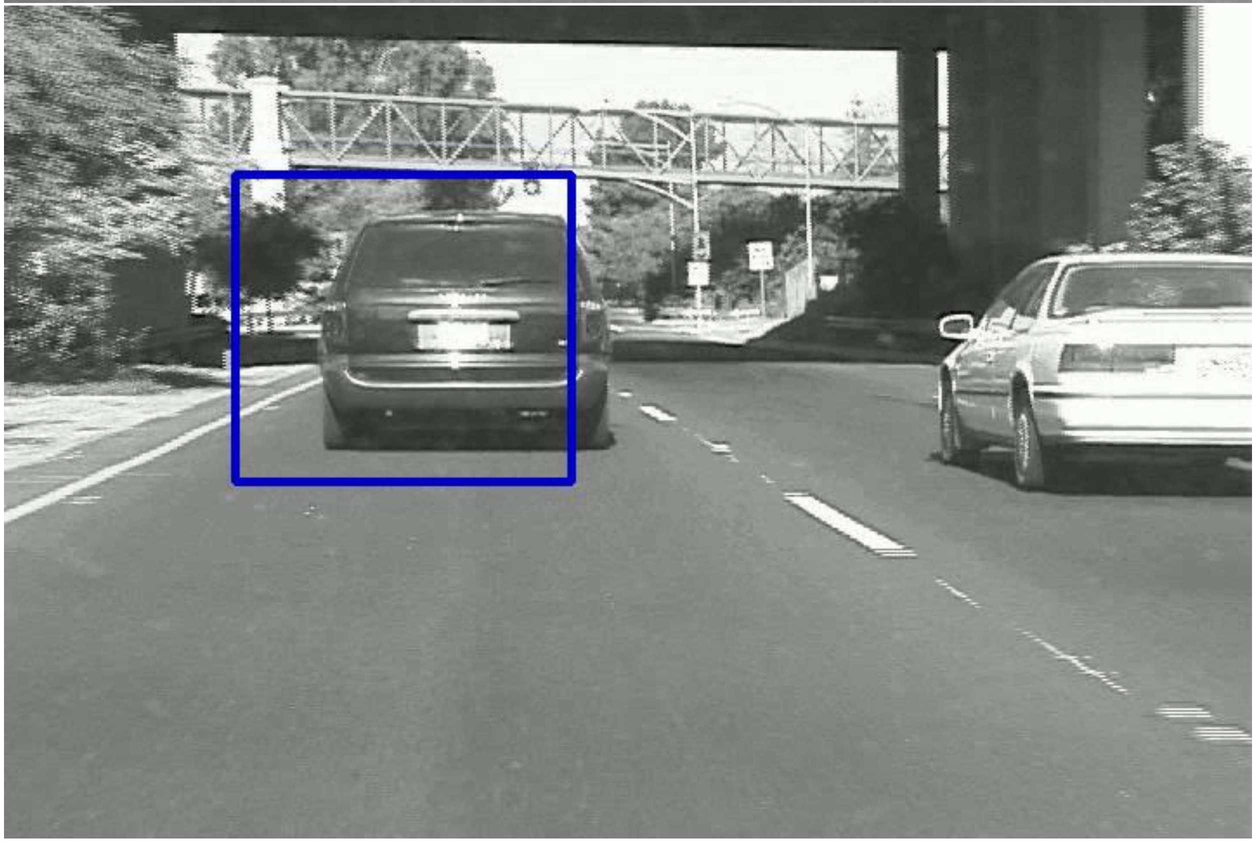
The brightness of the frame is increased by converting the frame initially to the 'hsv' format then the brightness value(v) of the 'hsv' image is increased and again converted back from 'hsv' to the 'bgr' image.

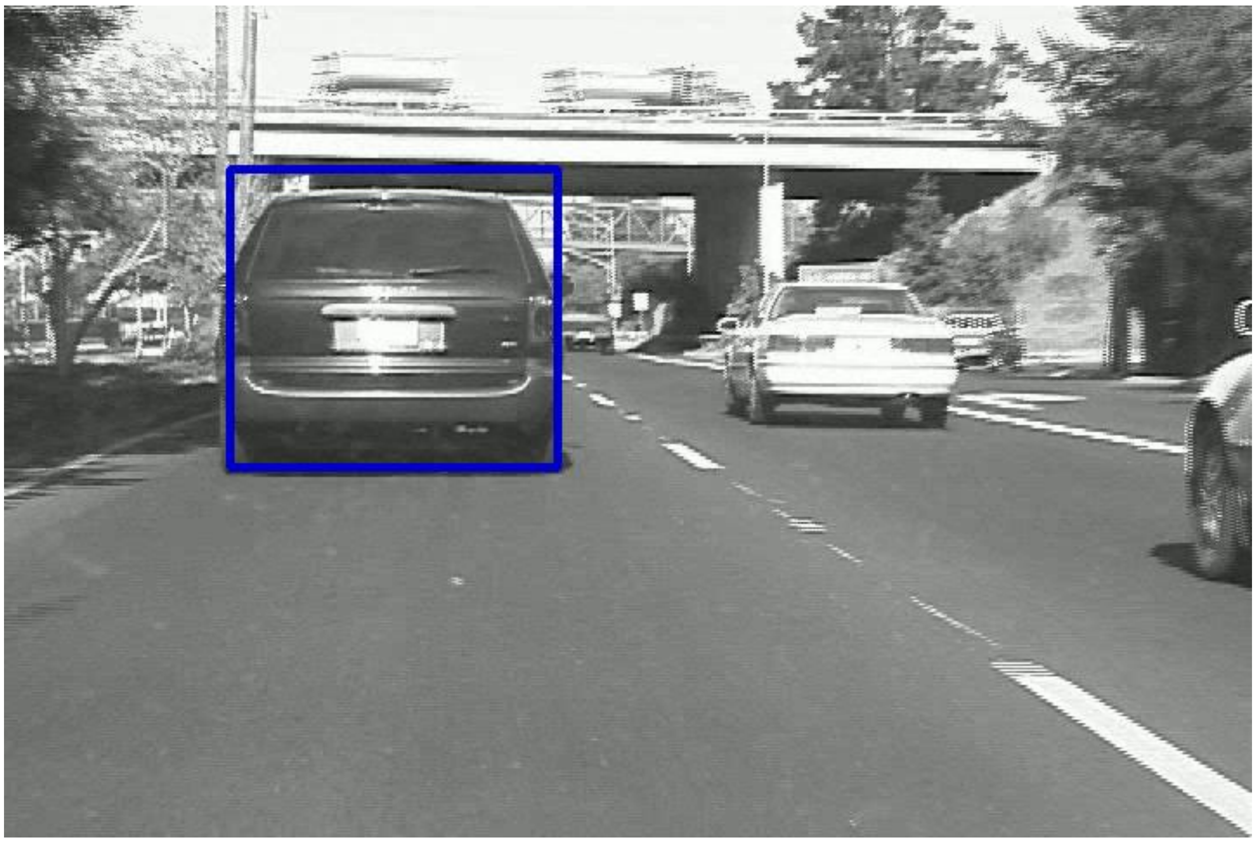
The same tracking algorithm is been followed again for tracking the object as mentioned above. The robustness of the algorithm in tracking the object is analyzed here.

### 1) Car Tracking :

The output video for car tracking is available in the following G-Drive link:

Some of the output tracking frames are mentioned below.

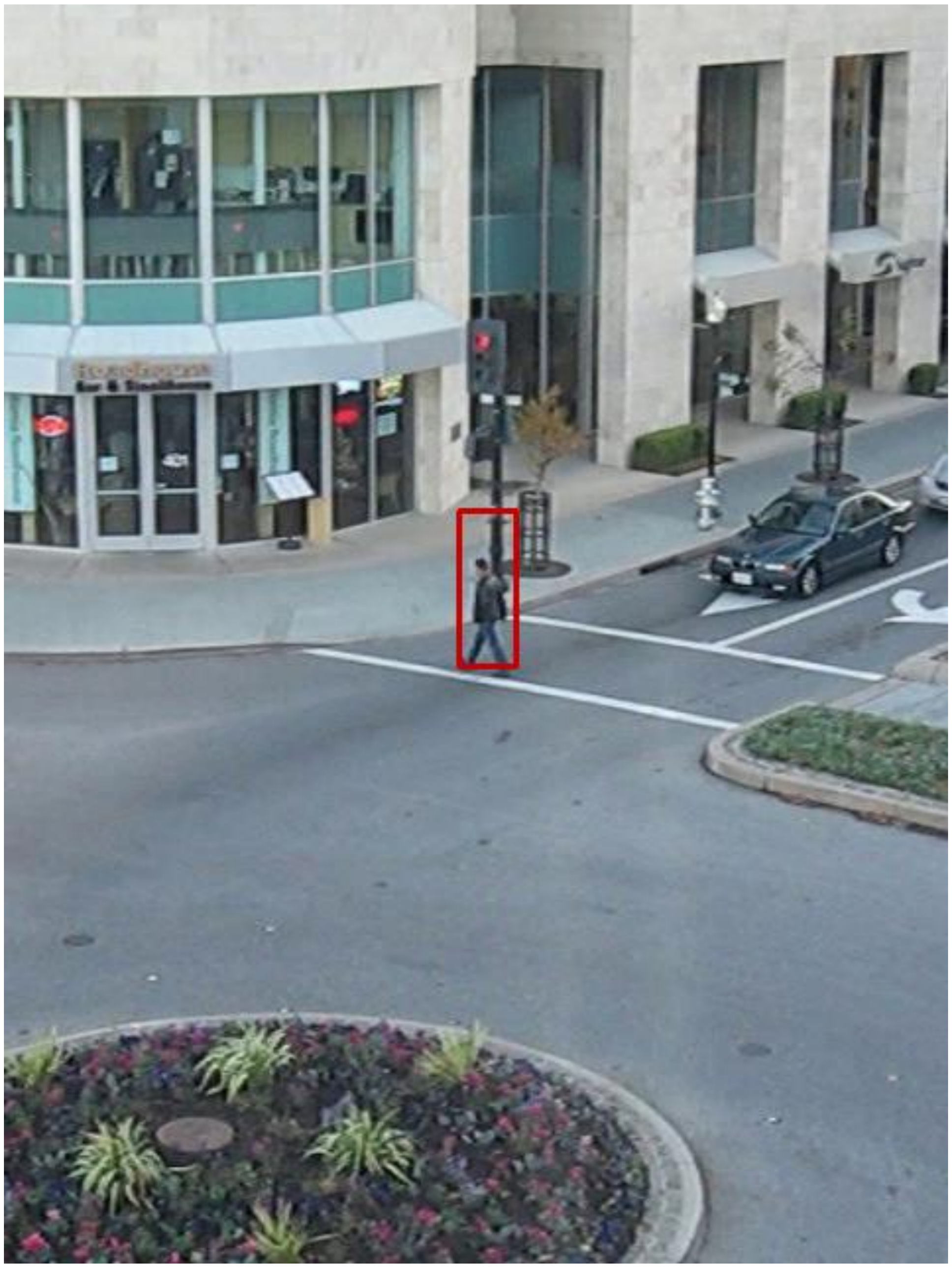


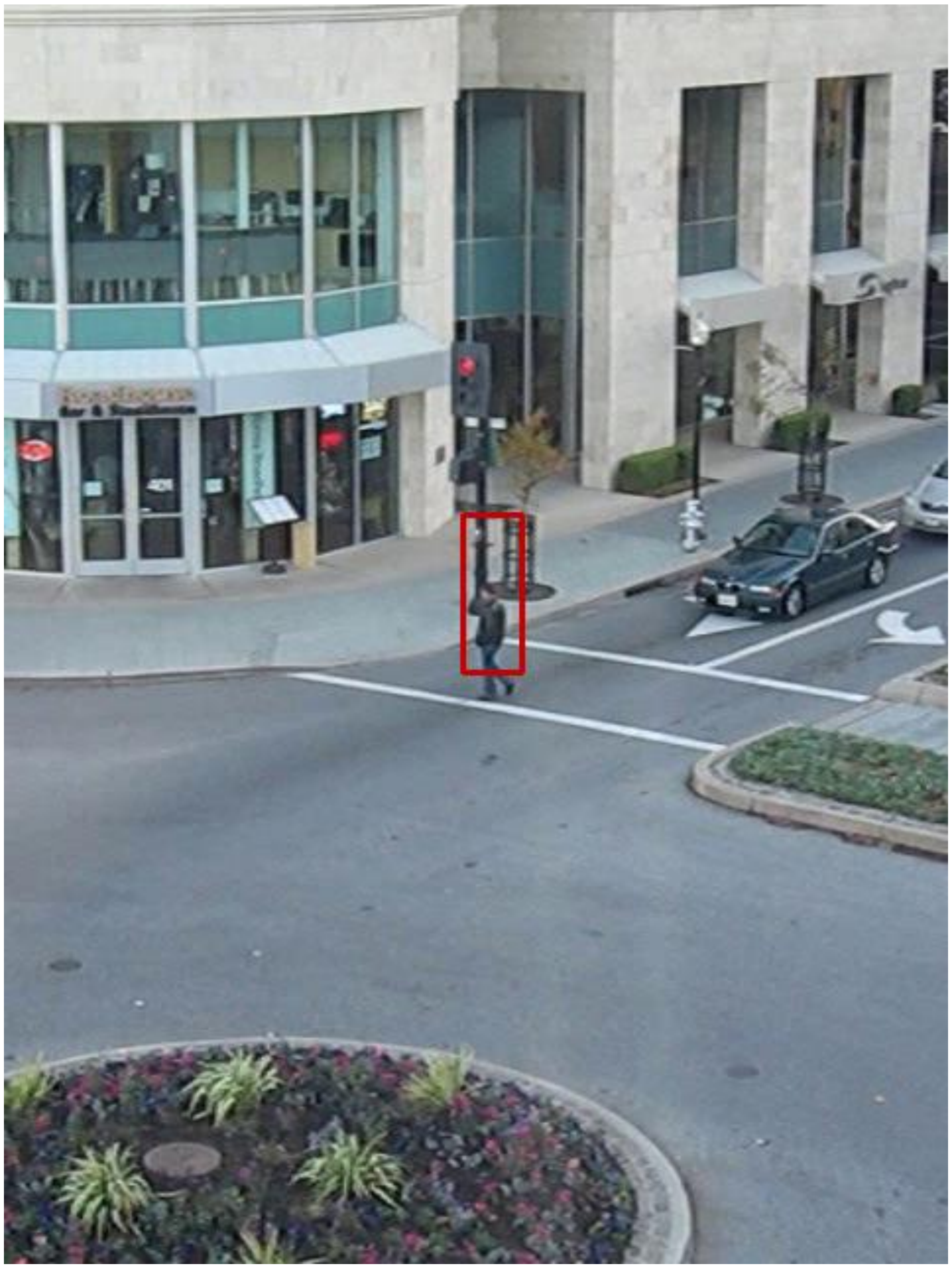
It observed that when the car goes under the bridge and moves to the rightmost lane nearly at the end of the video, the tracking rectangle deviates slightly from the object.

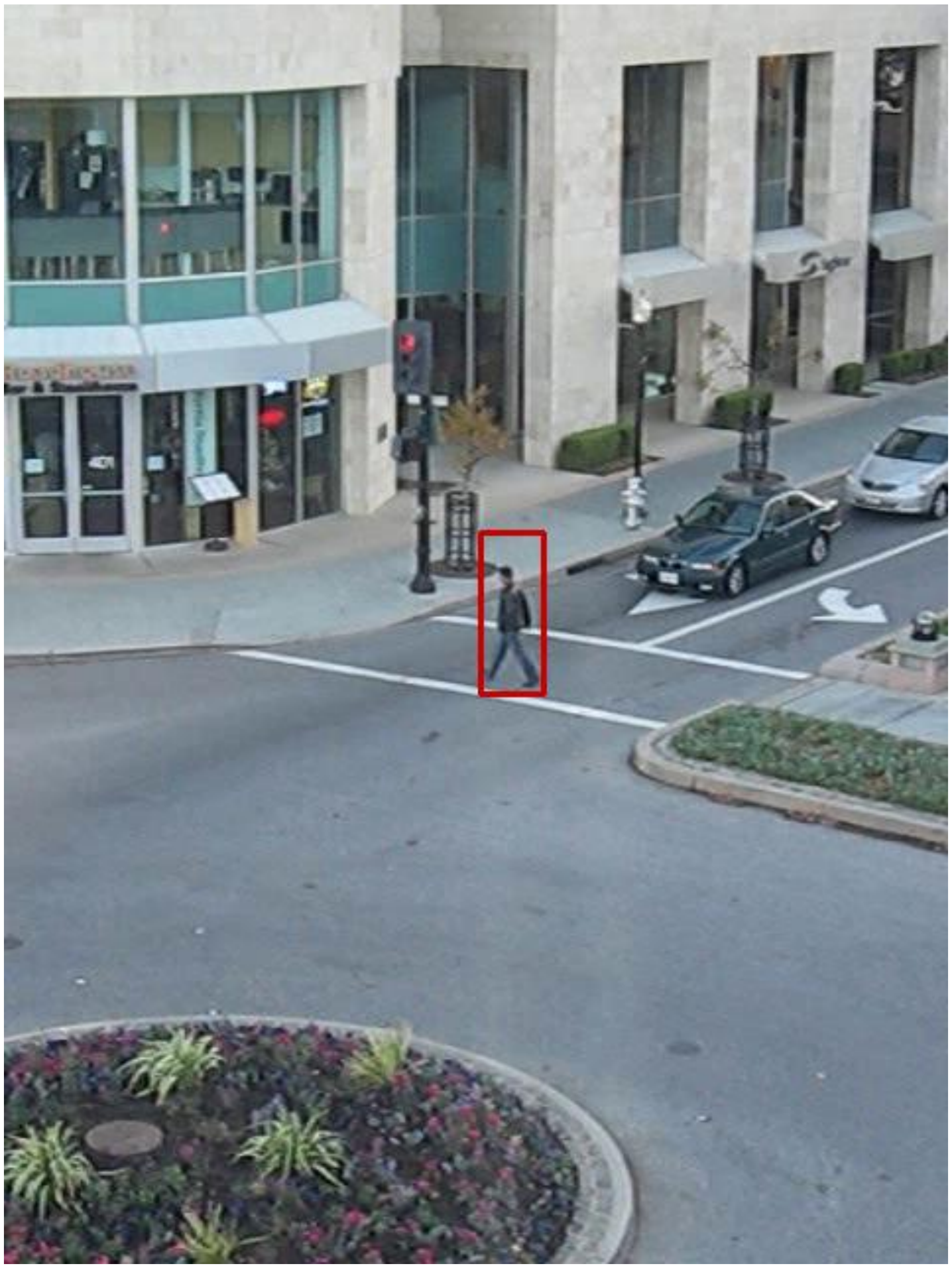
## 2) Human Tracking :

The output video for human tracking is available in the following G-Drive link:

Some of the output tracking frames are mentioned below.







It observed that the tracking rectangle deviates slightly from the moving object at some point but the human is able to be tracked throughout the entire video frames.

# Inference

**Normal Tracking:**

1) For car –
    a. The back of the given car was tracked accurately till the bridge came where the bounding box started to show signs of deviations and ultimately failed before the video sequence ended.
    b. We tried various methods like changing the threshold in turn changing the number of iterations, kernel size of sobel operator, tried different set of initial bounding rectangle points but only to find out that they failed to make a significant improvement in performance.
    c. The inference that can be derived from the output of car sequence is that the algorithm is failing to reject few outliers that adversely affects the cost minimization
    d. These outliers are points that do not lie in the bounds of the current frame. Thus can cause index errors in the program pertaining to the fact that they lie outside the range of the frames under consideration.
    e. These outliers can be minimized by increasing each pixel's brightness to some average brightness value or using a better cost minimization technique like M-estimators that employ more robust loss functions like Huber loss which can perform better than the least squares method which has been implemented in the initial stage.
2) For Human –
    a. We tried detecting the human for the whole video length. Even though we weren't successful for the whole video , we were able to detect the human halfway through the video.
    b. We pre-processed the images and frames using certain filters and morphological techniques , which improved the quality of the detection for the human.
    c. We tried increasing the brightness of the pixels , as suggested in the project description , and were able to improve the detection but only for a certain number of frames , and then the tracker breaks down again.
    d. We inferred that , with certain threshold values and proper set of points , we can detect the human with a greater accuracy.

3) For Vase-
    a. This was the most challenging part of the project , i.e. detection of the vase in the video.
    b. We tried several approaches , but we were not able to detect the rotation of the vase . There could be several reasons for that , including the speed of the rotation. The rotation happened at such a great pace that out algorithm failed for the rotation part.

c. We also tried using different shapes for the template , which would gave us better points for the vase , but couldn't successfully convert our idea into the code .

**Robust Tracking:**

1) In car video and Human video, the algorithm was able to track the car and human respectively for the whole video after scaling the brightness of each pixels in each consecutive frame. This shows that the developed algorithm is robust to track the objects in the respective frames.

## Templates:

1) **Fixed templates** are useful when object shapes do not change with respect to the viewing angle of the camera.

Image subtraction :
In this technique, the template position is determined from minimizing the distance function between the template and various positions in the image. Although image subtraction techniques require less computation time than the following correlation techniques, they perform well in restricted environments where imaging conditions, such as image intensity and viewing angles between the template and images containing this template are the same.

❖ Thus the tracking with the Fixed templates is better for the **Car** and **Human** Tracking videos

2) **Deformable template** matching approaches are more suitable for cases where objects vary due to rigid and non-rigid deformations. These variations can be caused by either the deformation of the object per se or just by different object pose relative to the camera. Because of the deformable nature of objects in most video, deformable models are more appealing in tracking tasks.
In this approach, a template is represented as a bitmap describing the characteristic contour/edges of an object shape.

❖ Thus this deformable template are good for tracking in the **vase** video

## G-Drive Output Links

The G-Drive Link for the all the output videos is provided below:

https://drive.google.com/open?id=1UeFDmZDvEt-vqc8R-YMI45GiDBMj6Gbk