Report: Enriching data with optical flow

Jiří Hörner

July 15, 2017

I have evaluated two optical flow algorithms for extracting flow information from video. Gunner Farneback's algorithm [1] for dense optical flow and Lucas-Kanade sparse flow algorithm [2]. I have used implementations of these methods from the OpenCV library. Source files in Python are attached.

I have focused on extracting raw information from the flow suitable for further processing by machine learning methods. My approach therefore does not necessarily generates a perfect information.

1 Dense optical flow

Farneback's method produces flow information for every pixel in the source frame of the video. The flow is computed between two consecutive frames in video.

Because the flow is computed for each pixel this method is computationally expensive. To get near real-time performance I have resized the input video to 480×234. Performance of this method might be a significant limitation, especially when processing must be done onboard of the drone with the limited compute power.

As the method have to generate flow information for every pixel, flow in some areas of the video frame contains a lot of noise. This is clearly visible along the long edges in the video, see figure 1. This makes the further processing hard. Flow in the featureless areas is usually zero.

Because the resulting flow is represented by $480 \times 234 \times 2 = 224640$ values, which seems really excessive for most of the learning methods, I have aggregated this flow. The aggregated flow represents flow in 3×3 rectangular areas of the frame. For each area a flow vector with the largest euclidean metric is selected to represent the area. This allows us to represent the flow in $3 \times 3 \times 2 = 18$ values. This approach is noise-sensitive and it might be necessary to reconsider this approach based on the output of the learning method. We might want to employ RANSAC-based method for extracting geometrically consistent information (but that collides with the philosophy to extract the raw flow).

1.1 Script

Attached to this report is a script for experimenting with Farneback flow. The script is written in Python and requires OpenCV 3 and numpy libraries. Usage example:

```
./farneback.py [<video_source>]
./farneback.py camera_auto10/camera10.AVI
```

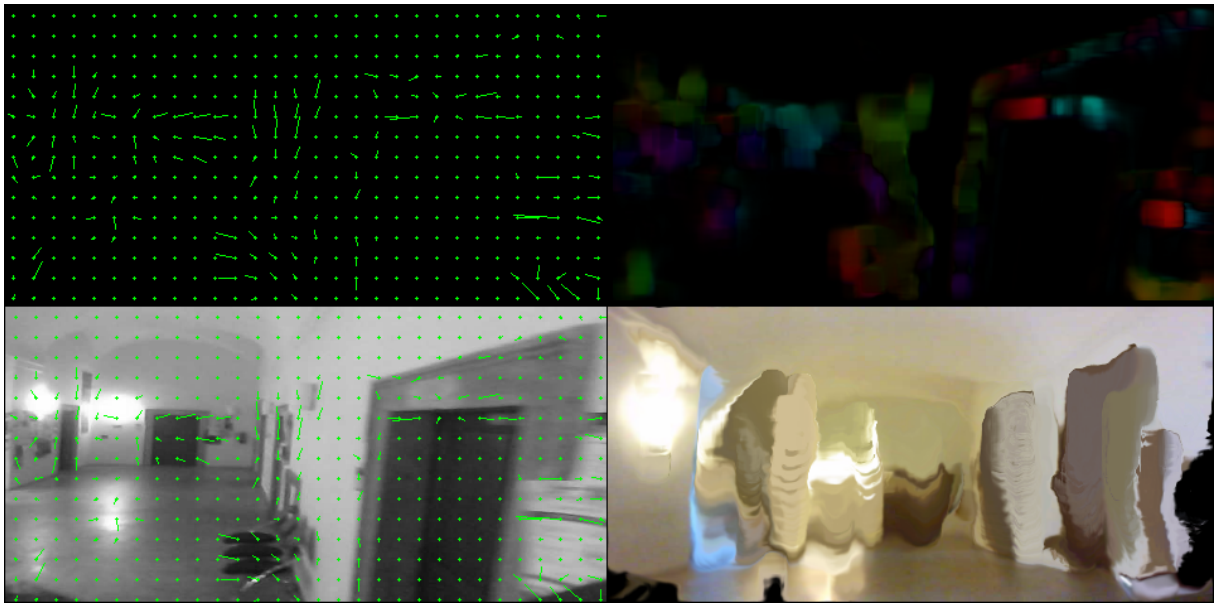

Figure 1: Top-left: dense flow visualised on black background. Bottom-left: dense flow visualised on original video frame. Note the noise around the doors. Top-right: HSV visualisation of the same flow. Bottom-right: Previous frame transformed according to the flow.

Script will show flow visualisation in the new window. CSV file with summarized flow is outputted alongside the video. File with the flow has the following format:

```
timestamp,<18 values comma separated>
```

Timestamp is position in video in milliseconds. 18 values are summarized flow for 3x3 image areas as discussed above (each area has flow in x and y direction).

2 Sparse optical flow

Lukas-Kanade method tracks only selected keypoints in frames. This method is therefore much less computationally demanding than the previous dense flow method. In my setup new keypoints are detected every 5 frames.

For tracking only a suitable keypoints are selected, using a corner quality measure. This leads to a considerably less noise in the computed flow. Moreover Number of tracked keypoints provides a good measure of confidence for this method. Confidence may be also useful output for machine learning, or could be used for sensor fusion using Kalman filter.

For our purposes we have decided after a discussion to reduce the number of features processed by machine learning. The minimum information available from sparse flow is a one keypoint. I have chosen to select a random keypoint in each frame and output its flow. Although this can't provide a complete motion information, it might help when estimating some specific moves, especially rotations. See figure 2 for flow computed for rotation and figure 3 for frame with translation.

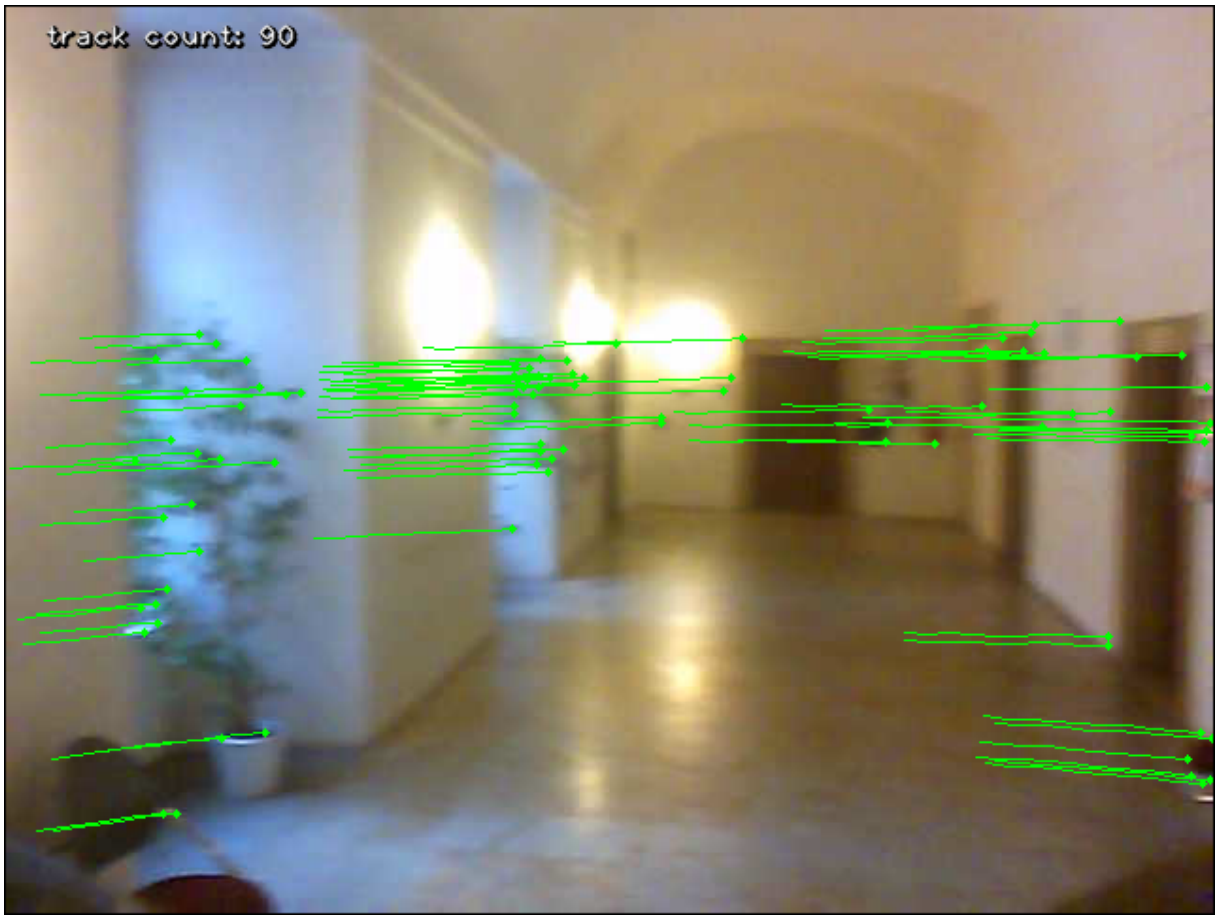

Figure 2: Frame recorded by drone with dominant rotation. Keypoints tracked by Lukas-Kanade method are show in green together with history of tracked positions of each keypoints.

Figure 3: Frame recorded by drone with dominant translation. Keypoints tracked by Lukas-Kanade method are show in green together with history of tracked positions of each keypoints.

2.1 Script

Attached to this report is a script for experimenting with Lucas-Kanade flow. The script is also in Python and requires OpenCV 3, numpy and external file `common.py`. Usage example:

```
./lucas-kanade.py [<video_source>]
./lucas-kanade.py camera_auto10/camera10.AVI 2>camera_auto10/camera10_flow_lucas-kana
```

Script will show flow visualisation in the new window. Flow information will be printed to stderr (It might be a good idea to redirect stderr to a file). Flow info has the following format:

```
timestamp,flow_x,flow_y
```

Timestamp is position in video in milliseconds. Flow is for 1 selected tracked point as discussed above.

References

[1] Gunnar Farnebäck. *Two-Frame Motion Estimation Based on Polynomial Expansion*, pages 363–370. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[2] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.