Optical Flow to Track and Detect Car Speed

ENPM 673: Perception for Autonomous Robots

Group Number: 27
Sivaram Dheeraj V
Onkar Kher
Nazrin G
Xinze Li

UNIVERSITY OF MARYLAND FEARLESSLY FORWARD

Optical Flow

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 a scene. Optical flow can also be defined as the distribution of apparent velocities of movement of brightness pattern in an image. Optical flow is used by roboticists, encompassing related techniques from image processing and control of navigation including motion detection, object segmentation, time-to-contact information, focus of expansion calculations, luminance, motion compensated encoding, and stereo disparity measurement. It can be of two types-Sparse Optical flow and Dense Optical flow. Various different methods are used to calculate optical flow such as Lucas-Kanade method, Horn-Schunck method, Gunnar-Farneback method etc. The most commonly used ones being the Lucas-Kanade and the Gunnar-Farneback or the Farneback method.

Sparse Optical Flow (Lucas-Kanade method)

Sparse optical flow selects a sparse feature set of pixels (interesting features such as edges and corners) to track its velocity vector(Motion). The extracted features are passed in the optical flow function from frame to frame to ensure that the same points are being tracked. It is a simple technique which can provide an estimate of the movement of interesting features in successive images of a scene. We would like to associate a movement vector (u, v) to every such ”interesting” pixel in the scene, obtained by comparing the two consecutive images.

UNIVERSITY OF MARYLAND FEARLESSLY FORWARD

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).

$$\frac{dI}{dx}u + \frac{dI}{dy}v + \frac{dI}{dt} = 0$$

WORKING

Since sparse optical flow methods identify features (corners) to determine the optical flow at these points, the data (frames) have to be pre-processed. The most common method of detecting corners is by using Shi-Tomasi Algorithm for corner detection, in which all corners below quality level are rejected. Then it sorts the remaining corners based on quality in the descending order. Then function takes first strongest corner, throws away all the nearby corners in the range of minimum distance and returns N strongest corners. This function is more appropriate for tracking. In a nutshell, we identify some interesting features to track and iteratively compute the optical flow vectors of these points.

UNIVERSITY OF MARYLAND FEARLESSLY FORWARD

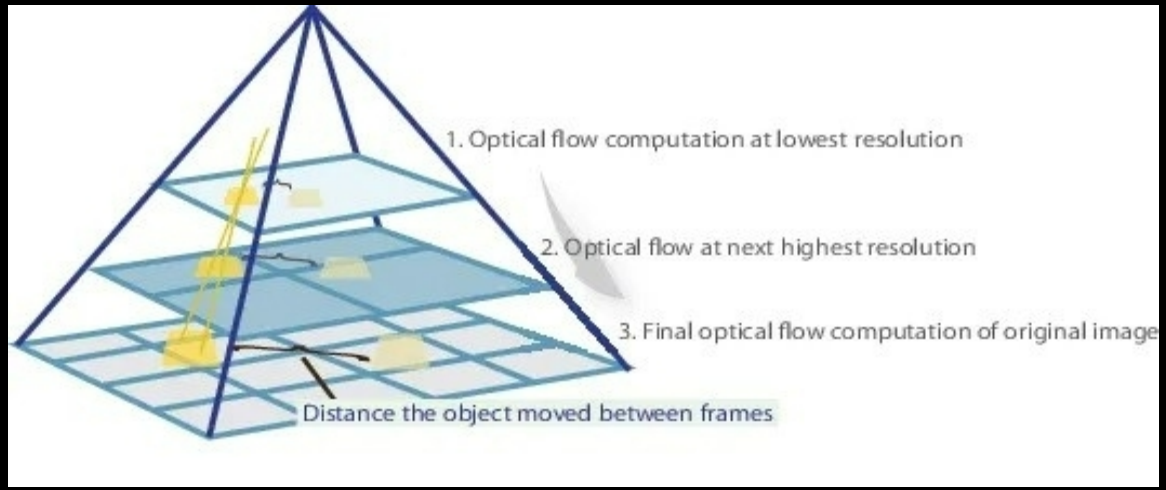
Dense Optical Flow (Farneback Method)

In dense optical flow, we look at all of the points(unlike Lucas Kanade which works only on corner points detected by (Shi-Tomasi Algorithm) and detect the pixel intensity changes between the two frames, resulting in an image with highlighted pixels, after converting to hsv format for clear visibility. It computes the magnitude and direction of optical flow from an array of the flow vectors, i.e., *(dx/dt, dy/dt)*. Later it visualizes the angle (direction) of flow by hue and the distance (magnitude) of flow by value of HSV color representation.For visibility to be optimal, strength of HSV is set to 255. OpenCV provides a function **cv2.calcOpticalFlowFarneback** to look into dense optical flow.

```
cv2.calcOpticalFlowFarneback(prev, next, pyr_scale, levels, winsize, iterations, poly_n, poly_sigma, flags[, flow])
```

Parameters:

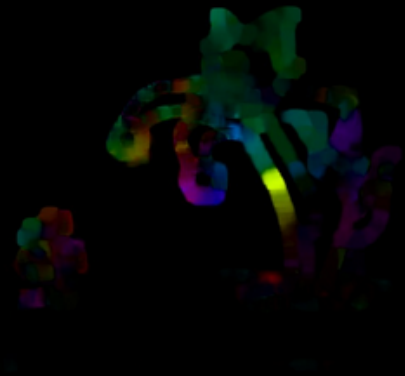
- **prev :** First input image in 8-bit single channel format.
- **next :** Second input image of same type and same size as *prev.*
- **pyr_scale :** parameter specifying the image scale to build pyramids for each image (scale < 1). A classic pyramid is of generally 0.5 scale, every new layer added, it is halved to the previous one.
- **levels :** *levels=1* says, there are no extra layers (only the initial image) . It is the number of pyramid layers including the first image.
- **winsize :** It is the average window size, larger the size, the more robust the algorithm is to noise, and provide fast motion detection, though gives blurred motion fields.
- **iterations :** Number of iterations to be performed at each pyramid level.
- **poly_n :** It is typically 5 or 7, it is the size of the pixel neighbourhood which is used to find polynomial expansion between the pixels.
- **poly_sigma :** standard deviation of the gaussian that is for derivatives to be smooth as the basis of the polynomial expansion. It can be 1.1 for *poly= 5* and 1.5 for *poly*= 7.
- **flow :** computed flow image that has similar size as *prev* and type to be CV_32FC2.
- **flags :** It can be a combination of-
  OPTFLOW_USE_INITIAL_FLOW uses input flow as initial approximation.
  OPTFLOW_FARNEBACK_GAUSSIAN uses gaussian *winsize*winsize* filter.



UNIVERSITY OF MARYLAND FEARLESSLY FORWARD

WORKING

Dense Optical flow computes the optical flow vector for every pixel of the frame which may be responsible for its slow speed but leading to a better accurate result. It can be used for detecting motion in the videos, video segmentation, learning structure from motion. There can be various kinds of implementations of dense optical flow. The example below will follow the Farneback method along with OpenCV. For OpenCV's implementation, the magnitude and direction of optical flow from a 2-D channel array of flow vectors are computed for the optical flow problem. The angle (direction) of flow by hue is visualized and the distance (magnitude) of flow by the value of HSV color representation. The strength of HSV is always set to a maximum of 255 for optimal visibility.
The method defined is calcopticalFlowFarneback() .



Results

Outputs

Final Project

UNIVERSITY OF MARYLAND FEARLESSLY FORWARD

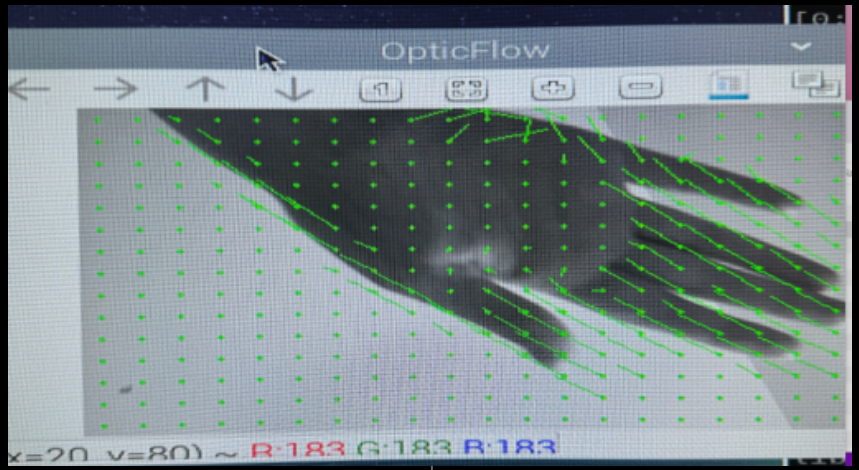
YOLO

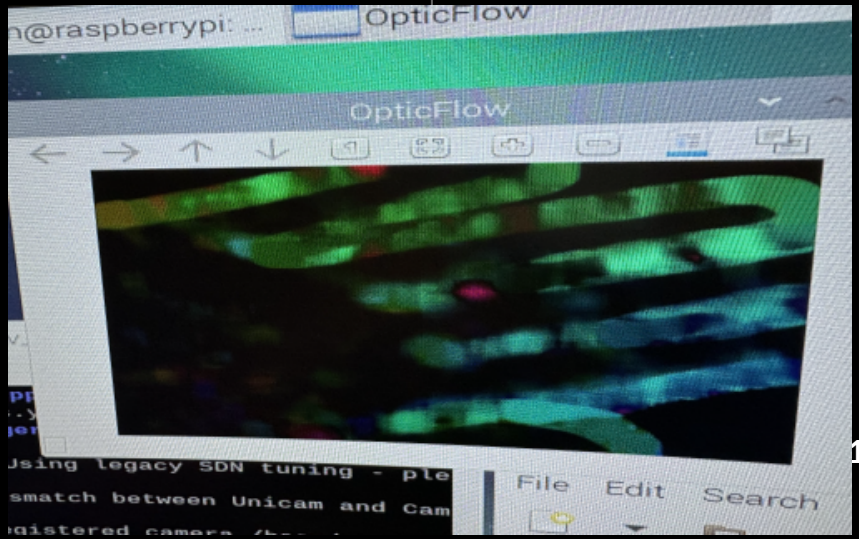
Object detection is a task that involves identifying the location and class of objects in an image or video stream. The output of an object detector is a set of bounding boxes that enclose the objects in the image, along with class labels and confidence scores for each box. Ultralytics YOLO is an efficient tool for professionals working in computer vision and ML that can help create accurate object detection models.

UNIVERSITY OF MARYLAND FEARLESSLY FORWARD

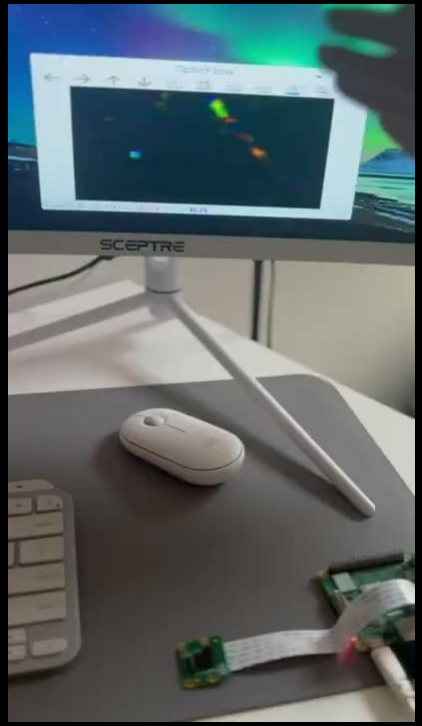
Hardware

The implementation of Dense Optical Flow on a Raspberry Pi 4 camera has been successfully achieved.





UNIVERSITY OF MARYLAND
FEARLESSLY FORWARD


SCEPTRE

Conclusion

Optical Flow

We have successfully implemented the concepts of optical flow in estimating the speeds of vehicles on a highway from a video stream.

YOLO

In addition to the optical flow method we have demonstrated a very basic implementation of the YOLO tool and used machine learning to detect, track and estimate the speeds of the objects from the same input video stream.

Hardware

Furthermore, we have tried to implement demonstrating optical flow by using hardware components such as Raspberry Pi, to demonstrate the concept of optical flow and its applications in real-life situations.

Future Work

Future Work

YOLO

Working on YOLO further, to make it identify other vehicles as well such as buses, trucks, trailers, bikes, etc. for the purpose of speed detection on highways.

RAFT

Using RAFT and other deep learning methods to track vehicles, segment between objects/vehicles, estimate their speeds, which has a great application in autonomous vehicles.

Versatile

Working towards a script that can identify any objects in any environment and estimate speeds, calculate and track trajectories and motion of objects. Also, deploying this on a hardware setup, so that it can be implemented in practical scenarios.

THANK YOU