******

***Lab 11***

**Optical Flow**

**Learning Objectives:**

* *By the end of this lab, students will be able to:*
* *Understand the concept of optical flow and its role in motion analysis.*
* *Apply dense optical flow techniques using the Farneback method.*
* *Track sparse feature points between frames using the Lucas-Kanade method.*
* *Visualize and interpret motion information in videos using OpenCV.*

**Theoretical Background:**

**What is Optical Flow?**

Optical flow is the pattern of **apparent motion** of objects, surfaces, and edges in a visual scene caused by the **relative motion** between an observer and the scene.

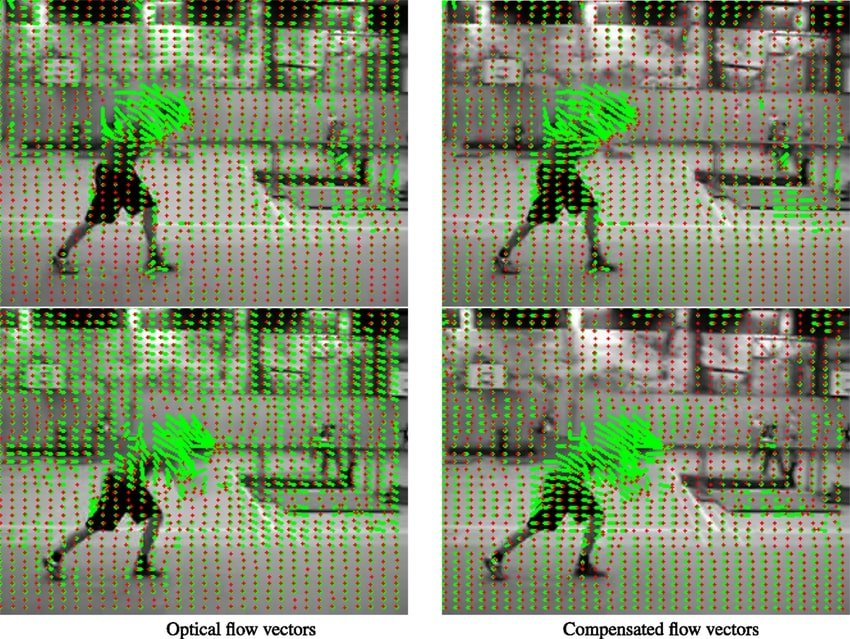
It provides information about:

* The **direction** in which pixels are moving.
* The **speed** of pixel movement.

Optical flow assumes (constraint):

* Brightness constancy: pixel intensity remains the same between consecutive frames.
* Small motion between frames.
* Smooth motion in neighboring pixels.



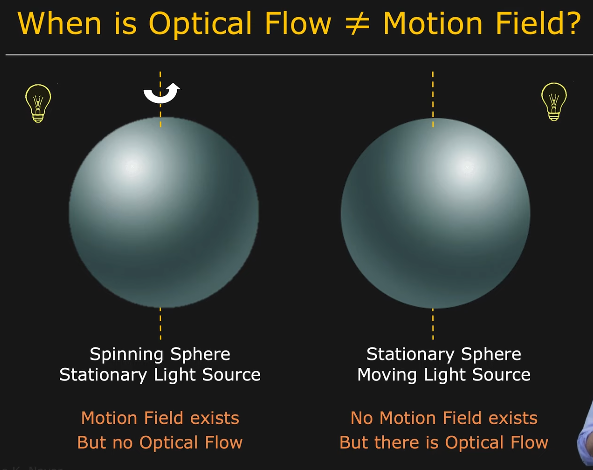
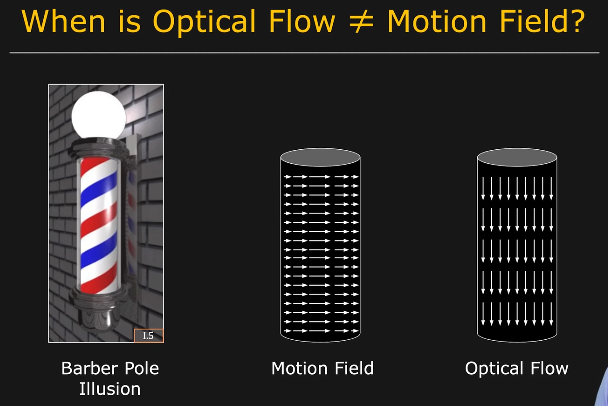


**What is the difference between (Optical flow and Motion field) ?**

***Motion Field:*** *The true physical motion of objects (or points) in the 3D world projected onto the 2D image plane.*

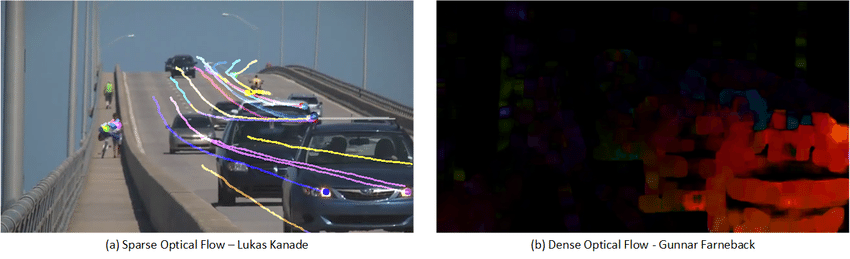
***Optical Flow:*** *The apparent motion computed from image intensity changes between two frames.*

**Ideally, Optical Flow = Motion Field, However Sometimes its not**

****

**Types of Optical Flow Estimation:**

1. **Dense Optical Flow**:
   * Calculates motion vectors for **every pixel** in the frame.
   * Example method: **Farneback’s method**.
   * Useful for capturing global motion fields.
2. **Sparse Optical Flow**:
   * Calculates motion vectors for **selected key points** (e.g., corners).
   * Example method: **Lucas-Kanade method**.
   * More efficient for tracking prominent features in videos.



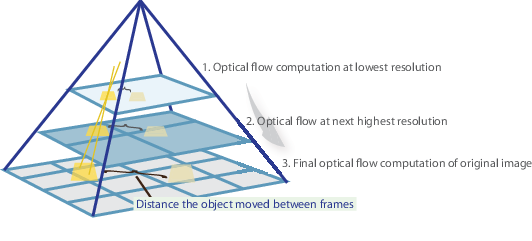
***Exercise 1: Dense Optical Flow with Farneback Method***

**Objective:**

Compute dense optical flow for all pixels between two consecutive frames.

**Method Overview:**

The Farneback algorithm approximates pixel neighborhoods with polynomials to estimate motion.



**Steps:**

**1.Read two consecutive frames from a video.**

**2.Convert both frames to grayscale**:

prev\_gray = cv.cvtColor(prev\_frame, cv.COLOR\_BGR2GRAY)

next\_gray = cv.cvtColor(next\_frame, cv.COLOR\_BGR2GRAY)

3.**Compute the dense optical flow**:

flow = cv.calcOpticalFlowFarneback(prev\_gray, next\_gray, None,

                                   pyr\_scale=0.5, levels=3, winsize=15,

                                   iterations=3, poly\_n=5, poly\_sigma=1.2,

                                   flags=0)

**4.Visualize the**:

- Use color coding (hue for direction, brightness for magnitude).

**Farneback Function Parameters Overview:**

* pyr\_scale: Image scale (<1) for pyramid building.
* levels: Number of pyramid layers.
* winsize: Window size for averaging.
* iterations: Iterations at each pyramid level.
* poly\_n, poly\_sigma: Parameters for polynomial expansion smoothing.
* flags: Optional flags like OPTFLOW\_USE\_INITIAL\_FLOW or OPTFLOW\_FARNEBACK\_GAUSSIAN.

***Exercise 2: Sparse Optical Flow with Lucas-Kanade Method***

**Objective:**

Track a set of selected points (features) between consecutive frames.

**Method Overview:**

The Lucas-Kanade method tracks points by solving motion equations over a small neighborhood.

**Steps:**

**1.Detect good features to track** in the first frame:

prev\_pts = cv.goodFeaturesToTrack(prev\_gray, maxCorners=100, qualityLevel=0.3, minDistance=7)

**2.Compute the optical flow for these points**:

next\_pts, status, err = cv.calcOpticalFlowPyrLK(prev\_gray, next\_gray,prev\_pts, None, winSize=(21,21), maxLevel=3,

            criteria=(cv.TERM\_CRITERIA\_EPS | cv.TERM\_CRITERIA\_COUNT, 30, 0.01))

3. **Draw the tracked points**:

* Draw lines or points to show how features have moved.

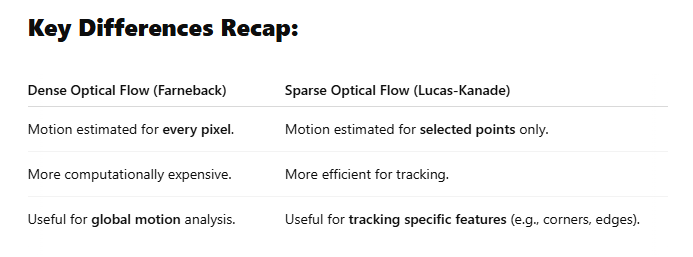
**Lucas-Kanade Function Parameters Overview**:

 winSize: Size of the search window.

 maxLevel: Number of pyramid levels.

 criteria: Termination criteria (max iterations or minimum movement).

 flags: Control behavior (e.g., use initial flow, get minimum eigenvalues).

****

|  |  |
| --- | --- |
|  |  |

**Additional reference:**

1. [**https://docs.opencv.org/3.4/d4/dee/tutorial\_optical\_flow.html**](https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html)
2. [**https://www.youtube.com/watch?v=WrlH5hHv0gE&t=237s&ab\_channel=NicolaiNielsen-ComputerVision%26AI**](https://www.youtube.com/watch?v=WrlH5hHv0gE&t=237s&ab_channel=NicolaiNielsen-ComputerVision%26AI)
3. [**https://www.youtube.com/watch?v=hfXMw2dQO4E&ab\_channel=NicolaiNielsen-ComputerVision%26AI**](https://www.youtube.com/watch?v=hfXMw2dQO4E&ab_channel=NicolaiNielsen-ComputerVision%26AI)

