S. B. JAIN INSTITUTE OF TECHNOLOGY, MANAGEMENT & RESEARCH, NAGPUR.

Practical No. 10

Aim: Develop Lucas-Kanade method for optical flow detection by fing points to traverse over a moving object.

Name of Student: Shrutika Pradeep Bagdi

Roll No.: CS22130

Semester/Year: VII/IV

Academic Session: 2025-2026

Date of Performance:

Date of Submission:

AIM: Develop Lucas-Kanade method for optical flow detection by fing points to traverse over a moving object.

OBJECTIVE/EXPECTED LEARNING OUTCOME:

The objectives and expected learning outcome of this practical are:

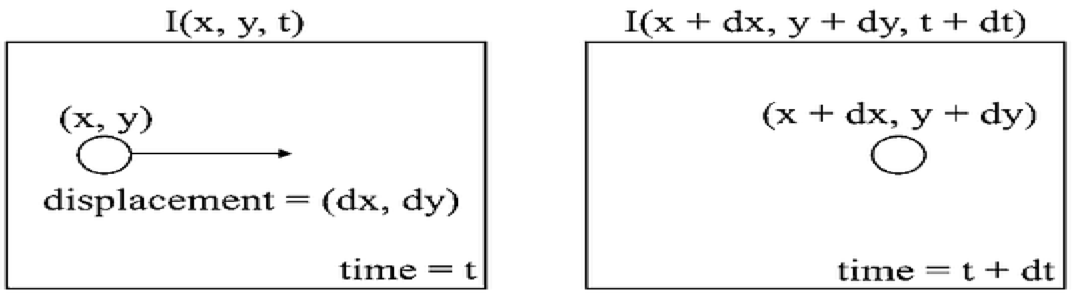
- To be able to understand principles behind the Lucas-Kanade optical flow algorithm
- To get knowledge about optical flow detection
- To be able to understand optical flow vectors to represent the direction and magnitude of object movement.

THEORY:

Object tracking is the ability of a computer or a system to follow and understand the movement of a single object as it navigates through the chaos of the world. This is crucial in many real-world scenarios as there are multiple objects and movements occurring simultaneously.

Why to Use Optical Flow?
Optical flow is based on the principle of analyzing motion in a video at a pixel-level. It computes the change in position of a pixel coordinates with respect to time to perceive the sense of direction and speed of the moving object.

1. **Detecting Movement:** Optical flow helps us detect movement in the video. It notices that pixels in the video are changing from one frame to the next, indicating that something is in motion.
2. **Quantifying Motion:** It not only identifies movement but also quantifies it. For instance, it can tell that the pixels at the front of the car are moving faster than those at the back, suggesting that the car is moving forward.
3. **Direction and Speed:** Optical flow provides information about the direction of movement. In our example, it reveals that the car is moving from left to right. Additionally, it can estimate the speed of the movement.
4. **Pixel-Level Understanding:** It works at a pixel level, meaning it's not just recognizing that the overall scene is changing, but it can pinpoint specific pixels (or points) and tell how they are moving.



Optical flow Description

Syntax: cv2.calcOpticalFlowPyrLK(prevImg, nextImg, prevPts, nextPts[, winSize[, maxLevel[, criteria]]])

Parameters:

prevImg – first 8-bit input image

nextImg – second input image

prevPts – vector of 2D points for which the flow needs to be found.

winSize – size of the search window at each pyramid level.

maxLevel – 0-based maximal pyramid level number; if set to 0, pyramids are not used (single level), if set to 1, two levels are used, and so on.

criteria – parameter, specifying the termination criteria of the iterative search algorithm.

Return:

nextPts – output vector of 2D points (with single-precision floating-point coordinates) containing the calculated new positions of input features in the second image; when OPTFLOW_USE_INITIAL_FLOW flag is passed, the vector must have the same size as in the input.

status – output status vector (of unsigned chars); each element of the vector is set to 1 if the flow for the corresponding features has been found, otherwise, it is set to 0.

err – output vector of errors; each element of the vector is set to an error for the corresponding feature, type of the error measure can be set in flags parameter; if the flow wasn't found then the error is not defined (use the status parameter to find such cases).

- **Lucas-Kanade Algorithm**

The Lucas-Kanade Method stands out as one of the widely used approaches for calculating Optical Flow. It works on two basic assumptions to calculate the values of u and v in the Optical Flow equation:

1. Flow is smooth locally. It means that the motion of objects or points in a small, localized region of an image is relatively consistent and does not exhibit sudden or abrupt changes.
2. Neighboring pixels have the same displacement. In other words, neighboring pixels in the selected region are assumed to have similar motion, allowing the algorithm to represent the overall motion with a single vector.

Let's assume that we have a 5*5 image patch which gives us 25 Optical Flow equations corresponding to 25 pixels in the image. These equations can be represented in Matrix Form:

Press enter or click to view image in full size

$$\underset{25\times 2}{A}\;\begin{bmatrix} I_x(\boldsymbol{p}_1) & I_y(\boldsymbol{p}_1) \\ I_x(\boldsymbol{p}_2) & I_y(\boldsymbol{p}_2) \\ \vdots & \vdots \\ I_x(\boldsymbol{p}_{25}) & I_y(\boldsymbol{p}_{25}) \end{bmatrix} \underset{2\times 1}{x}\;\begin{bmatrix} u \\ v \end{bmatrix} = - \underset{25\times 1}{b}\;\begin{bmatrix} I_t(\boldsymbol{p}_1) \\ I_t(\boldsymbol{p}_2) \\ \vdots \\ I_t(\boldsymbol{p}_{25}) \end{bmatrix}$$

Matrix Representation of 5*5 Image

$$\overset{A^\top A}{\begin{bmatrix} \sum\limits_{p\in P} I_x I_x & \sum\limits_{p\in P} I_x I_y \\ \sum\limits_{p\in P} I_y I_x & \sum\limits_{p\in P} I_y I_y \end{bmatrix}} \overset{\hat{x}}{\begin{bmatrix} u \\ v \end{bmatrix}} = - \overset{A^\top b}{\begin{bmatrix} \sum\limits_{p\in P} I_x I_t \\ \sum\limits_{p\in P} I_y I_t \end{bmatrix}}$$

Taking transpose of A on both sides we get

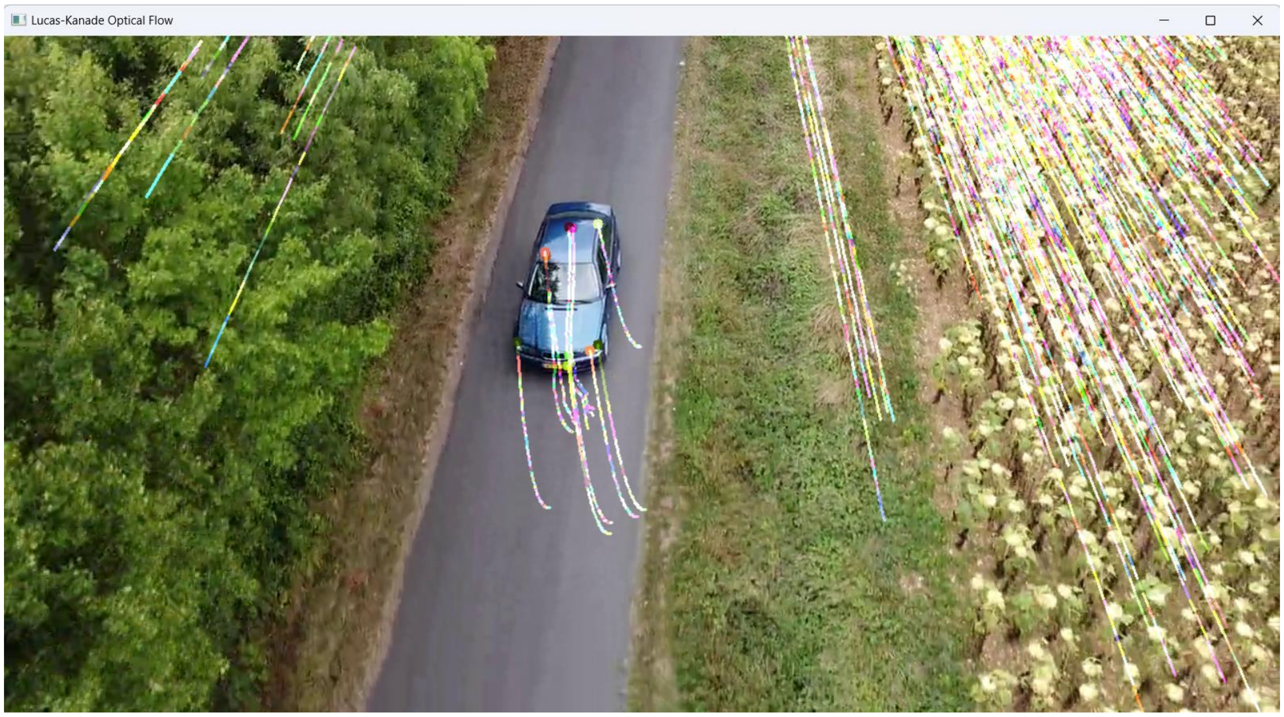
Finally, taking inverse on both sides we can compute the vector x that contains the unknowns u and v.

$$x = (A^\top A)^{-1} A^\top b$$

Algorithm:

Code:

INPUT & OUTPUT:



CONCLUSION:

The implementation of the Lucas-Kanade method for optical flow detection successfully enables tracking of motion by analyzing feature points across sequential image frames

DISCUSSION QUESTIONS:

1) What Lucas–Kanade does?
2) What is the Lucas–Kanade Method?
3) What is the the Optical Flow Equation?
4) How to Choose Points to Traverse a Moving Object?

REFERENCES:

- Computer Vision: Algorithms and Applications, Richard Szeliski, 2010, Springer.
- Computer Vision - A modern Approach, D. Forsyth, J. Ponce, 2nd Edition 2011, Pearson India.
- OpenCV Computer Vision with Python, Joseph Howse, 2013, Packt Publishing.
- Dictionary of Computer Vision and Image Processing, R. B. Fisher, T. P. Breckon, K. Dawson-Howe, A. Fitzgibbon, C. Robertson, E. Trucco, C. K. I. Williams, 2nd Edition, 2014, Wiley.
- https://scikit-learn.org/stable/
- https://matplotlib.org/3.3.1/contents.html
- https://www.geeksforgeeks.org/learning-model-building-scikit-learn-python-machine-learning- library/
- https://pypi.org/project/opencv-python/