Stereo odometry based on careful feature selection and tracking

Igor Cvišić
Department of Control and Computer Engineering
Faculty of Electrical Engineering and Computing
University of Zagreb
Unska 3, 10000 Zagreb, Croatia
Email: igor.cvisic@fer.hr

Ivan Petrović
Department of Control and Computer Engineering
Faculty of Electrical Engineering and Computing
University of Zagreb
Unska 3, 10000 Zagreb, Croatia
Email: ivan.petrovic@fer.hr

***Abstract*—In this paper we present a novel algorithm for fast and robust stereo visual odometry based on feature selection and tracking (SOFT). The reduction of drift is based on careful selection of a subset of stable features and their tracking through the frames. Rotation and translation between two consecutive poses are estimated separately. The five point method is used for rotation estimation, whereas the three point method is used for estimating translation. Experimental results show that the proposed algorithm has an average pose error of 1.03% with processing speed above 10 Hz. According to publicly available KITTI leaderboard, SOFT outperforms all other validated methods. We also present a modified IMU-aided version of the algorithm, fast and suitable for embedded systems. This algorithm employs an IMU for outlier rejection and Kalman filter for rotation refinement. Experiments show that the IMU based system runs at 20 Hz on an ODROID U3 ARM-based embedded computer without any hardware acceleration. Integration of all components is described and experimental results are presented.**

I. Introduction

Odometry is the process of estimating the change in robot position relative to its surrounding. Odometry plays an essential role in autonomous robot navigation. It is a dead reckoning process, which means that the robot determines its motion incrementaly, by integrating the current motion to the previously determined position. As such, dead reckoning is subject to cumulative errors. The most common sensors used for odometry are wheel encoders. If the surrounding scene is rich with texture, one of the most accurate sensors for odometry is camera, and this process is commonly known as visual odometry. A detailed overview of this topic is given in [1].

From two views taken from a single camera, rotation and translation up to the scale can be obtained. On the other hand, having two views from a calibrated stereo camera, both the rotation and translation between them can be exploited. The basic principle of stereo odometry is motion estimation between two successive frames, and integrating the motion through time to obtain the current position. Methods for estimating motion from two views can be divided into two groups - feature based methods and direct methods. Feature based methods use feature detectors/descriptors such as Harris, SIFT, SURF or FAST features to detect a sparse set of salient points from the real world and match them between the frames. Feature based methods can be improved by tight-coupling with IMU data. The weakest point of feature-based methods are wrong correspondences, which can be resolved through simple IMU integration. Direct methods operate directly on pixel intensities. They use more information from the image then methods with features and therefore can eventually provide more accurate estimation. Several direct methods for monocular cameras have been recently presented [2], [3].

978-1-4673-9163-4/15/$31.00 © 2015 IEEE

Odometry based on motion integration, obtained from two consecutive frames only, is prone to drift. The most common methods for reducing the drift are simultaneous localization and mapping (SLAM) and bundle adjustment (BA), both computationally more complex. From a user's perspective, the most important and usually the only information provided by the odometry is the current position of the vehicle. The KITTI [14] vision benchmark scoreboard uses translation error as the sole parameter for ranking, and the rotation error is shown for reference only. However, it is worth mentioning here that in dead reckoning process such as odometry, the final position is more sensitive to rotational errors during the integration than to translational ones. Therefore, it is beneficial to put additional effort to recover rotation with extra precision.

Hence, in this paper we propose an effective and robust visual odometry algorithm based on simple feature tracking (SOFT). Egomotion estimation is split into two parts, one for estimating rotation, and the other for estimating translation. This concept is experimentally proven to boost overall performance of the algorithm. SOFT shows an average translational error of 1.03% of the traveled distance and 0.0029 deg/m rotational error on the KITTI vision benchmark dataset. According to KITTI leaderboard, it outperforms all other validated visual odometry methods [1].

The paper is organized as follows. After the introduction, related work is presented in Section II. Section III presents the core parts of proposed feature processing algorithm, followed by Section IV in which the resulting feature coordinates are used to estimate the ego motion. Section V describes modified version of the algorithm more suitable for embedded systems. Experimental results for both versions of the algorithm are shown in Section VI and conclusions are made in Section VII.

II. Related work

Visual odometry is a special case of a more general problem called structure from motion (SfM). SfM simultaneously recovers relative camera poses with 3D structure of viewed

[1] www.cvlibs.net/datasets/kitti/eval_odometry.php

object and it is commonly used in computer vision community. The term visual odometry together with some innovative concepts first appeared in [5]. More recently, notable improvements have been made by means of feature integration [6] and bundle adjustment [17]. Work in [8] showed that calibration model of a real camera is inadequate to capture exact physical property of the lens. Authors propose to alleviate the disturbance model by a local camera model formulated as a deformation field over a rectangular superpixel lattice in the two images of stereo pair, learning the parameters from groundtruth motion. While significant improvement in accuracy of the camera motion is achieved, this method requires large amount of training data and a priori known groundtruth motion, which is not always available. Also, groundtruth acquired with GPS-IMU combination in outdoor environment is only approximate of the real world trajectory. When working with outdoor datasets, one must be aware that provided groundtruth between any two consecutive frames may contain a larger error than the one of a visual odometry estimate. The lack of an exact groundtruth restrains development and testing of new methods, as it is hard to single out the moment when error in odometry estimate rises above the mean.

One of the available libraries for visual odometry is libviso [9]. By examining libviso performance on a KITTI dataset and its dependance on a few key parameters, several important conclusions can be drawn out.

- This algorithm estimates motion by minimizing image reprojection errors. However, resulting trajectory is much more precise when reweighting is employed, i.e. features that are more distant from the image center are taken into account with lower weight, implying inaccurate image rectification. Since the image is considered rectified, there is no information upon which an optimal reweighting function could be determined.
- Corespondances between the features in the left and the right image of the KITTI dataset have tolerance of few pixels on the y axis. This suggests an existence of inter-camera calibration error.
- Outliers are rejected by random sampling consensus (RANSAC), and the inlier threshold is set to two pixels. While it is feasible to determine feature position with subpixel precision, majority of correct matches will be rejected if the inlier threshold is less than two pixels. Apart from revealing an inadequate calibration model, this means that RANSAC method is ineffective for rejecting false matches that appear close to the correct solution.

The previous discussion gives an insight to different error sources, where the trajectory estimation strongly relies on the accuracy of the lens model. Hence, improving the feature localization on a subpixel level or using a better quality descriptor cannot bring significant improvements in egomotion accuracy as long as the calibration model is not adequate. This problem is present in any rectified image captured by a real lens, but it would be impractical to obtain an accurate mathematical model of each lens used for visual odometry. Therefore, in order to obtain more accurate trajectory estimation, one should choose an egomotion estimation method which is less sensitive to inaccurate calibration.

III. Feature processing

In this section, we describe the feature processing part of the algorithm. The results of this part are image coordinates (u, v) of a subset of points that will be used in egomotion estimation.

A. Feature matching

The first stage of the algorithm is extraction of corner-like features in left and right images of the current frame. For corner detector, we utilize blob and corner masks on an input image as described in [9]. After the corners are detected, non-maximum suppression is employed. Correspondence between remaining corners is determined through the sum of absolute differences (SAD) over a sparse set of pixels in a fixed-size window in the image derivative domain. SAD method is simple and fast, but susceptible to outliers. At this stage, some of the outliers are rejected by circular matching. Circular matching implies that each feature has to be matched between left and right images of two consecutive frames, requiring four matches per feature. Features in time step t are tracked and matched through sequence of frames, starting from left frame in time step $t-1$ and ending with the same frame. If the feature is correctly matched in all frames of the aforementioned sequence, the circle is closed and the feature in the last frame of the sequence coincides with the feature in the first frame. If that is not the case, circle of matches is not closed and the feature is rejected as outlier. If the circle is successfully closed, additional check is performed with normalized cross correlation (NCC) on a 25×25 pixels patch around the feature position. NCC is more reliable than sparse SAD, but it is also significantly slower. In order to maintain an overall real-time performance, NCC is used only for gating after fast SAD circular matching. Remaining outliers are rejected with RANSAC.

B. Feature selection

Using only a subset of features for egomotion estimation significantly reduces computational time of the algorithm, thus enabling a real-time performance. Furthermore, experiments showed that if the subset is carefully selected, including additional feature points into the algorithm only deteriorates overall performance. Precise estimation of the egomotion requires that both far and near features are used in calculation of egomotion, and that features are uniformly distributed over the image [10]. These conditions are fulfilled by means of bucketing. The image is divided into 50×50 pixels sized rectangles, i.e. buckets. In every bucket some limited number of features are retained, and the others are discarded. The selection of features that will be retained inside the bucket is described as follows.

1) Features are separated into four distinct classes (corner max, corner min, blob max and blob min).
2) Features inside each class are sorted by strength, i.e. by pixel value in image created by filtering the input image with blob and corner masks.
3) The strongest feature from each class is pushed into the final list.
4) Step 3) is repeated until all features are pushed into the final list
5) First n features from the final list are selected for motion estimation.

Straightforward sorting of all features together inside the bucket can result in first n features from the sorted list all belonging to the same class, which could cause bias in remaining stages of the algorithm.

C. Feature tracking

Each feature is represented by the following properties:

- unique identifier (ID),
- age,
- refined current position in the image,
- feature strength,
- belonging class,
- initial descriptor.

Upon a feature match, the position of the feature in the current frame is refined on a subpixel level with respect to the feature position in the previous frame, while its age is increased by one. Refined position is stored and used in the next iteration. As long as the feature is alive, the same initial descriptor is used for feature position refinement. Using the same descriptor reduces drift and serves as a measure of dissimilarity between the current appearance of the feature and the initial one. Intense change in perspective relative to the first appearance of the feature will result in unsuccessful refinement and the death of the particular track. Experiments showed this method provides better results than the one employing the current descriptor at each iteration. This way the descriptor is allowed to gradually change through time, which extends feature life. However, such descriptor is prone to drift and therefore feature quickly loses coincidence with initial real-world point being tracked. Features that are tracked for longer period of time are considered to be more reliable, with lower probability of being an outlier, therefore one should always chose the older feature for the next iteration. Also, some features with strong image appearances can appear even lacking the correspondence in the real world, e.g. intersection of horizontal line in the foreground with the vertical line in the background [11]. Therefore, the comparison function for feature selection from particular bucket is modified as follows:

$$select(x,y) = \begin{cases} stronger(x,y), & \text{if } age(x) = age(y) \\ older(x,y), & \text{if } age(x) \neq age(y). \end{cases}$$

Drift is reduced through the mechanism of propagating refined position. Therefore, matches are established between current and previous occurrence only, since the information about displacement relative to the first occurrence is preserved in the refined feature position. In order to retain as much tracks as possible and to further decrease the number of outliers, two pass matching is implemented. Firstly, only one feature from each bucket is used and initial rotation and translation is estimated. Secondly, the pass is executed with greater number of features per bucket, and the matching is constrained with initially estimated rotation and translation. Constrained matching provides set of features with less outliers which leads to more accurate estimation.

IV. Egomotion estimation

Egomotion estimation is split into two parts. Firstly, the rotation is estimated using a direct method, and secondly, the resulting rotation is used for estimating translation by employing minimization of reprojection errors in the image plane.

A. Rotation estimation

Rotation is estimated with the five point method typically used in monocular cases [12]. Therefore, only left camera is used for rotation estimation. Since the camera is calibrated, epipolar constraint between two views can be expressed with

$$q'^T E q = 0, \tag{1}$$

where the matrix $E = [t]_x R$ is the essential matrix, while q and q' are corresponding homogeneous image coordinates of the two views. Matrix E is 3x3 matrix, but since the scale is unobservable, eight unknowns have to be solved. Upon five correspondences, five equations can be constructed. By using the following properties of the essential matrix,

$$det(E) = 0 \tag{2}$$

and

$$2EE^T E - tr(EE^T)E = 0, \tag{3}$$

additional constraints are imposed to the system of five equations, and solution of such system lays in one of the roots of the tenth degree polynomial [12]. Five point method is used in conjunction with RANSAC. A number of random five point subsets are taken from the total set of points, and essential matrix is calculated for each subset. Finally, the essential matrix that has the largest set of inliers among all the points is selected as the final solution. In our experiments, refining the final solution with more than five points, i.e. inliers that are very close to the best five point solution, only increases the average odometry error, hence we do not discuss this possibility of refinement any further. In some rare situations, configurations of inliers appeared such that the method converged to a local minimum, resulting with rotation shifted for a few degrees off the original five point result. It is stated in [12] that this method may not always converge to the global minimum, hence we have employed the five point configuration only. The five point method offers superior performance in rotation estimation when processing realistic images with outliers. This stems from several reasons:

- It has a closed-form solution and does not depend on an initial guess like the optimization methods.
- Its solution stems from minimum set of points, probability that one of them is an outlier is small.
- As monocular methods, it uses pairs of features instead of quads of features. Since feature pair is determined by one correspondence, and feature quad requires four correct correspondences, pairs are correctly matched with higher probability.
- Monocular method is not affected by imperfect calibration between the left and the right camera.

Based on the feature tracking, the rotation between current frame in time t and frame from $t-2$, denoted q^t_{t-2}, can be

calculated. Since the previous rotation q_{t-2}^{t-1} from $t-2$ to $t-1$ is known, the current rotation can alternatively be calculated as

$$q_{t-1}^{t'} = (q_{t-2}^{t-1})^{-1} q_{t-2}^{t}, \tag{4}$$

where $q_{t-1}^{t'}$ denotes rotation from $t-1$ to t obtained with q_{t-2}^{t} and q_{t-2}^{t-1}. Based on the two different measurements of the current rotation, namely $q_{t-1}^{t'}$ and q_{t-1}^{t}, the final rotation, denoted Q_{t-1}^{t}, is calculated through spherical linear interpolation (Slerp) [13].

$$Q_{t-1}^{t} = q_{t-1}^{t}((q_{t-1}^{t})^{-1} q_{t-1}^{t'})^{0.5} \tag{5}$$

SOFT algorithm employs Slerp for rotation estimation. Fig. 1 shows the rotation error for eight point, five point and the SOFT algorithms evaluated on KITTI dataset. This experiment was performed employing eleven different trajectories. The results represent mean rotational error values gained for each trajectory following the same calculation approach as is given in [14].

B. Translation estimation

In order to estimate the translation in metric scale, a stereo method has to be employed. Having determined all the correspondences between the features, two 3D point clouds can be reconstructed - one for the previous, and one for the current view. However, the motion estimation obtained by minimizing image reprojection error is more accurate than the one obtained by minimizing euclidian error in 3D space [5]. This is due to the fact that feature position uncertainties are not taken into account during minimization. In Euclidian space error covariances are highly anisotropic, whereas in the image space errors are more uniformly distributed between dimensions [6].

Therefore, 3D point cloud is triangulated from the previous view, and then reprojected onto image plane of the current view through

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \pi(X; R, t) = \begin{bmatrix} f & 0 & c_u \\ 0 & f & c_v \\ 0 & 0 & 0 \end{bmatrix} [R|t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{6}$$

where $[u\ v\ 1]^T$ are homogeneous image coordinates, π is reprojection function, f is the focal length, $[c_u\, c_v]^T$ is the image principal point, $[R|t]$ are rotation and translation between two views, and $[X\ Y\ Z\ 1]$ are homogeneous coordinates of the 3D point in the world. Translation is then calculated by iteratively minimizing

$$\sum_{i=1}^{n} \left\| x_i^l - \pi^l(X; R, t) \right\|^2 + \left\| x_i^r - \pi^r(X; R, t) \right\|^2 \tag{7}$$

with respect to translation only. Fig. 2 shows comparison between methods with and without enabled feature tracking, and the final SOFT algorithm.

V. IMU FUSION

Since five point algorithm is computationally complex and only used to improve rotational accuracy, we explore the possibility of increasing the rotation accuracy by using rotation estimation already available from the 3-point scheme fused with IMU measurements. Rotation is tracked in a Kalman filter, with the state vector given by

$$S = [q \quad b]^T, \tag{8}$$

where q is quaternion representing rotation from the body frame to the world frame, and b is gyroscope bias. Gyroscope measurement is modeled by

$$\omega_m = \omega + b + n, \tag{9}$$

where ω is true angular velocity, b is bias and n is measurement noise. Quaternion is propagated through zeroth order quaternion integrator [15]

$$q_t = q_{t-1}(I_{4x4} + \frac{\Delta t}{2}\Omega(\omega)) \tag{10}$$

where

$$\Omega(\omega) = \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix}. \tag{11}$$

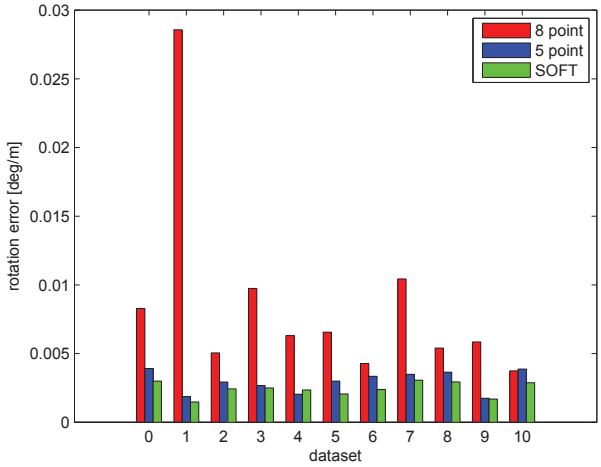

Fig. 1. Relative rotation error in the training dataset

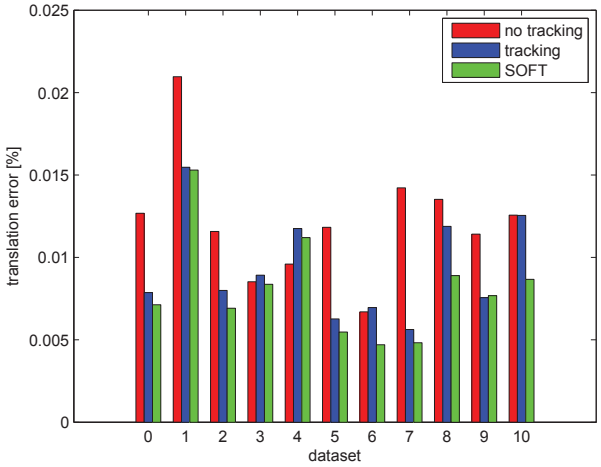

Fig. 2. Relative translation error in the training dataset

Bias is modeled as a random walk process. It changes slowly through the time, and if it is correctly estimated, rotation between two successive frames can be determined accurately. The prediction step is performed every time a new IMU reading is obtained, while the update step executes every time a new image is captured. Rotation resulting from visual odometry is used as a measurement for filter update, so the filter state is measured directly.

In this approach, rotation accuracy is increased with employment of the Kalman filter. Rotation from the gyroscopes is used for outlier rejection. After the inliers are determined, the three point method (P3P) combined with RANSAC, is used for calculating both rotation and translation. Then, rotation is delivered to the Kalman filter as a new measurement, and after updating the filter, it is replaced with the filtered rotation. Overview of the method is shown in Fig. 3. Hardware is implemented with two PointGrey USB 2.0 Firefly MV cameras and Invensense MPU-6050 IMU mounted on a custom rig (Fig. 4). IMU operates at 500 Hz, while its clock divided by 25 is used as a signal for triggering cameras at 20 Hz synchronous to IMU readings (Fig. 5). This low-cost stereo-IMU system proved to be efficient enough to obtain satisfactory results in the experiments we conducted. The IMU based concept provides significant improvement over the base method, which is especially manifested in urban environments. During the forward driving, distant features are visible and can be tracked for a longer period of time. The rotation has no drift and the gyroscope bias is well estimated. Interestingly, while taking a sharp turn at the crossroad, a scene has suddenly changed, hence no features survived the turn and therefore the drift in the pure image based estimation has arisen. However, rotation estimation in the Kalman filter is driven by gyroscopes and is not affected by occasional measurement errors emerged from sharp turns. Also, while processing two consecutive image pairs, relative rotation between them is a priori known. Therefore, this rotation is used in the matching phase to predict new feature positions in the image. This prediction significantly reduces number of outliers and also narrows the search space, which results in speeding up the performance.

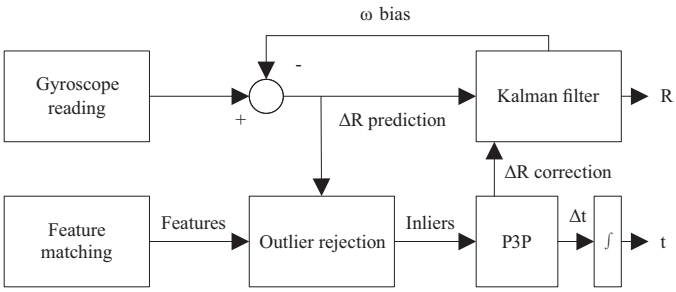

Fig. 3. Overview of the IMU fusion algorithm

Fig. 4. Stereo camera with IMU

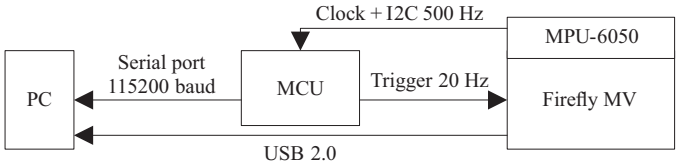

Fig. 5. Hardware synchronization

TABLE I. KITTI LEADERBOARD

Method	Transl.	Rotation	Runtime	Environment
SOFT [this]	**1.03 %**	**0.0029 [deg/m]**	0.1 s	2 cores @ 2.5 Ghz
cv4xv1-sc [16]	1.09 %	0.0029 [deg/m]	0.145 s	GPU @ 3.5 Ghz
MFI [6]	1.30 %	0.0030 [deg/m]	0.1 s	1 core @ 2.2 Ghz
TLBBA [17]	1.36 %	0.0038 [deg/m]	0.1 s	1 Core @2.8 GHz
2FO-CC [8]	1.37 %	0.0035 [deg/m]	0.1 s	1 core @ 3.0 Ghz

VI. Experimental results

SOFT is evaluated on the KITTI benchmark, which provides very thorough analysis of rotational and translational error in urban, rural and highway scenarios. The proposed algorithm shows an average translational error of 1.03% of the traveled distance and 0.0029 deg/m rotational error, outperforming all other validated methods. Table I shows current top five stereo odometry scores from KITTI leaderboard. Two of the estimated paths are shown in Fig. 6 and Fig. 7. More detailed results can be found on KITTI leaderboard. These real world scenarios appears to be very demanding, since the best methods from synthetic datasets perform below average when moved to the real world, which is our primary goal. Complete solution is implemented in C++ and optimized for real-time performance. Code is executed in two separate threads. Optimization by threading is chosen due to suitability for embedded systems. While GPU or SSE optimizations are still limited to certain types of computers, wide range of CPUs have multiple cores. The IMU based version of the system is implemented and tested on ODROID U3 quad-core ARM based embedded computer.

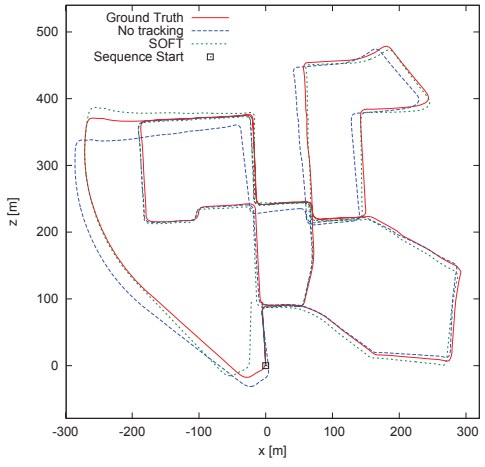

Fig. 6. Reconstructed path of the KITTI00 dataset

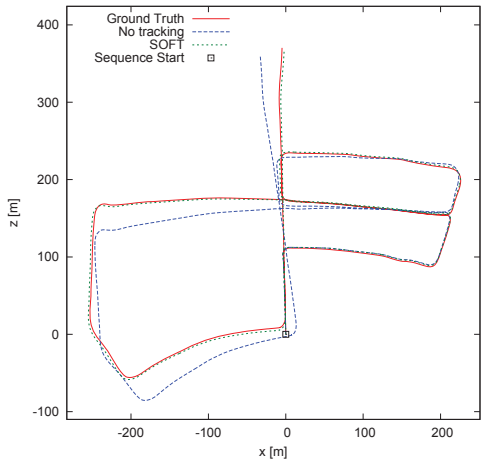

Fig. 7. Reconstructed path of the KITTI05 dataset

TABLE II. PROCESSING TIME

Method	Image processing	Rotation	Translation	Total
5 point	35 ms	60 ms	5 ms	100 ms
IMU	35 ms	0	5 ms	40 ms

Unfortunately, the KITTI benchmark, along with several other related benchmarks, does not contain IMU data. Therefore, the IMU based solution is tested on a self-made dataset. However, associated groundtruth could not be acquired, and the only measure of error is the distance from the groundtruth endpoint, which is the same as the starting point. Fig. 8 shows estimated path overlayed over the satellite map of the city block. The red line corresponds to estimation provided without the utilization of an IMU unit (error 2.47%), while the blue line corresponds to the estimated path obtained by the IMU-based solution (error 1.39%). The groundtruth would match the closest corresponding road path, but as such is not specifically marked in the figure. Table II shows the approximate processing times for proposed algorithms. The five point based algorithm is executed on two cores of an Intel i7 on 2.2 GHz and the time corresponds to the KITTI dataset with images of the size 1248x376 pixels, while the IMU based algorithm is performed on ODROID-U3 on images of the size 752x480 pixels.

Fig. 8. Path estimated from self recorded dataset

VII. CONCLUSION

In this paper, we have presented a novel algorithm for stereo visual odometry that reduces the drift based on careful selection and tracking of stable features that we have dubbed SOFT. The rotation is estimated with the five point method, while the translation is obtained with the three point method. The proposed approach is proven to be the best performing algorithm using the challenging KITTI vision benchmark dataset. In the future we intend to test different feature detectors/descriptors in conjunction with proposed algorithm. Also, we intend to investigate the efficiency of particular methods in urban, rural and highway scenarios.

ACKNOWLEDGMENT

This work has been partly supported by the European Regional Development Fund under the project *Advanced technologies in power plants and rail vehicles*.

REFERENCES

[1] D. Scaramuzza, and F. Fraundorfer, "Visual odometry [tutorial]," Robotics & Automation Magazine, 18(4):8092, 2011.

[2] J. Engel, J. Sturm, and D. Cremers, "Semi-Dense Visual Odometry for a Monocular Camera," in IEEE International Conference on Computer Vision (ICCV), 2013.

[3] C. Forster, M. Pizzoli, and D. Scaramuzza, "Fast Semi-direct Monocular Visual Odometry," in IEEE International Conference on Robotics and Automation, 2014.

[4] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets Robotics: The KITTI Dataset," in International Journal of Robotics Research (IJRR), 2013.

[5] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in Proc. Int. Conf. Computer Vision and Pattern Recognition, 2004, pp. 652-659.

[6] H. Badino, A. Yamamoto, and T. Kanade, "Visual Odometry by Multi-frame Feature Integration," International Workshop on Computer Vision for Autonomous Driving @ ICCV, December, 2013.

[7] W. Lu, Z. Xiang, and J. Liu, High-performance visual odometry with two-stage local binocular ba and gpu, in Intelligent Vehicles Symposium, 2013, pp. 11071112.

[8] I.Krešo, and S.Šegvić, "Improving the Egomotion Estimation by Correcting the Calibration Bias," VISAPP, 2015.

[9] A. Geiger, J. Ziegler, and C. Stiller, "Stereoscan: Dense 3d reconstruction in real-time," in IEEE Intelligent Vehicles Symposium, 2011, pp. 963–968.

[10] B. Kitt, A. Geiger, H. Lategahn, "Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme," in Intelligent Vehicles Symposium, 2010, pp. 486-492.

[11] J. Shi and C. Tomasi, "Good Features to Track," 1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94), 1994, pp. 593–600.

[12] D. Nistér, "An Efficient Solution to the Five-Point Relative Pose Problem," in IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004, pp. 756–777.

[13] E. B. Dam, M, Koch, and Martin Lillholm, "Quaternions, interpolation and animation," Technical Report DIKU-TR-98/5, 1998.

[14] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

[15] N. Trawny, and S. I. Roumeliotis, "Indirect Kalman Filter for 3D Attitude Estimation," Technical Report Number 2005-002, 2005.

[16] M. Persson, T. Piccini, R.Mester and M. Felsberg, "Robust Stereo Visual Odometry from Monocular Techniques," in IEEE Intelligent Vehicles Symposium, 2015.

[17] W. Lu, Z. Xiang and J. Liu, "High-performance visual odometry with two stage local binocular BA and GPU," in IEEE Intelligent Vehicles Symposium (IV), 2013, pp. 1107–1112.