SVO – semi-direct visual odometry as data enrichment tool

Peter Zborovsky

ABSTRACT
In this paper we propose usage of visual odometry tool (SVO - semi-direct visual odometry introduced by C. Forster) to get relative change in pose between two consecutive camera frames. Such information describes camera egomotion in iterative manner and thus provide an efficient data enrichment tool.
The paper is organized as follows. First, in section 1 we briefly describe semi-direct visual odometry pipeline as well as problems introduced by its usage. Proposed solution and system overview are in section 2. Experiments description and discussion are described in sections 3 and 4 and conclusion can be found in section 5.

1. Semi-direct visual odometry

1.1 SVO overview

SVO is monocular visual odometry algorithm that is calculating global camera pose estimation solely from visual (camera) input. Algorithm is operating directly on image intensities (direct method). Rather than evaluating intensity differences along the whole image, only small patches around tracked features are used instead (sparse model) which significantly reduces computational time.

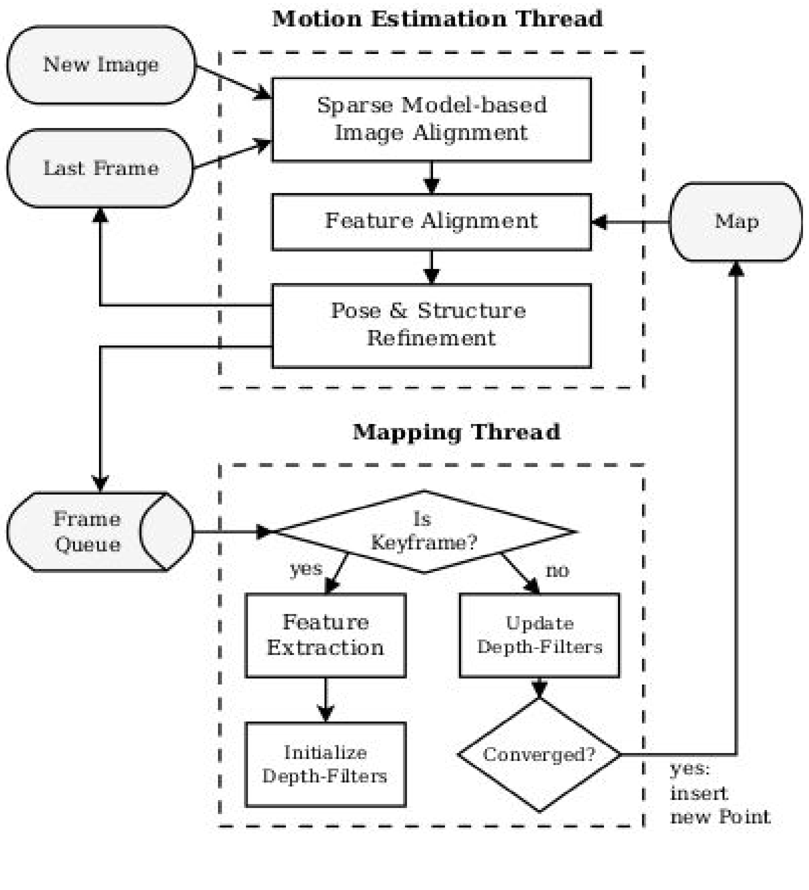

Fig. 1: SVO pipeline: algorithm itself is divided into two main threads: Motion Estimation Thread and Mapping Thread.

1.1.1 Motion Estimation Thread

Firstly, a frame-to-frame motion is estimated by minimizing intensity difference of detected features that correspond to the projected real world 3D points (This is called Sparse Model-based image alignment). In order to work with picture intensities, small patches around detected features are used. For computational reasons, these patches are not transformed. This requires small frame-to-frame motion and also small patches. This step aligns camera pose of new frame with respect to the last frame.
Second step optimize camera pose with respect to the 3D map but for the price of violating geometrical constraints (relaxation step).
In last step, both camera pose and 3D map are optimized in order to minimize the reprojection error introduced by previous optimization step (this step is called local bundle adjustment).

1.1.2 Mapping Thread

The key contribution of Mapping Thread is depth filter. For each detected feature in the filter estimates its depth over time as new frames are being observed. The estimation is based on probability distribution: each new frame where the already known feature is observed is reducing the variance of probability distribution of that particular feature depth. When the variance is small enough, the depth estimate is used to calculate a new 3D point in the map.
Another important part of mapping thread is keyframe selection. When sufficient distance between current frame and last keyframe is detected the current frame is selected to be the new keyframe. Keyframes are used to extract new features and for pose and structure optimization.

Further details and derivations can be found in [1] and [2].

1.2 SVO usage problems

For purposes of this application we are using low-cost cameras which encounters several problems.

1.2.1 Hardware problems

The main disadvantages of low-cost cameras are small frame rate and rolling shutter. Small frame rate introduces loss of tracked features with fast camera movements as sparse model image alignment expects only small frame-to-frame motion (see section 1.1.1). Rolling shutter with fast camera movements causes image deformation (skewed image) and thus 2D coordinates of tracked features does not fully match the projection of corresponding 3D points which again results in loss of tracked features.

1.2.2 Software problems

First problem is same as in previous section: small frame-to-frame requirement of sparse model image alignment for which the most suitable camera would be one with higher frame rate and global shutter as we have empirically found out.
Second disadvantage is represented by keyframe selection which is designed for downward looking cameras (i.e. translation movements to the side instead of forward motion or pure rotation). New keyframe is selected when the distance between new frame relative to all keyframes exceeds 12% of average scene depth. With forward looking camera the average scene depth is high (unless there is an big obstacle, e.g. wall) and thus it is unlikely for new frame to be selected as a keyframe during forward motion. Furthermore, the depth-filter is updated only with new frames (not old ones) and with

forward motion the parallax of a feature is naturally small which causes depth-filter to converge slowly.
Last problem, which is not tight with SVO implementation but rather with physical model of motion is pure rotation. During pure rotation movement no parallax is observed and thus it is impossible to add new 3D points and neither is possible to update depth-filters and track new features as no new keyframes are added.

These problems may reflect in losing too many features and consequently resulting in inability of camera motion estimation (losing motion track).

Some of above mentioned problems could be removed by tuning and changing SVO implementation but that is out of scope of this work.

2. Proposed solution

Main goal of application is to enrich sensor readings by information about camera egomotion. By using SVO pipeline we are able to gain following informations about camera motion:

1. Global position - current frame transformation against global frame $T_{k,w.}$
2. Relative pose change between two consecutive camera frames $T_{k,k-1.}$

Since we may encounter losing motion track problem (see section 1.2), it is preferable to use relative motion estimation $T_{k,k-1}$ which relies solely on previous frame.

$$T_{k,k-1} = T_{k,w} . T_{k-1,w}^{-1} \tag{1}$$

In our system we are using two main features in order to gain as much reliable data (for data enrichment purposes) as possible: System restart and “mark” variable. System restart is initiated when SVO pipeline is unable to track motion for too long (approximately 3 seconds). Mark variable labels each frame with “quality” of tracked data. There are three options for such label:

1. SVO is tracking motion correctly.
2. SVO is lost but after some time it recovers camera position. Missing motion estimates are linearly interpolated between last known position and the first recovered position.
3. SVO is lost for too long and system restart is initiated. The missing motion estimates between last known position and the frame at which system restart was initiated are filled with last known motion estimate.

Mark variable is assigned to each frame and serves as distinguishing parameter.

2.1 Input and output of the application

Application input is video sequence of calibrated camera. After system processes the video sequence, table with 8 columns is created and saved as .csv file. The table is organized as follows:

1st column:	timestamp of processed frame.
2nd -4th column:	translation vector between current and previous camera frame.
5th - 7th column:	rotation vector between current and previous camera frame.
8th column:	mark variable value (described in previous section).

2. System setup
For purposes of this project we have decided to use easy accessible low-cost cameras:
cam1: …SriCam??... (640 x 480 pixel resolution, rolling shutter)
cam2: GoPro Hero 3+ camera (used only for comparison purposes – 1280 x 960 pixel resolution, rolling shutter)

2.2 System overview

When designing the usage of SVO as data enrichment tool accompanied with low-cost camera, one have to consider high probability of losing motion track problem occurrence ("getting lost") and also the fact that output data must be organized as sequence of sensor readings.

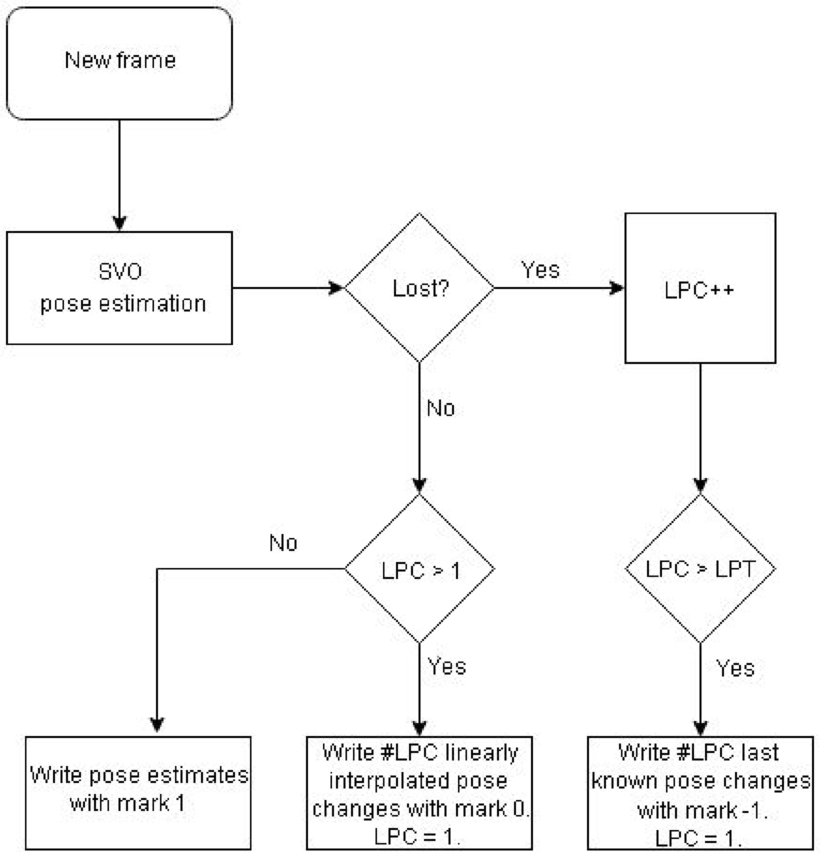
New frame
SVO pose estimation
Lost?
Yes
LPC++
No
No
LPC > 1
LPC > LPT
Yes
Yes
Write pose estimates with mark 1
Write #LPC linearly interpolated pose changes with mark 0. LPC = 1.
Write #LPC last known pose changes with mark -1. LPC = 1.

Fig. 2: proposed SVO usage as data enrichment tool.

As can be seen on Figure 2 we introduce following variables and constants:
LPC – Lost Pose Counter – counter of frames that were not localized by SVO (they are "lost").
LPT – Lost Pose Threshold – if LPC exceeds this threshold the system is reinitialized.
Mark – label for type of data written to an output file:

1 – SVO motion estimation.
0 – interpolated motion estimation.
-1 – last known motion estimation.

Mark variable is used for further possible usage of data for reinforcement learning. In such way learning algorithm can distinguish between estimated motion, interpolated motion and solely guessed motion estimation.

3. Experiments and observations

For testing purposes we have used 3 video datasets:
dat1: camera10.AVI - translation and rotation movements of camera mounted on quadcopter
dat2: PICT0001.AVI – forward & backward motion of camera mounted on quadcopter
dat3: gopro.mp4 – slow translation movements of hand-held camera.

3.1 Observation 1

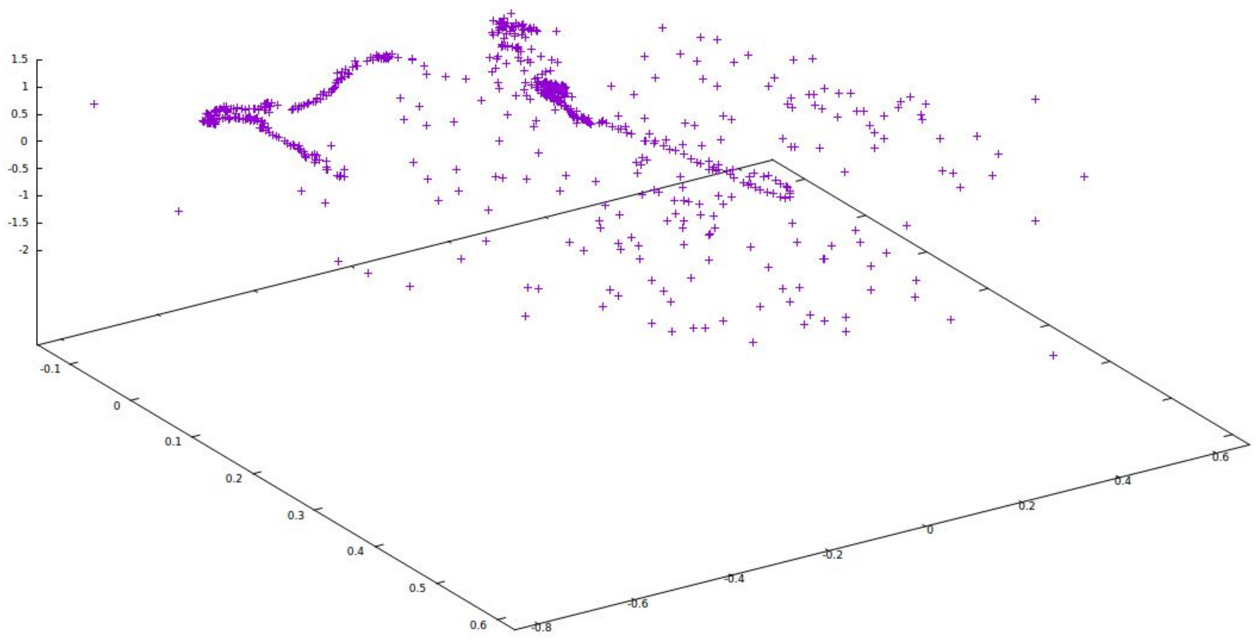

Fig3. Observation 1: cam1 dat1 – 1807 frames, 796 estimated poses (translation and rotation movements).
14 system resets initiated, 12 successful.

3.2 Observation 2

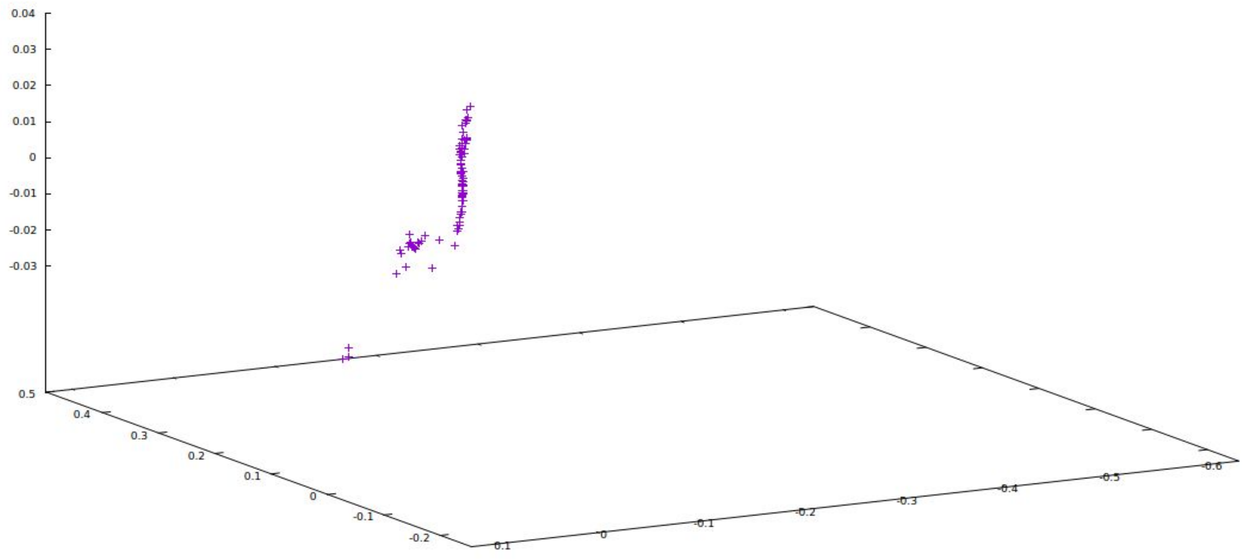

Fig.4: Observation 2: cam1 dat2 – 1221 frames, 113 estimated poses (forward & backward motion).
4 system resets initiated, 0 successful.

3.3 Observation 3

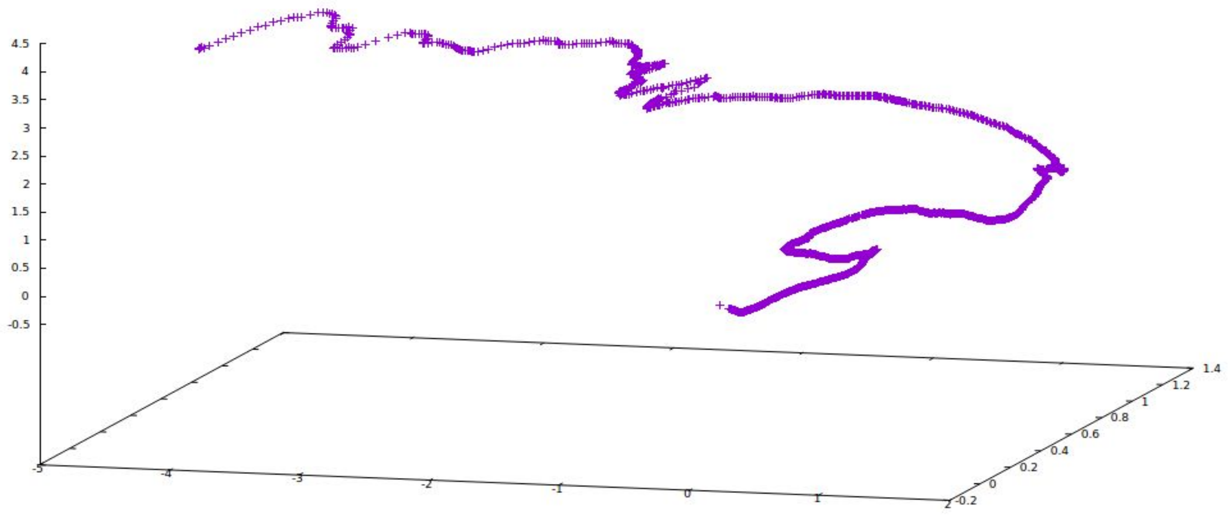

Fig.5: cam2 dat3 – 2333 frames, 2046 estimated poses (slow linear transformation).
0 system resets initiated, 0 successful.

4. Discussion

On figures 3 – 5 are visualized only data marked with n. 1 (in order to obtain number of points estimated directly by SVO pipeline). As can be seen, SVO pipeline does not have problems with estimating linear motion with few rotations, however, when rotation or forward motion is introduced algorithm fails and is forced to reset the system.
In observation 1 on dataset 1 is presented mixture of rotation and linear translation movements. After significant rotation is presented system fails to estimate the motion and consequently it initiate the system restart after which it successfully estimate the motion again in most of the cases (12 out of 14). On figure 4 the problem of forward motion can easily be observed. Only initial pointcloud of upward motion is presented and after initiation of forward motion SVO gets immediately lost and not even system reset helps to estimate the motion again.
Finally, on figure 5 – observation 3 on dataset 3 is obvious the benefit of wide field of view gopro camera that together with appropriate motions exhibits 0 outliers and needs 0 system restarts.

5. Conclusion

In this work we have briefly described SVO pipeline for motion estimation and proposed its application for data enrichment for further usage in machine learning. SVO algorithm as described in [1] has troubles with motion estimation of pure rotation as well as forward motion. While our system successfully overcame the rotation problem directly by reseting the SVO algorithm, we were not able to solve the problem of forward motion with such direct method. However, by introducing the Mark variable which distinguish the “quality” of data, our system is suitable for machine learning purposes.

REFERENCES

[1]C. Forster, M. Pizzoli, and D. Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In
IEEE Int. Conf. on Robotics and Automation (ICRA), Hong Kong, 2014.
[2]C. Forster, Z. Zhang, M. Gassner, M. Werlberger, D. Scaramuzza. SVO: Semi-Direct Visual Odometry for Monocular and Multi-Camera Systems. IEEE Transactions on Robotics, Vol. 33, Issue 2, pages 249-265, Apr. 2017.