Anirudh Topiwala (UID: 115192386)
ENPM 673, Robotics Perception
Homework: Optical Flow Estimation

1) Your algorithm will be validated on the Middlebury grayscale dataset, which you can find here. Each of these dataset items contains 8 frames of motion. Test your implementation of the Lucas Kanade algorithm on the Grove and Wooden sets. Make quiver plots for your optical flow computations. Store these frames as a video file called Grove_LK and Wooden_LK, respectively.
Upload the code, a representative quiver plot from each sequence and the videos.

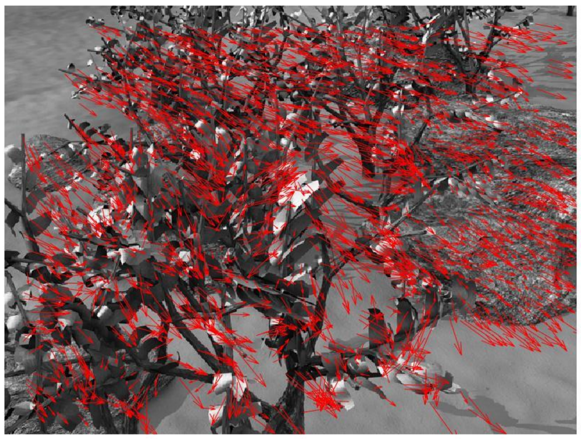
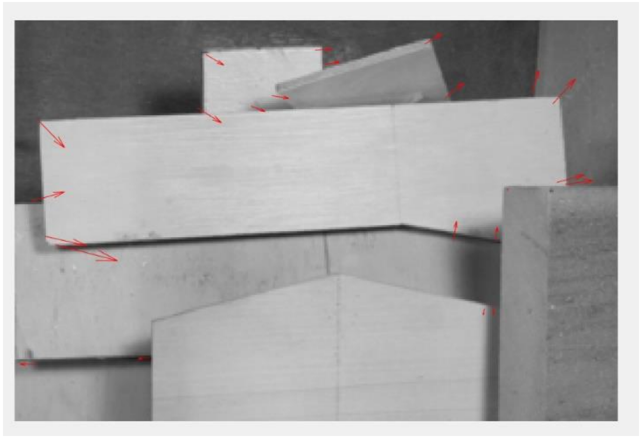
- Lucas Kanade method is a local method using the spatial coherence assumption. That is, the neighborhood pixels of a window sigma, all move coherently and share the same flow. From the dataset given and observations made, Lucas kanade works well for the Wooden Set, where the corners are easily identified, and textures are less as compared to the Grove set. Let's now see the implementation of the Lucas kanade Method.
- **Methodology:**
  - Consider an Image I(x,y). For a very small displacement, the new image can be represented by H(x,y) = I(x+u, y+v)
  - Here (u,v) represents the displacement of the pixel.
  - To get the Lucas Kanade equation, we will solve the following equation

$$\begin{bmatrix} \sum_{i=1}^{3} G_{i\sigma} I_{ix}{}^2 & \sum_{i=1}^{3} G_{i\sigma} I_{ix} I_{iy} \\ \sum_{i=1}^{3} G_{i\sigma} I_{ix} I_{iy} & \sum_{i=1}^{3} G_{i\sigma} I_{iy}{}^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} \sum_{i=1}^{3} G_{i\sigma} I_{ix} I_{it} \\ \sum_{i=1}^{3} G_{i\sigma} I_{iy} I_{it} \end{bmatrix}$$

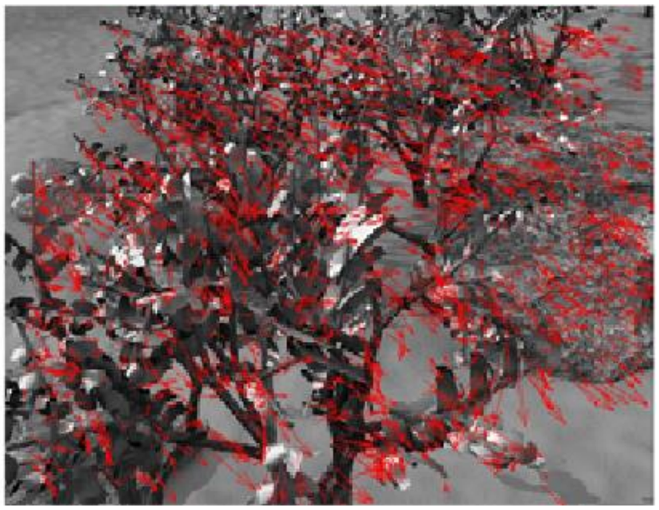
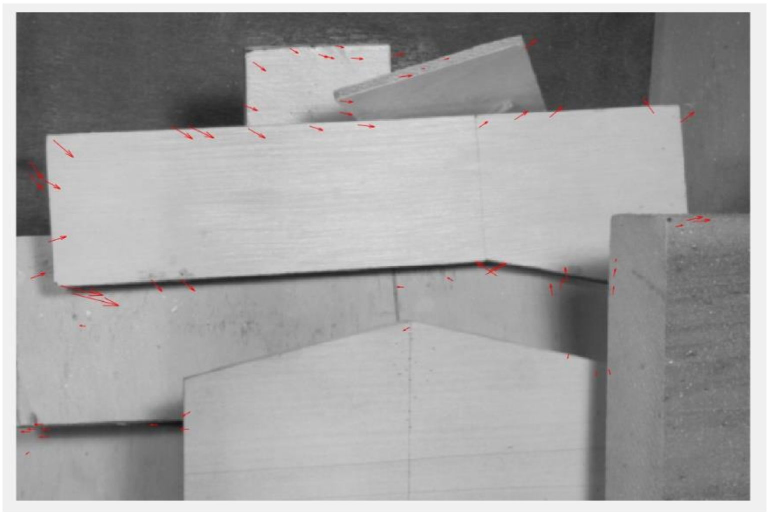
  - The solution exists when the system matrix is invertible. This happens when it has rank 2 since both eigenvalues are not zero and the system can be solved using Cramer's rule.

- I have implemented Lucas Kanade using three ways:
  1. Considering all the points of the Image.
  2. Just on the Corner points detected by Harris Corner Detection Algorithm (Matlab's Inbuilt)
  3. Using Mine Harris Corner Detection Algorithm.

- Let's now briefly observe the outputs from each method. (Note: The optical flow for the complete data set can be seen in the videos attached. Here I have only considered the first two frames.)
  1. **Applying Lucas kanade On Harris Corners Detected by Inbuilt MATLAB Function.**
     - Here we can see the optical flow vectors are just on the corner points detected.
     - We can see the direction of velocity of each corner point. This comparison is between the first two images.
     - One thing to note here is that, the corners on the extremities of the image must be removed as, the window won't fit otherwise. The window size taken here is w=5.

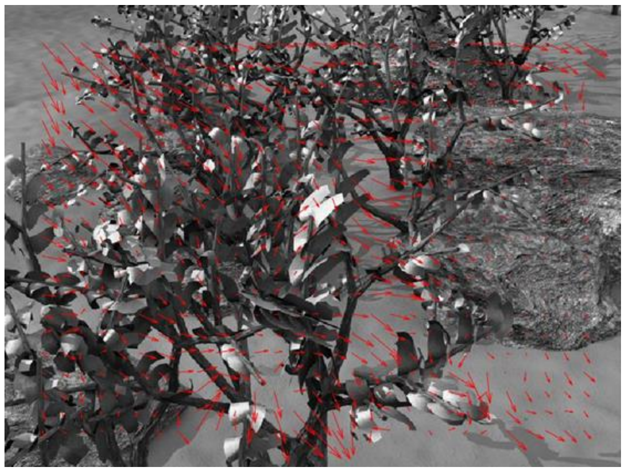
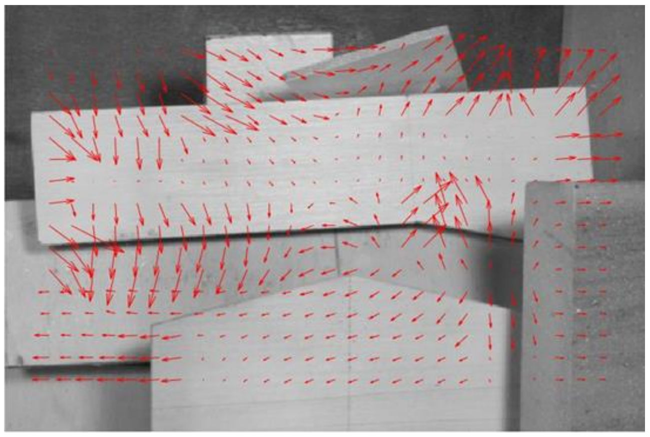



- The results are similar to the ones obtained by MATLAB's inbuilt function. The only difference is that there are more points or corners, as I have manually adjusted the threshold to identify more corners.

3. Applying Lucas Kande on all points of the image.

- In this approach, we form a window of size 'w' over all the points of the image. The important thing to note here is, there are no vectors formed in corners of the image as the corner pixels are discarded when we take a square window. Therefore, more the window size, more pixels will be discarded.

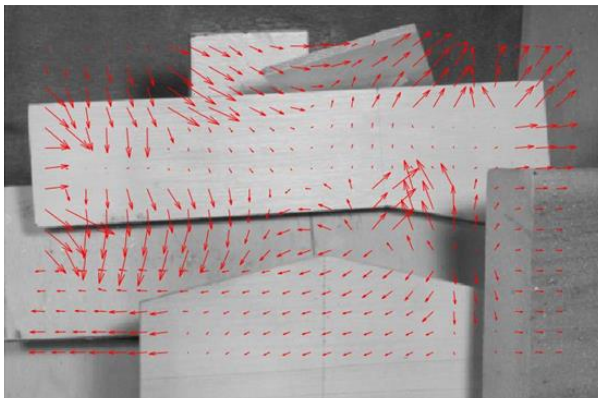
Comparison Between using the Whole Image and Using Just the Corners.

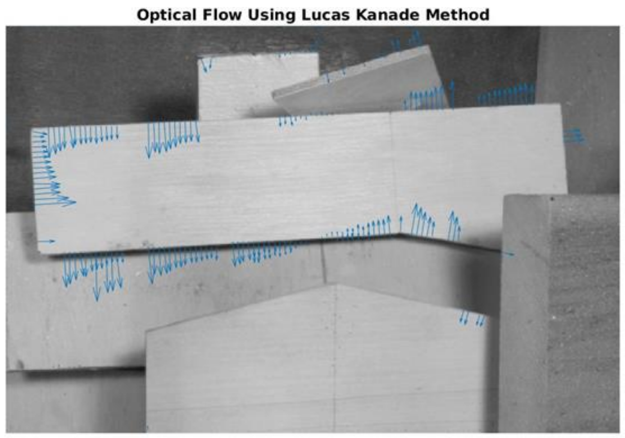
- Using all the points of the image give more uniformity and more accuracy compared to just taking the corners.
- Image with very few features will have very less corners as seen in the wooden image and therefore using all the points of the image gives better output in such cases.
- The time taken just using Harris corners is less than if we use all the points of the image and therefore useful if the image has more corners or more features and needs to be time efficient.

[Part 2] After this, you compare your output with inbuilt MATLAB implementations of Lucas-Kanade, Farneback and Horn-Schunck. Evaluate them on the same dataset mentioned in Part 1, and write a report comparing how the different methods behave in textured regions, non-textured regions, and at object boundaries.

Comparison Between User Define Lucas Kanade and MATLAB's Lucas Kanade

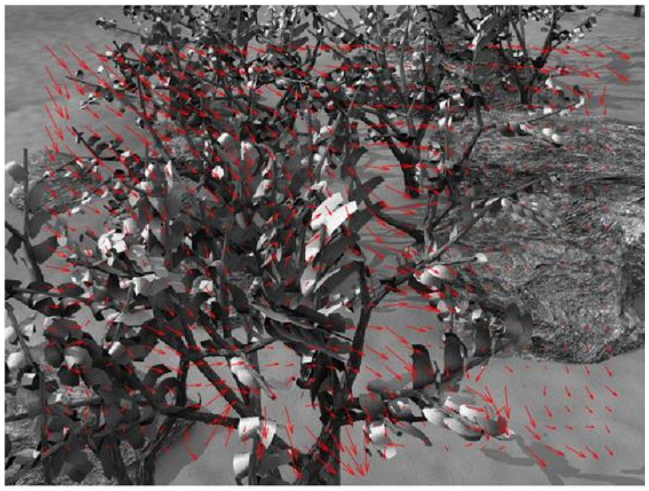
User Implementation of Lucas kanade on Wooden Set

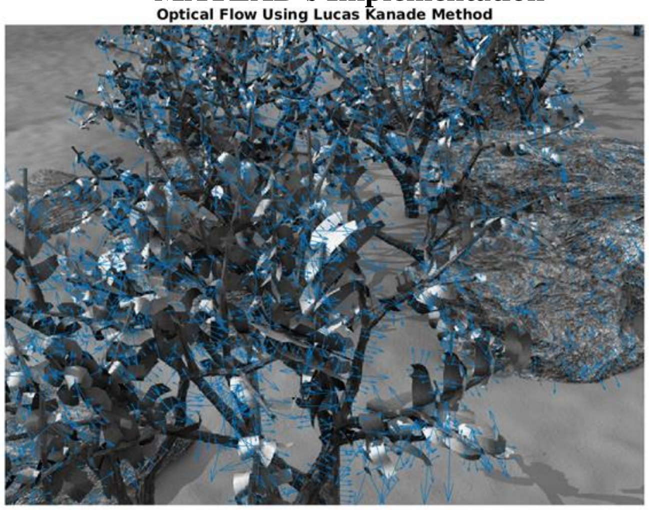




- We can see that the velocity vectors are accurately depicting the motion of the pixels and it is comparable to the output of MATLAB.
- The general observation is that in Lucas Kanade, the velocity vectors are generated near the edges and therefore we will see vectors with maximum magnitude near the edges.
- The increased number of velocity vectors in the left image is because I think there is a thresholding with size of the vector in MATLAB's output and therefore the velocity vectors in the middle part of the wooden pane and some other areas with less magnitude velocity vectors are discarded.
- The window size taken here is of 25, if we increase the window size we get more generalized output, or we get the flow of the surrounding region. With less window size, the output is more accurate, and it is more concentrated near the edges as seen in mahlabs output.

User Implementation of Lucas kanade on Grove Set **MATLAB's Implementation**



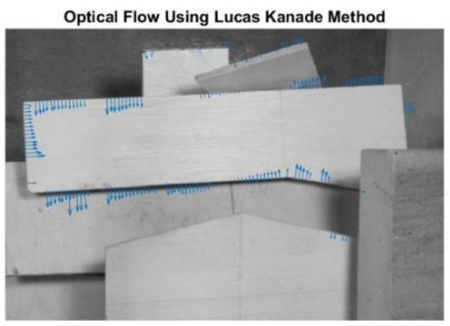


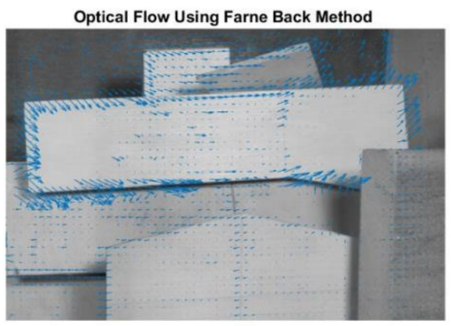
- For Grove data set, which is a more textured region we have multiple velocity vectors defining the motion of each edge or pixel. The observations are similar to that of wooden data set.

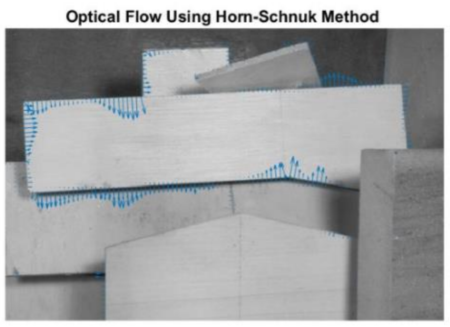
- The reason why my velocity vectors are more aligned than MATLAB's velocity vector is mainly because of the window size. As I have a bigger window size of 25, I have a more continuous output as compared to MATLAB's output.

Comparison Between the Three methods: Lucas Kanade, Horn-Schnuk and FarneBack.

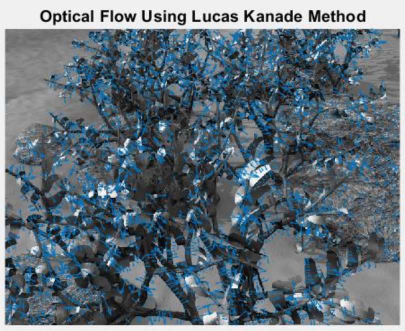
Wooden Data Set:

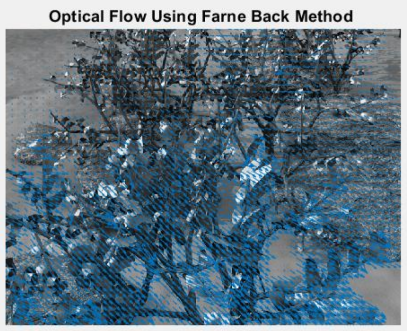


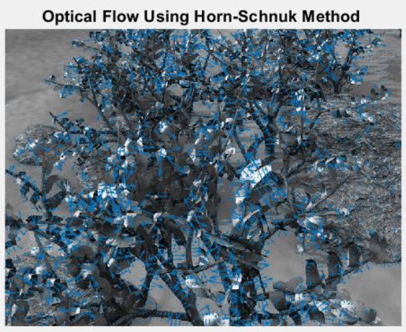




Grove Dataset:







- From, the above outputs we can see that, FarneBack method is the best for both textured and non-textured method.
- For all the methods, we can see that, the velocity vectors are only generated in area where edges are detected. Although, Farne Back method gives small velocity vectors even in non textured region.
- For the wooden image, Horn schnuk method gives slightly better results than Lucas kanade. This may be due to the reason that Horn Schnuk method does not have a window and therefore considers all the pixels at a time whereas, Lucas kanade only considers the pixels of a fixed window at a time. Horn schnuk algorithm also assumes smoothness of flow of the whole image. Thus, it tries to minimize distortions in flow and prefers solutions which show more smoothness.
- Another difference is that, the magnitude of velocity vectors is Horn Schnuk is very small as compared to other methods. It is because of rescaling we are able to visualize the output.
- For the Grove image, both Lucas Kanade and Horn Schnuk gives similar output whereas Farne Back output gives the best visualization of the flow.

References:

1) https://www.mathworks.com/help/vision/ref/opticalflowlk-class.html

2) https://www.mathworks.com/help/vision/ref/opticalflowfarneback-class.html

3) https://www.mathworks.com/help/vision/ref/opticalflowfarneback-class.html

4) https://www.mathworks.com/matlabcentral/fileexchange/

5) https://www.youtube.com/watch?v=5VyLAH8BhF8

6) http://vision.middlebury.edu/flow/floweval-ijcv2011.pdf