Monocular Visual Odometry With Cyclic Estimation

Fabio Irigon Pereira
PGMICRO
Universidade Federal do Rio Grande do Sul
Av. O. Aranha, 103 - POA - BRASIL
fabio.pereira@ufrgs.br

Gustavo Ilha
Joel Luft
PPGE- UFRGS
joel.luft@ufrgs.br
gustavo.ilha@ufrgs.br

Marcelo Negreiros
and Altamiro Susin
DELET - UFRGS
negreiro@eletro.ufrgs.br
altamiro.susin@ufrgs.br

***Abstract*—Monocular Visual Odometry (MVO) estimates the camera position and orientation, based on images generated by a single camera. In this paper a new sparse MVO system for camera equipped vehicles is proposed. Three view cyclic Perspective-n-Point with adaptive threshold is used for camera pose estimation, perspective image transformations are used to improve tracking, and a multi-attribute cost function selects ground features for scale recovery. Results using the KITTI dataset show that the proposed system achieves 1.29% average translation error and average rotation precision of 0.0029 degrees per meter.**

I. Introduction

Monocular Visual Odometry (MVO) or camera tracking is the problem of estimating the camera position and orientation for each provided image. This is an area of paramount importance in computer vision with applications in autonomous vehicle navigation, camera equipped drone control, driver assistance, augmented reality systems and special effects for movie industry among others.

Camera pose estimation is tightly coupled to the perceived scene structure. Sparse systems select discrete image regions, named features, search those regions in subsequent images, and use relative image displacement to estimate the 3D positions for those features. Camera positions and orientations are calculated to match the estimated feature 3D positions, to the found pixel coordinates on each image plane.

The scale of both scene structure and camera translation is an inherently absent information in MVO systems. Scene scale must be recovered using specific methods such as previously known image characteristics (absolute size or distance of recognizable items) or external sensors (GPS, Li-Dar, odometer, IMU etc).

This paper proposes a new sparse MVO system that is divided in three main modules: graphical, geometric and scale recovery. The proposed system introduces modified algorithms for feature tracking, camera pose estimation and scale adjustment.

The image processing unit uses Shi-Tomasi [1] and Lukas-Kanade [2] algorithms to detect and track features with the use of perspective image projections to improve feature tracking on the ground plane, and avoid feature tracking errors along epipolar lines.

The geometric module uses the features matched in the last three images, to perform a local bundle-adjustment. Two view triangulation is used to determine features distance to the camera, and Perspective-n-Point (PnP) used to refine camera pose. The module uses a cyclic algorithm to select pairs of reference images in an alternating order, and calculate relative pose and feature depth, until reprojection error falls below an adaptive threshold.

Finally scale is inferred from the known camera distance to the pitch corrected ground plane. The method used selects a single ground feature in each image, based on the feature position, image gradient along the epipolar line, and depth difference from the estimated ground position.

The main contributions of this work are:

- The use of perspective image transformations to improve tracking, and avoid mismatch due to similar features along epipolar lines.
- Camera pose estimation through a Cyclic 3-view Bundle Adjustment method with variable re-projection error threshold.
- Scale estimation based on multi-attribute ground point selection, and pitch-sensitive ground plane.

Results on the KITTI dataset [3] show average translation errors of 1.29% and rotation errors of 0.0029 degrees per meter, outperforming state-of-the-art published monocular odometry systems. This paper is outlined as follows: section II makes a brief review of related works, notation is defined on section III the system is explained in section IV, results using the KITTI dataset [3] are summarized in section V and discussed in section VI and finally section VII concludes this work.

II. Related Work

This section makes a brief overview of important works in this area. One of the first successful monocular localization and mapping techniques is MonoSLAM [4], where an extended Kalman filter is used to update a probabilistic 3D scene map. Filtering approach was a popular technique at the time, shared by other implementations [5], [6]. The milestone work PTAM from Klein and Murray [7] was the first work to split image tracking and geometrical pose estimation in separate threads and implement a keyframe based framework that influenced many solutions [8], [9] and is still a reference for present implementations [10], [11].

Dense or semi-dense propositions [12] use the whole image, acting directly on pixel intensities warping the image to

produce a dense 3D structure of the scene. Those methods are very interesting to build a detailed scene map, as depth is estimated for every pixel presenting relevant image gradient, when used for camera tracking however, tend to be more susceptible to outliers in non-static scenes.

Sparse systems on the other hand, must select traceable image regions with methods such as Shi-Tomasi [1] or FAST [13] corner detectors, and use either direct pixel gradients in Lucas-Kanade [14] variations, or feature descriptors like SIFT [15], and ORB [16], that try to cope with scale and perspective variation.

A common approach in sparse systems [7], [10], [17] is the use of keyframes, where new features are only detected in a subset of frames used as reference for pose prediction.

Camera pose can be directly estimated by reprojection error minimization in PNP based algorithms [18], [19] or extracted from fundamental matrix [20] [21].

Full Bundle Adjustment (BA) refines simultaneously the camera pose and feature positions, while structure only BA and motion only BA, limit the objective to either feature depth or camera pose respectively. As full BA for a large set of images is computationally demanding, most solutions tend to use a subset of the images in what is called local BA. ORB-SLAM [22] from Artal Et al. use a covisibility graph to select frames for local BA, while the most common approach is to use the last frames as used by [9] and [23], and this work.

Two recent monocular works achieved notable results when tested with the database used in this work. The work proposed in [10], used a local bundle adjustment with the last 10 frames, and scale recovered from multiple cues (ground plane and recognizable objects) to build a system with great accuracy. The work [23] used the five point algorithm [21] for essential matrix estimation, and a probabilistic multiple view triangulation for scene structure calculation. This work, instead of using several frames for pose estimation or triangulation, uses only the last three frames.

The interesting work of Fanani [24], jointly minimizes image disparity (i.e. pixel intensity discrepancy) and geometrical reprojection error to refine camera pose estimation.

The KITTI dataset [3] is used to evaluate the proposed system. A collection of 22 image sequences, made by a camera equipped vehicle in trajectories recorded by a high precision GPS. Images are recorded at 10 frames per second, and car speed varies from 0 to about 90 Km/h. Steep turns, high inter-frame displacement in non-static scenes (where cars, bicycles and trees move), make this a highly challenging application.

III. Notation

In this work the notation used in Forster et. al [8] will be adopted. A point in the k_{th} frame, $\mathbf{p} = (x, y, z)^T$ is mapped to an image coordinate $\mathbf{u} = (u, v)^T$ by the camera projection model $\pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$ by $\mathbf{u} = \pi({}_k\mathbf{p})$. The point's 3D position can be recovered from the projected position and the associated depth $d_\mathbf{u}$ with ${}_k\mathbf{p} = \pi^{-1}(\mathbf{u}, d_\mathbf{u})$. The set of all point coordinates in the image k, is noted as $\mathbf{U}_k$.

The projection model π is dependent of the intrinsic camera parameters, determined from calibration. The camera pose is a 3x4 matrix characterizing the rigid-body transformation $\mathbf{T}_{k,w} \in SE(3)$. A point from world coordinate frame is mapped to the camera frame of reference by ${}_k\mathbf{p} = \mathbf{T}_{k,w} \cdot {}_w\mathbf{p}$. Projection error minimization for pose estimation is a transformation $\mathbf{T}_{k,w}$ that minimizes the difference between viewed features, and the calculated projection on the image plane as shown in equation 1.

$$\mathbf{T}_{k,w} = \arg\min_{\mathbf{T}_{(k,w)}} \sum_i ||\mathbf{u}_i - \pi(\mathbf{T}_{(k,w)} \cdot {}_w\mathbf{p}_i)||^2 \qquad (1)$$

IV. System Architecture

In this section the architecture of a new sparse MVO system is described. The system is composed by three main modules: graphical unit, geometrical unit and scale unit, as shown in figure 1. The modules are described in the following subsections.

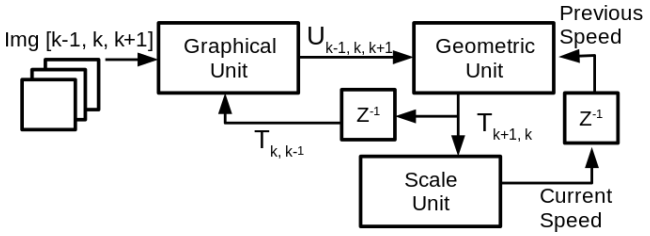


Fig. 1. System architecture

A. Graphical Unit

The graphical unit detects new traceable image regions, and locates them in subsequent frames. Tracking is performed with Shi-Tomasi [1] feature detector, and Lucas-Kanade [2] is used to search for correspondences.

The module receives three images and the last relative camera displacement $T_{k,k-1}$ as input. Corners are vectors of (u, y) image coordinates, detected in the central frame and tracked in the previous and next frames, denoted by subscripts k, k-1 and k+1, features detected in the past are not used. The output of the module is the set of all pixel coordinates of tracked features in the three images, grouped into the matrix $\mathbf{U}_{k-1,k,k+1}$.

Features on the ground are especially important for scale estimation, and especially difficult to track due to low gradients on asphalt and high pixel wise distance for high speeds. The ground plane however, remains in a virtually constant position and can be foreseen as shown in figures 2a, 2b and 2c, where the box show a feature on the ground.

Four constant image coordinate pairs, in the bottom-center image area, are projected in the previous frame, using the relative camera displacement $T_{k,k-1}$ and depth from the constant estimated ground plane. No pixel intensities are used in this operation, only geometric point projection. With the initial coordinates and the projections, a perspective transformation is calculated and applied to the second image as shown in

(a) img 0

(b) img1

(c) img 0 warped to match ground

(d) img 0 warped to match close features

Fig. 2. Perspective warping of figures: figs (a) and (b) show the original images, fig (c) is figure (a) warped for ground matching (as shown by the box) and fig (d) is fig (a) for close object matching (as shown by the circle)

figure 2c. It is possible to note that pixels on the ground from the warped image 0 (fig 2c) are similar to the ground pixels on image 1 (fig 2b). Features are tracked in the warped image, and then back in the original to avoid outliers.

A problem that occurs especially when the car drives in higher speeds, are similar patterns along epipolar lines. On figure 2a the guard rails on both sides of the road are very similar and thus difficult to differentiate for correct matching. In stereo cases, the stereo calculated depth can be used to help choosing the correct match, but on monocular situations, the correct correspondence is very difficult to find, as different estimated depths provide different matches with equivalent reprojection errors.

To mitigate this problem, in a similar way to ground plane estimation, four constant seed points, but this time around the image center are assigned a constant depth not too large (in the present case 15 meters), and projection is calculated. The image is warped simulating a situation in which every pixel had the same depth, as if belonging to a plane parallel to the image plane, as can be seen on figures 2a and 2d. We can see that close objects, as marked by the circle in fig 2d, are projected to their location on image 1 (fig 2b), while points distant to the camera bear larger discrepancy. By tracking

function CYC_BA($T_{k,k-1}, U_{k-1,k,k+1}$)
  num_cyc $\leftarrow$ 0,
  rep $\leftarrow$ max_rep
  threshold $\leftarrow$ init_threshold
  $T_{k-1,k} \leftarrow Invert(T_{k,k-1})$
  while rep>threshold & num_cyc<max_cyc **do**
    $d_{k-1} \leftarrow triangulate(T_{k-1,k}, U_k, U_{k-1})$
    $T_{k+1,k-1} \leftarrow SolvePnP(U_{k-1}, d_{k-1}, U_{k+1})$
    $d_{k+1} \leftarrow triangulate(T_{k+1,k-1}, U_{k-1}, U_{k+1})$
    $T_{k,k+1} \leftarrow SolvePnP(U_{k+1}, d_{k+1}, U_k)$
    $d_k \leftarrow triangulate(T_{k,k+1}, U_{k+1}, U_k)$
    $T_{k-1,k} \leftarrow SolvePnP(U_k, d_k, U_{k-1})$
    $rep \leftarrow GetReproj(U_{k-1}, U_k, d_k, T_{k-1,k})$
    num_cyc, threshold $\leftarrow$ num_cyc+1, threshold + δ
  end while
  return $Invert(T_{k,k+1})$
end function

Fig. 3. Cyclic BA algorithm

features in two images, projected with different depths on the seed points, the guard-rails on the road will be matched to different positions, and can be classified as dubious matches and be discarded.

B. Geometric Unit

This section describes the method used to estimate the new camera position, using point correspondences in the last three images. The module receives as inputs the (u, v) image coordinates of corresponding matched image regions, and uses last camera displacement $T_{k,k-1}$ and current speed as shown in fig 1 to calculate the new camera displacement $T_{k+1,k}$. The method is outlined in algorithm of figure 3 and explained below.

In the algorithm of figure 3, features in pairs of images are used in a cyclic way, i.e. first correspondences in images $k, k-1$ are used, then $k-1, k+1$ and finally $k+1, k$. Triangulation [25] is used to calculate depth, - distance from the feature to the camera. Perspective-n-Point is used for relative pose estimation. SolvePnP uses Levenberg-Marquardt [26] nonlinear optimization in a RANSAC approach to estimate camera displacement. At each iteration a subset of features is randomly selected and camera pose is estimated based on the subset. The median error is calculated for all features and the solution with the lowest error is elected.

Small errors in camera pose estimation, cause larger reprojection errors in close features due to large displacement along an incorrect epipolar line, and RANSAC can select only far points which are less sensitive to camera translation. To avoid this situation, the system divides the points in two sets with the same number of points, based on their distance to the camera. The median is calculated for close points and far points and the average of medians is used as the fit criterion for RANSAC.

When system is initialized the first relative camera pose estimation must be performed without previous knowledge of motion. Three initial guesses of forward motion are made, with

three different speeds. The solution with lowest reprojection error is selected and scale of camera translation is set to the first recovered depth to ground estimation, with the method outlined in the next section, and used as a first approximation of translation scale.

C. Scale Extraction

For the current application of a camera equipped vehicle, the camera distance to the ground is nearly constant, and can be used to estimate the scene scale as will be discussed in this section.

The absolute distance of a point on the ground can be calculated by triangle similarity, ignoring usually negligible rotations around optical axis, as shown in figure 4 and equation 2, and by comparing the depth from triangulation and the expected ground depth, the whole scene scale can be approximated. In the figure, f is the focal length, V_h is the position of the horizon in the image, v_o is the vertical position of the feature, and h is the constant camera height.

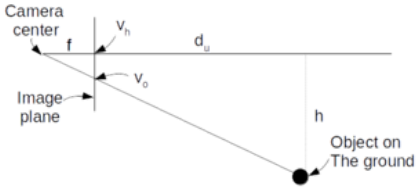


Fig. 4. Absolute scale

$$d_u = \frac{fh}{v_h - v_o} \tag{2}$$

A good ground point, should be in the center bottom area of the image (see the line in figure 2a), should present a good image gradient along the epipolar line to enhance matching precision and is expected to be close to the estimated ground.

A multi-objective cost function based on these three characteristics, is applied to the tracked points and the point with lowest score is selected. The positional cost penalizes points far from the bottom center of the image with the equation 3 where q_u parameter is used to widen the ground area and (gnd_v, gnd_u) is a constant point in the center lower area of the image.

$$l_i = \sqrt{(v - gnd_v)^2 + (u - gnd_u)^2 / q_u} \tag{3}$$

Traceability quality is measured by averaging the pixel intensity difference in two image areas dislocated along the epipolar lines. The first image area is a box centered in the projected feature location. The second box center is calculated by applying a δz to the feature depth, projecting the feature and normalizing the vector from the first box position to the new projection, in order to have an Euclidean distance of one pixel. Both positions present sub-pixel accuracy and interpolation is used to find pixel intensities.

Finally features above the ground are avoided by setting a higher cost to feature with calculated depths far from the expected ground distance.

A smooth transition function of the form of equation 4 is used to scale and smooth the costs of points. Costs are bounded in the $[0, 1]$ interval and a smooth threshold is used to distinguish good ground point from low confidence ones, by adjusting parameters m the mean and a the aperture, as shown in figure 5.

$$s = \frac{1}{1 + e^{\frac{x-m}{a}}} \tag{4}$$

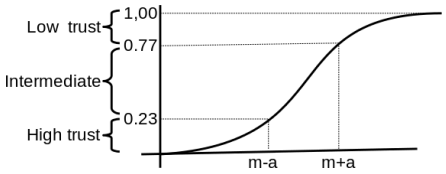


Fig. 5. Sigmoid function

Camera is not perfectly horizontally oriented, and different trajectories may present a slightly different pitch angle (camera angle with respect to the ground plane). The average pitch angle is estimated by the ratio of the forward translation and vertical translation as shown in equation 5, where f is the focal length, t_y and t_z are up and forward translation, c_y is the vertical camera center, λ is a smooth parameter and v_h is the vertical position of the horizon. If the camera is pointed slightly upward, vertical translation will be negative in average, while for the opposite case positive vertical translation is expected.

$$v_h = \lambda_v \cdot (\frac{f t_y}{t_z} + c_y) + (1 - \lambda_v) v_h \tag{5}$$

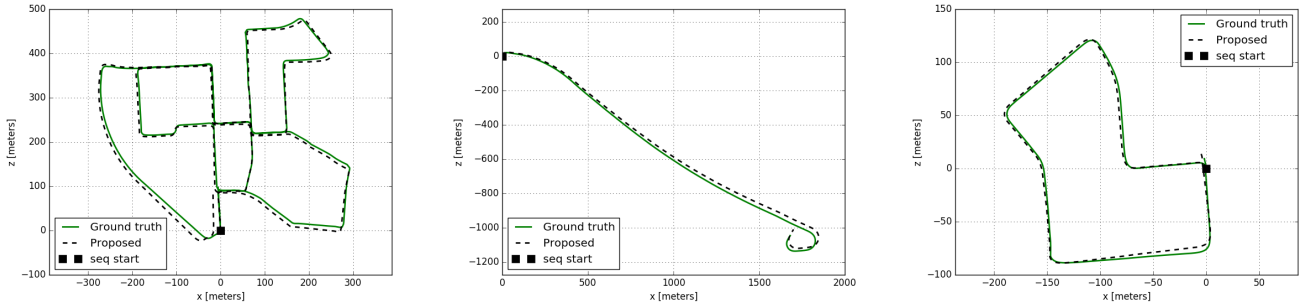
Next section presents the results of the proposed system.

V. Results

The system was implemented in Python, using SciPy [27] and OpenCV [28], and tested on the KITTI database [3]. This section presents the achieved results.

The following parameters were used: Minimum feature distance used was 15 pixels, Lucas-Kanade used 3 pyramid levels. The localization step used subsets of 10 features, and 55 repetitions. The median re-projection error is on the order of 0.15 pixels, and feature set ranges between 150 and 300 features per image set. The used camera height was 1.65m, provided by the database developers. Initial threshold for cyclic estimation was set to 0.125 pixels and $\delta = 0.05$. Parameters m and a used were: positional cost 80 and 40; tracking quality 8 and 4 after normalization by dividing SAD by the number of pixels in the patch; speed 0.4 and 0.2 meters per frame.

The single thread implementation was executed in an Intel i7-3770K CPU @ 3.5GHz with an average time per frame of approximately 0.17 seconds. To meet real time requirement,



(a) seq 00 (b) seq 01 (c) seq 07

Fig. 6. Estimated trajectories

TABLE I
PER TRAJECTORY TRANSLATION ERRORS IN [%] AND ROTATION ERRORS IN [DEG/M]

seq	num frames	translation [%]	rotation [deg/m]
00	4540	1.03	0.0030
01	1100	1.37	0.0023
02	4660	1.33	0.0038
03	800	0.87	0.0022
04	270	0.86	0.0024
05	2760	0.99	0.0037
06	1100	0.73	0.0026
07	1100	1.12	0.0057
08	4070	1.23	0.0028
09	1590	1.54	0.0028
10	1200	1.02	0.0024
Avg		**1.23**	**0.0028**

the worst case allowable time per frame should be below 0.1 second for the used frame rate of 10 images per second. A low level multi-threaded C/C++ implementation, aimed at meeting real time requirements for that platform is left as a future work.

The metrics used to evaluate the system proposed by the database developers are degrees per meter and translation percentage error. Trajectories are split in segments of 100 to 800 meters, and errors are averaged over all trajectory segments. Table I shows the results for the first 11 routes.

Figure 6 shows the car path in meters for sequences 0, 1 and 7, estimated path in black and ground truth in green, for a qualitative idea of estimation precision.

VI. ANALYSIS AND COMPARISON

This section carries out a brief analysis of the presented results and compares the proposed system to other implementations.

The database developers do not provide ground truth data for trajectories 11 to 21. Estimations can be uploaded to their web-site [29] and results can be compared. Table II compares the proposed system's results to be best qualified Monocular Visual Odometry published works.

TABLE II
RESULTS COMPARISON FOR TRAJECTORIES 11-22

Work	Transl [%]	Rot [deg/m]
VISO2-M [30]	11.9	0.0234
MLM-SFM [10]	2.54	0.0057
FTMVO [23]	2.24	0.0049
FVO [this work]	1.29	0.0031

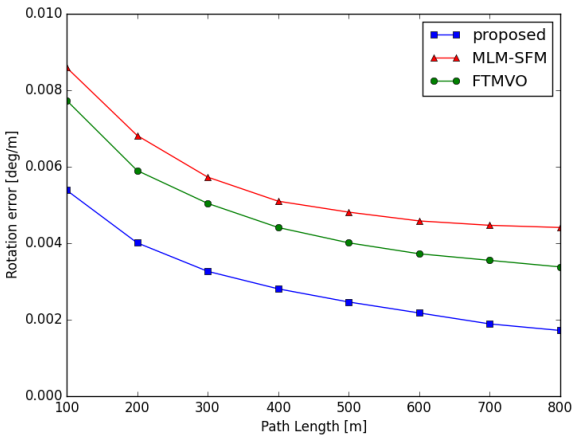


Fig. 7. Rotation Error × Distance

Figures 7 and 8 compare rotation errors for different traveled distances, and translation errors for different car speeds.

It is particularly interesting to analyze the translation error behavior with respect to car speed, as shown in figure 8. For lower speeds all three algorithms provide similar results, which is already interesting, considering that [10] uses multiple frames for pose estimation and [23] use multiple frames for depth calculation, while the proposed cyclic BA method uses only the three last images. Most of the average result improvement however, comes from the higher speed scenarios. In those cases, most solutions face the problems discussed in section IV-A, especially bad tracking due to similar matches along epipolar lines, and few correctly matched points on

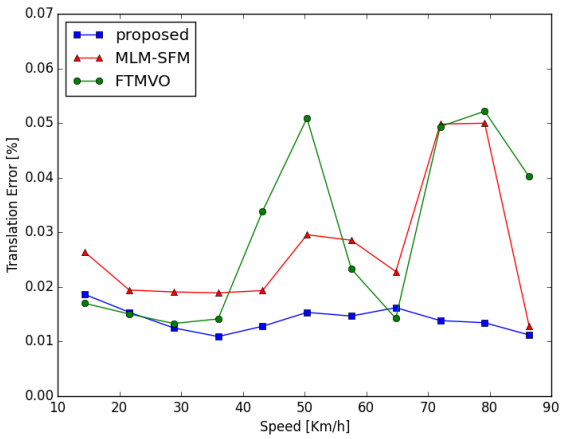


Fig. 8. Translation Error × Speed

the ground. The perspective image warping performed by the graphical unit is especially important in steep turns and high speed sequences both to accurately locate features on the ground and to eliminate possible track failures, improving robustness for those cases.

VII. Conclusion

This paper presented a monocular visual odometry system for camera equipped vehicles. The system used perspective image transformation to improve Lukas-Kanade tracking, and remove outliers. A cyclic PnP algorithm, with variable reprojection threshold was used to calculate camera pose with the three last images. Scale was recovered with a multi-attribute ground feature selector.

A python and OpenCV implementation tested on the KITTI dataset, achieved rotation and translation precision superior to other published works.

References

[1] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, Jun 1994, pp. 593–600.

[2] B. D. Lucas and T. 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*, ser. IJCAI'81. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981, pp. 674–679. [Online]. Available: http://dl.acm.org/citation.cfm?id=1623264.1623280

[3] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, June 2012, pp. 3354–3361.

[4] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[5] D. Chekhlov, M. Pupilli, W. Mayol-cuevas, and A. Calway, "Real-time and robust monocular slam using predictive multi-resolution descriptors," in *In 2nd International Symposium on Visual Computing*, 2006.

[6] X. Meng, F. Hong, and Y. Chen, "Monocular simultaneous localization and mapping with a modified covariance extended kalman filter," in *2009 IEEE International Conference on Intelligent Computing and Intelligent Systems*, vol. 2, Nov 2009, pp. 900–903.

[7] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, Nov 2007, pp. 225–234.

[8] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO : Fast Semi-Direct Monocular Visual Odometry," *IEEE International Conference on Robotics and Automation*, 2014.

[9] S. Song, M. Chandraker, and C. C. Guest, "Parallel, real-time monocular visual odometry," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4698–4705, 2013.

[10] ——, "High Accuracy Monocular SFM and Scale Correction for Autonomous Driving," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 4, pp. 730–743, 2016.

[11] M. Persson, T. Piccini, M. Felsberg, and R. Mester, "Robust stereo visual odometry from monocular techniques," *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, pp. 686–691, 2015.

[12] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1449–1456, 2013.

[13] E. Rosten and T. Drummond, *Machine Learning for High-Speed Corner Detection*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 430–443. [Online]. Available: http://dx.doi.org/10.1007/11744023_34

[14] J. yves Bouguet, "Pyramidal implementation of the lucas kanade feature tracker," *Intel Corporation, Microprocessor Research Labs*, 2000.

[15] D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 2, 1999, pp. 1150–1157 vol.2.

[16] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, Nov 2011, pp. 2564–2571.

[17] C. Forster, M. Faessler, F. Fontana, M. Werlberger, and D. Scaramuzza, "Continuous On-Board Monocular-Vision based Elevation Mapping Applied to Autonomous Landing of Micro Aerial Vehicles," *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[18] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, "Complete solution classification for the perspective-three-point problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 8, pp. 930–943, Aug 2003.

[19] V. Lepetit, F.Moreno-Noguer, and P.Fua, "Epnp: An accurate o(n) solution to the pnp problem," *International Journal Computer Vision*, vol. 81, no. 2, 2009.

[20] M. H. Mirabdollah and B. Mertsching, *On the Second Order Statistics of Essential Matrix Elements*. Cham: Springer International Publishing, 2014, pp. 547–557. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-11752-2_45

[21] D. Nister, "An efficient solution to the five-point relative pose problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 756–770, June 2004.

[22] R. Mur-Artal, J. M. M. Montiel, and J. D. Tards, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, Oct 2015.

[23] U. o. P. Mirabollah M., Mertshching B. (GET Lab, "Fast Techniques for Monocular Visual Odometry," in *GCPR 2015*, 2015, pp. 297–307.

[24] N. Fanani, M. Ochs, H. Bradler, and R. Mester, "Keypoint Trajectory Estimation Using Propagation Based Tracking," no. Iv, 2016.

[25] R. I. Hartley and P. Sturm, "Triangulation," *Computer Vision and Image Understanding*, vol. 68, no. 2, pp. 146–157, 1997. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1077314297905476

[26] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963. [Online]. Available: http://dx.doi.org/10.1137/0111030

[27] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001–, [Online; accessed 2016-07-11]. [Online]. Available: http://www.scipy.org/

[28] G. Bradski, "Opencv," *Dr. Dobb's Journal of Software Tools*, 2000.

[29] A. Geiger, P. Lenz, and R. Urtasun, "The kitti vision benchmark suite," http://www.cvlibs.net/datasets/kitti/eval_odometry.php, accessed: 2017-01-20.

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