EE 576 Project: # 5

Ahmet Furkan Akıncı

17.05.2024

I. Introduction

This report discusses the algorithms implemented in the ImageFlow program for optical flow and tracking, along with sample results and their comparative performance measured by the difference norm.

II. Algorithms

A. Optical Flow

The optical flow algorithm uses the Farneback method to calculate the dense optical flow between two consecutive frames. This method estimates the flow for each pixel by considering the polynomial expansion of neighborhoods of pixels. The steps are as follows:

1) **Convert images to grayscale:** Both input images are converted to grayscale using the `cvtColor` function from OpenCV. This reduces computational complexity and focuses on intensity changes.
2) **Compute optical flow:** The `calcOpticalFlowFarneback` function computes the dense optical flow between the grayscale images.
3) **Apply a mask:** A mask image is applied to the first frame to isolate regions of interest. This helps in focusing the flow analysis on specific objects.
4) **Identify distinct objects:** The `connectedComponentsWithStats` function labels distinct objects in the mask and provides their statistics and centroids.
5) **Calculate average flow:** For each labeled object, the average flow vector is computed by averaging the flow vectors of all pixels within the object.
6) **Draw flow vectors:** The average flow vector for each object is drawn on the output image using the `arrowedLine` function.

B. Tracking

The tracking algorithm employs the Lucas-Kanade method for sparse optical flow to track points within labeled regions between two consecutive frames. This method estimates the flow of a sparse set of points by solving the optical flow equations for patches around each point. The steps are:

1) **Convert images to grayscale:** Both input images are converted to grayscale to simplify the tracking process.
2) **Apply a mask:** The mask image is applied to the first frame to focus the tracking on specific regions.
3) **Identify distinct objects:** The `connectedComponentsWithStats` function labels distinct objects in the mask and provides their statistics and centroids.
4) **Collect points to track:** Points within each labeled region are collected. These points will be tracked across frames using the optical flow method.
5) **Track points:** The `calcOpticalFlowPyrLK` function tracks the collected points from the first frame to the second. It uses a pyramidal implementation of the Lucas-Kanade method, which allows for tracking at multiple scales.
6) **Calculate average flow:** For each labeled object, the average flow vector is computed by averaging the displacement vectors of all successfully tracked points.
7) **Draw flow vectors:** The average flow vector for each object is drawn on the output image using the `arrowedLine` function.

III. Sample Results

Fig. 1: Sample Result 1

Fig. 2: Sample Result 2

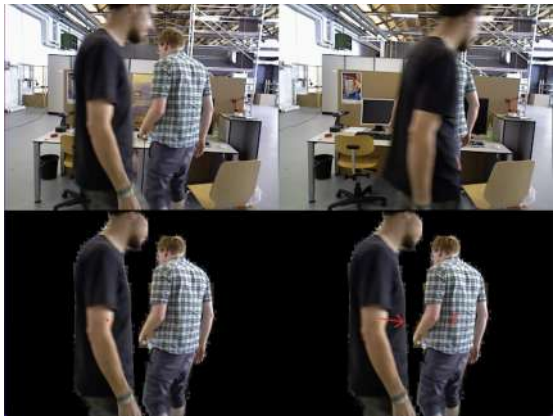

Fig. 3: Sample Result 3

IV. Comparative Performance

The performance of the algorithms is measured by the difference norm, which quantifies the average flow vector for each object. The norm of the flow vectors provides an indication of the movement detected by each method.

Sample No	Object	Method	Norm
1	1	Optical Flow	12.7934
1	2	Optical Flow	13.3138
1	1	Tracking	15.0667
1	2	Tracking	25.3081
2	1	Optical Flow	3.4695
2	2	Optical Flow	8.8852
2	1	Tracking	3.57009
2	2	Tracking	13.7077
3	1	Optical Flow	7.6247
3	2	Optical Flow	4.82807
3	1	Tracking	52.3584
3	2	Tracking	35.0374

TABLE I: Comparison of average flow vector norms for two objects in each image.
Object 1: Person with black Tshirt
Object 2: Person with square patterned shirt.

V. Conclusion

Both optical flow and tracking algorithms effectively identify and visualize motion in the image sequences. The comparative performance shows slight differences in the average flow norms, with tracking generally producing higher values. This indicates that tracking may be more aggressive in estimating motion. In most cases tracking seems to be slightly more accurate.

VI. References

https://github.com/opencv/opencv.git