Lab #8: Optical Flow Using Lucas-Kanade Algorithm with OpenCV

Fiseha B

April 9, 2025

1 Objective

In this lab, students will delve into the concept of optical flow, a crucial component in computer vision applications such as motion detection, video compression, and autonomous vehicles. Using the Lucas-Kanade method and OpenCV, we aim to:

- Understand the estimation of optical flow.
- Implement feature point tracking in a video with `cv.calcOpticalFlowPyrLK()`.
- Compute a dense optical flow field using `cv.calcOpticalFlowFarneback()` and visualize it in RGB format.

2 Pre-requisites

- Basic knowledge of Python programming.
- Python and OpenCV installed on the student's machine.

3 Installation Steps

Ensure Python and the required libraries (OpenCV, NumPy) are installed on each student's machine. If it is already installed skip the following instructions.

3.1 For Anaconda users:

1. Open the Anaconda Prompt or your terminal (for MacOS/Linux).
2. To create a new environment for this lab, run: `conda create --name cvlab python=3.8`
3. Activate the newly created environment: `conda activate cvlab`
4. Install OpenCV and NumPy within this environment by running: `conda install -c conda-forge opencv numpy`

3.2 For non-Anaconda (pip) users:

1. Ensure Python is installed on your system.
2. Open your command line interface (CLI).
3. Run `pip install opencv-python-headless numpy` to install the necessary libraries.

4 Task 1: Implementing Optical Flow with Lucas-Kanade Algorithm

4.1 Objective:

Track specified feature points in a video using the Lucas-Kanade optical flow method.

4.2 Instructions:

1. Open a video file in OpenCV.
2. Detect Shi-Tomasi corner points in the first frame.
3. Iteratively track these points using Lucas-Kanade optical flow.

```
# Import necessary libraries
import cv2
import numpy as np

# Open video file
cap = cv2.VideoCapture('video.mp4')

# Read the first frame
ret, prev_frame = cap.read()
prev_gray = cv2.cvtColor(prev_frame, cv2.COLOR_BGR2GRAY)

# Create a mask for drawing optical flow
mask = np.zeros_like(prev_frame)

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    # Convert frame to grayscale
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Calculate optical flow
    lk_params = dict(winSize=(15, 15), maxLevel=2,
                     criteria=(cv2.TERM_CRITERIA_EPS | cv2.
    TERM_CRITERIA_COUNT, 10, 0.03))
    prev_points = cv2.goodFeaturesToTrack(prev_gray, maxCorners
    =100, qualityLevel=0.3, minDistance=7, blockSize=7)
```

```
    new_points, status, _ = cv2.calcOpticalFlowPyrLK(prev_gray,
    gray, prev_points, None, **lk_params)

    # Draw optical flow vectors
    for i, (new, old) in enumerate(zip(new_points, prev_points)):
        a, b = new.ravel().astype(int)
        c, d = old.ravel().astype(int)
        mask = cv2.line(mask, (a, b), (c, d), (0, 255, 0), 2)
        frame = cv2.circle(frame, (a, b), 5, (0, 0, 255), -1)

    # Combine the frame with the mask
    output = cv2.add(frame, mask)

    # Display the resulting frame
    cv2.imshow('Optical Flow', output)

    # Update previous frame and points
    prev_gray = gray.copy()
    prev_points = new_points

    if cv2.waitKey(25) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Listing 1: Python code for Lucas-Kanade optical flow

4.3 Task Analysis:

- Experiment by changing the parameters like window size, number of levels in the image pyramid, and termination criteria.
- Reflect on how these adjustments influence the optical flow estimation. How does the Lucas-Kanade algorithm perform under various conditions (e.g., different textures or motions in the video)?

5 Task 2: Dense Optical Flow with Farneback Method

5.1 Objective:

Extend Task 1 to compute and visualize dense optical flow using the Farneback method.

5.2 Instructions:

- Implement dense optical flow estimation using `cv.calcOpticalFlowFarneback()`.
- Visualize the results in a colored format to distinguish different flow directions and magnitudes.

6 Additional Resources

For further exploration of optical flow, Lucas-Kanade method, and their applications, consult the OpenCV Documentation.

7 Task 3: Mini-Project

You may consider developing a simple application, such as a motion detection system in surveillance footage, utilizing the optical flow techniques learned in this lab.