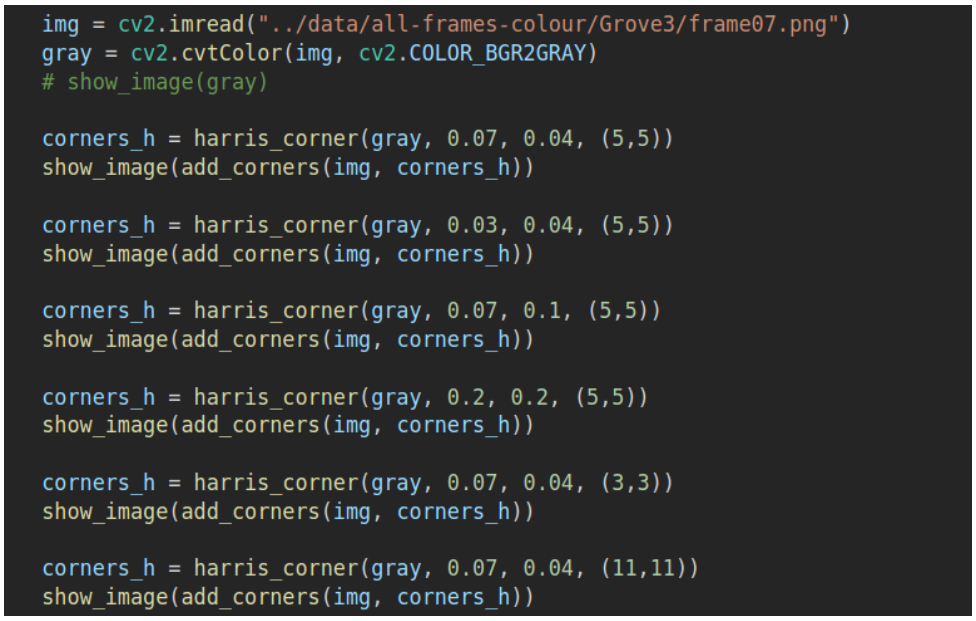
Computer Vision - Assignment 2 Report

Laksh Nanwani - 2021701002

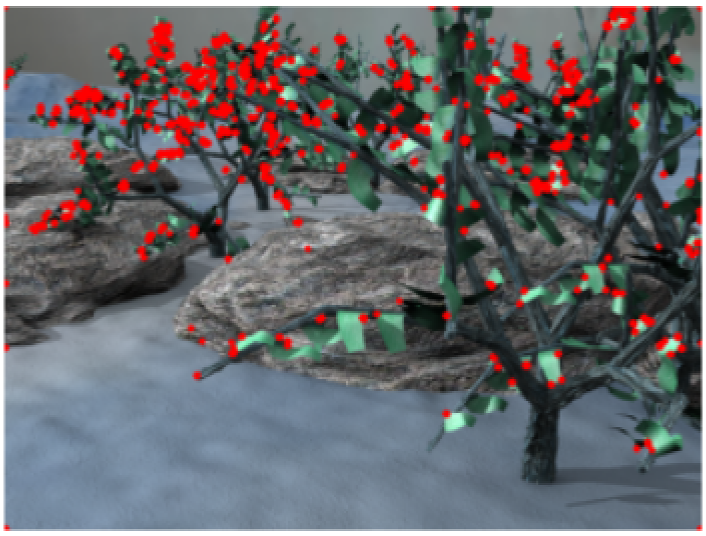
2.1.1 Harris corner detector

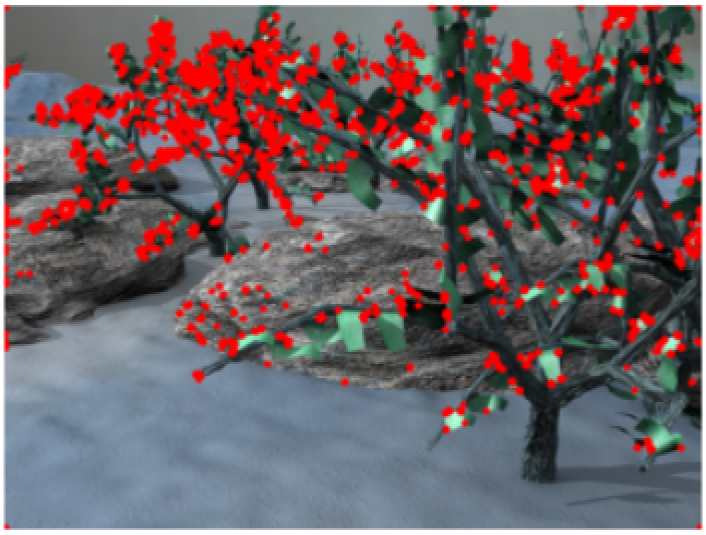
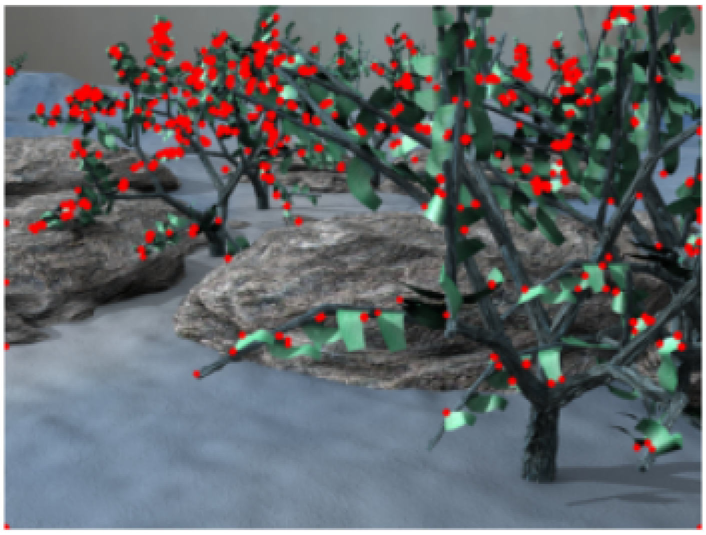
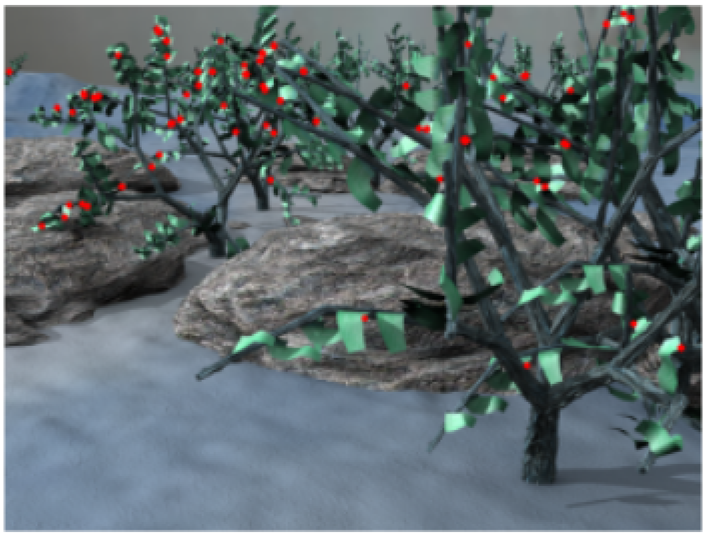
Harris method uses a matrix H which is obtained using image derivatives in x and y directions, and then the eigenvalues of this matrix are used to determine the interest points for detection corners. The function used to obtain the final corners is
f = det(H) - k * (trace(H)^2), where corners are detected using a threshold.
f > threshold gives the corners.

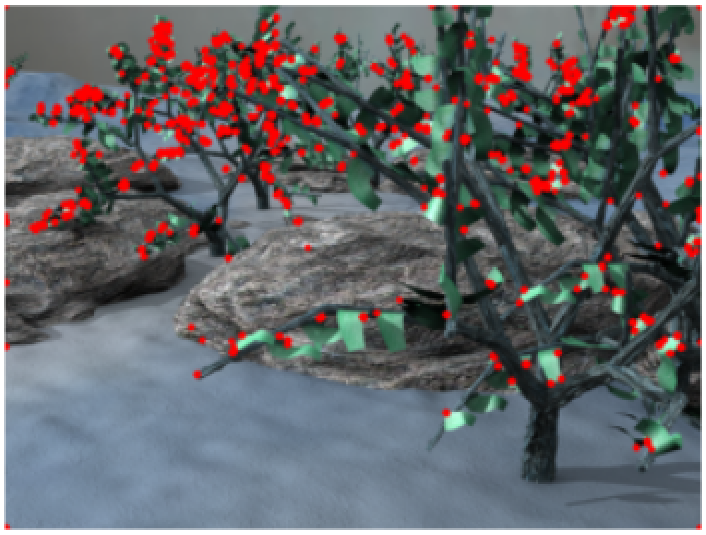
Experiments: Trying to find corners with different values of k and threshold.

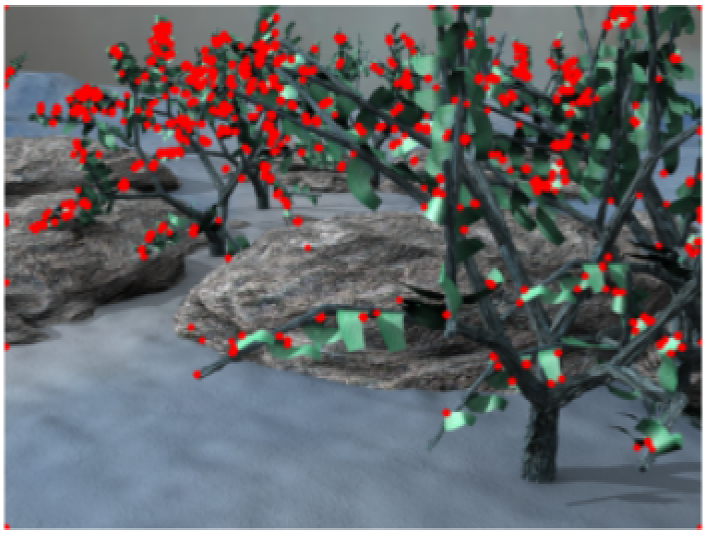


```python
img = cv2.imread("../data/all-frames-colour/Grove3/frame07.png")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# show_image(gray)

corners_h = harris_corner(gray, 0.07, 0.04, (5,5))
show_image(add_corners(img, corners_h))

corners_h = harris_corner(gray, 0.03, 0.04, (5,5))
show_image(add_corners(img, corners_h))

corners_h = harris_corner(gray, 0.07, 0.1, (5,5))
show_image(add_corners(img, corners_h))

corners_h = harris_corner(gray, 0.2, 0.2, (5,5))
show_image(add_corners(img, corners_h))

corners_h = harris_corner(gray, 0.07, 0.04, (3,3))
show_image(add_corners(img, corners_h))

corners_h = harris_corner(gray, 0.07, 0.04, (11,11))
show_image(add_corners(img, corners_h))
```







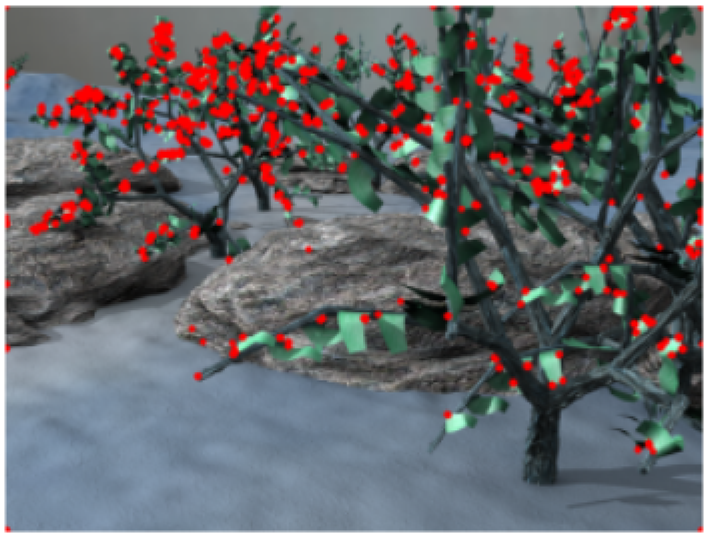
Observations: Increasing k gives more points
Increasing threshold gives lesser points

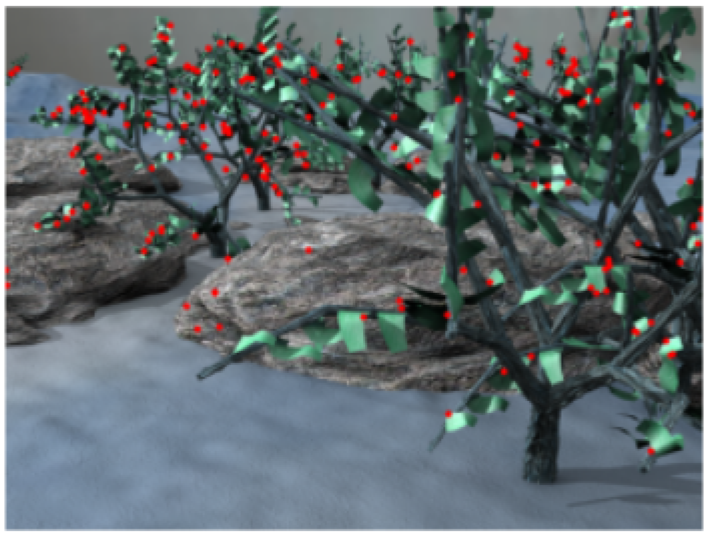
2.1.1 Shi tomasi corner detector

Shi tomasi method uses a similar matrix H which is obtained using image derivatives in x and y directions, and then the eigen values of this matrix is used to determine the interest points for detection corners. The function used to obtain the final corners is
f = min(eig1, eig2), where corners are detected using a threshold. f > threshold gives the corners.

Experiments: Trying to find corners with different values of threshold.

Observations:





Increasing threshold gives lesser points but with higher responses.

2.2 Forward-Additive Sparse Optical Flow

Lukas Kanade method assumes brightness constancy and also that the motion of the pixels in a small neighborhood is identical. It uses Taylor series expansion for linearisation:

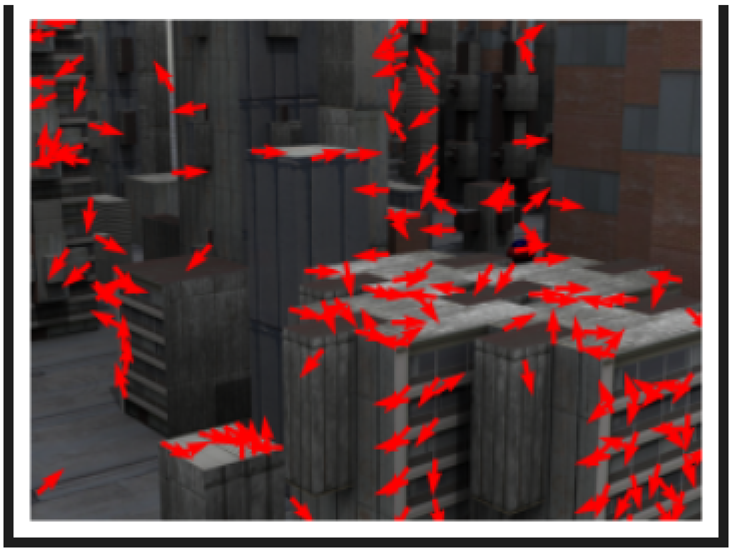
We get I(x + u, y + v, t1) − I(x, y, t) ≈ Ixu + Iyv + It

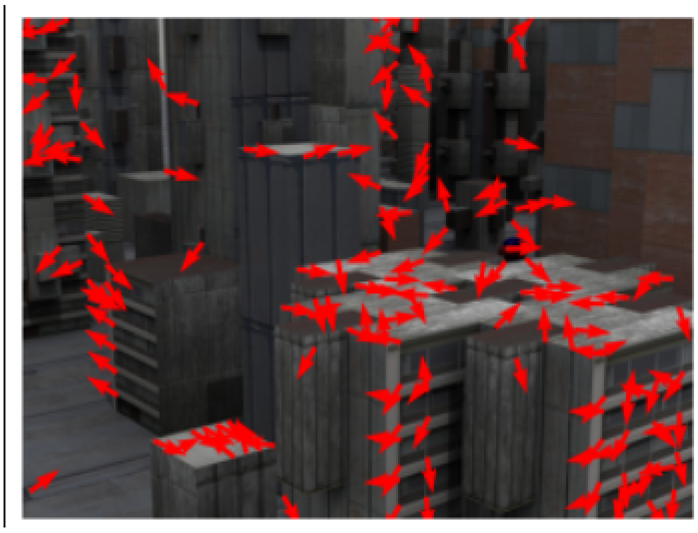
Assuming same shift of all pixels in a neighbourhood, we get least square problem as: $\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u & v \end{bmatrix} = \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$

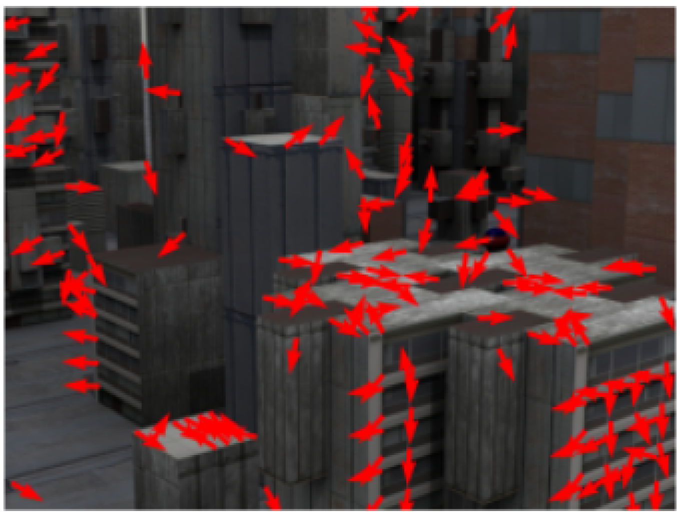
The eigen values should be very high for this method to work.

Experiments: Tried with different window sizes, (3,3), (5,5), (11,11)

Observations: The following flows are achieved.







Conclusion:

3. Multi-Scale Coarse-to-fine Optical Flow

We downsample the images to a very low level by 2(1 - num_levels). We then calculate the optical flow between the two images.
Shift the image at time t = 2
With this initial estimate and the image at time t = 1
Shifted window. Then upscale both the image and the optical flow.
2. This new optical flow acts as an estimate for the upsampled image. Continue this Step until you reach the final level. This method is useful for calculating flows.
A very big movement of the conventional Lukas Kanade optical flow method may fail.

Experiments: Tried with different number of levels

Observations: The following images are achieved.