Infinite Visual Odometry

BMVC 2016 Submission # 250

Abstract

In this work we revisit the problem of visual odometry. Visual odometry is the process of estimating the motion of the camera by examining the changes that the motion induces on the images made by it. The approach we propose exploits a scene structure typical for that seen by a moving car and is suitable for use in either the stereo or the monocular setting. We recover the rotation and the translation separately, thus dealing with two separate, smaller problems. The rotation is estimated by means of the infinite homography. The rotation estimation algorithm operates on distant image points using the 3-D to partition them into the distant and the near-by ones. We start with an initial estimate and then refine it using an iterative procedure. After the rotation is compensated for, the translation is found by means of the 1-point algorithm in the stereo setting and epipole computation for pure translational motion in the monocular setting. We evaluate our algorithm on the KITTI dataset [5].

1 Introduction

Visual odometry refers to the problem of recovering camera motion based on the images taken by it. This problem naturally occurs in robotics, wearable computing, augmented reality and automotive.

Wheel odometry, recovers the motion of the vehicle by examining and integrating the wheel turns over time. In a similar manner, visual odometry operates by estimating relative motion of the camera between subsequent images by observing changes in them. Later, these estimates are combined into a single trajectory. Just as in wheel odometry, visual odometry is subject to error accumulation over time. Contrary to wheel odometry, visual odometry is not affected by wheel slip in a rough terrain. Visual odometry is able to produce motion estimates with errors that are lower than those of the wheel odometry. Another advantage of visual odometry is that cameras are low cost and low weight sensors. All these make visual odometry a viable supplement to other motion recover methods such as global positioning systems (GPS) and inertial measurement units (IMUs).

Visual odometry becomes a harder problem as the amount of detail in the images diminishes. The images should have sufficient overlap and the scene needs to be illuminated. In the stereo setup, the scene must be static or the images taken at the same time. Also, the video processing incurs a computational burden.

Visual odometry is an active fields of research with a large amount of published work. We review only the most pertinent works. [23] provides a more complete survey.

Similar to [21] we partition visual odometry algorithms by four traits:

1. Feature-based vs direct

2. Global vs local

3. Filter based vs bundle adjustment based

4. Monocular vs stereo

Visual odometry algorithms use large number of corner detectors (e.g., Moravec [17], Harris [7], Shi-Tomasi [24], Fast [22]) and blob detectors (e.g., SIFT [15], SURF [3]). Corners are faster to compute and usually are better localized, while blobs are more robust to scale change. The choice of a specific feature point depends mainly on the images at hand. Motion estimation results for different feature points are presented in [6]. In this work we choose Harris [7] corners, but this choice is not crucial. We view the feature point choice as a parameter, which needs to be determined from the data (e.g., by cross-validation).

The features are either tracked [10] or matched [4] (i.e., freshly detected in each new frame) between subsequent images. While the early works chose to track features, most of the current works detect and match them. The output of this stage are pairs of the image features, which are the projections of the same 3-D point.

Matched features are used as an input for a motion estimation procedure. Whether the features are specified in 2-D or 3-D, the estimation procedures, may be classified into 3-D-to-3-D [16], 3-D-to-2-D [4] and 2-D-to-2-D [19]. Most of the early works were of the 3-D-to-3-D type. More recent works [19] claim that this approach is inferior to the latter two. Popular techniques that participate in most algorithms in some way are essential matrix estimation and (possibly) its subsequent decomposition [19], perspective 3-point algorithm [13], and re-projection error minimization [4].

Global methods [1], [18] keep the map of the environment and make sure that motion estimates are globally consistent with this map, while local methods do not. Some local methods [2] also keep track of a (local) map, but the underlying philosophy is different: global vs local. Global methods usually more accurate since they make use of a vast amount of information (which, of course, comes at a computational price). Note that accuracy does not imply robustness, since outliers that made their way into the map may greatly skew subsequent pose estimates.

Methods that explicitly model system state uncertainty tend to use filtering mathematical machinery, e.g., [14], [20], [12]. Another alternative to maintain map/pose estimate consistency is to use the bundle adjustment approach [26]. Monocular systems [25] make use of a single camera, while stereo systems [4] rely on a calibrated stereo rig. In the monocular setup the translation of the camera may only be estimated up to scale, while in stereo all six motion parameters may be recovered. An additional advantage of the stereo setup is that more information is available at each step, which may be one of the reasons why stereo algorithms perform better.

1.1 Our method

In this work we present a novel algorithm for camera motion estimation. The novelty of the algorithm is in camera rotation estimation procedure. We rely on the fact that for scene points that are infinitely far from the camera, the motion of the projected (image) points may be described by an homography (the infinite homography). For distant points this assumption is nearly true. Our algorithm starts by partitioning the scene points into two sets: distant and near-by. Then, camera rotation is estimated from the distant points and, subsequently, the translation is recovered from the near-by points.

We present two versions of the algorithm: one for the monocular and the other for the stereo settings. These versions differ in the way we partition points into the distant and the near-by ones and in the way the algorithms estimate translation.

With respect to the classification of the visual odometry methods given in the introduction, our work is local, feature based, stereo odometry. We do not use bundle adjustment, however the results of our algorithm may be subsequently improved with some form of bundle adjustment.

The outline of the our method:

1. Feature detection. We use Harris [7] corners.
2. Feature matching. The matching is done both across the stereo pair images as well as previous vs. current pair. We enforce epipolar constraint, chierality and use circle heuristics similar to [4] to reject outliers.
3. Partition the scene points into two sets: distant and near-by.
4. Estimate the rotation of the camera from the distant points.
5. Estimate the translation of the camera from the near-by points.

We choose the work [4] as our baseline (our implementation of their work). The results in the Section 4 show that on the KITTI dataset our rotation estimation method outperforms the baseline.

2 Preliminaries and Notation

2.1 Image Point Mapping Related to Camera Motion

Suppose the camera matrices are those of a calibrated stereo rig P and P′ with the world origin at the first camera

$$\mathrm{P} = \mathrm{K}[\mathrm{I} \mid 0], \quad \mathrm{P}' = \mathrm{K}'[\mathrm{R} \mid \mathbf{t}]. \tag{1}$$

Consider the projections of a 3D point $\mathbf{X} = (X,Y,Z,1)^T$ into the image planes of both views:

$$\mathbf{x} = \mathrm{P}\mathbf{X}, \quad \mathbf{x}' = \mathrm{P}'\mathbf{X}. \tag{2}$$

If the image point is normalized as $\mathbf{x} = (x,y,1)^T$ then

$$\mathbf{x}Z = \mathrm{P}\mathbf{X} = \mathrm{K}[\mathrm{I} \mid 0]\mathbf{X} = \mathrm{K}(X,Y,Z)^T.$$

It follows that $(X,Y,Z)^T = \mathrm{K}^{-1}\mathbf{x}Z$, and:

$$\begin{aligned}
\mathbf{x}' &= \mathrm{K}'[\mathrm{R} \mid \mathbf{t}](X,Y,Z,1)^T && (3)\\
&= \mathrm{K}'\mathrm{R}(X,Y,Z)^T + \mathrm{K}'\mathbf{t} && (4)\\
&= \mathrm{K}'\mathrm{R}\mathrm{K}^{-1}\mathbf{x}Z + \mathrm{K}'\mathbf{t}. && (5)
\end{aligned}$$

We divide both sides by Z to obtain the mapping of an image point $\mathbf{x}$ to image point $\mathbf{x}'$

$$\mathbf{x}' = \mathrm{K}'\mathrm{R}\mathrm{K}^{-1}\mathbf{x} + \mathrm{K}'\mathbf{t}/Z = \mathrm{H}_\infty\mathbf{x} + \mathrm{K}'\mathbf{t}/Z = \mathrm{H}_\infty\mathbf{x} + \mathbf{e}'/Z. \tag{6}$$

H_∞ is the infinite homography that transfers the points at infinity to the points at infinity. If $R = I$ (e.g., pure translation) the point $\mathbf{x}$ will undergo a motion along a corresponding epipolar line:

$$\mathbf{x}' = \mathbf{x} + K'\mathbf{t}/Z = \mathbf{x} + \mathbf{e}'/Z. \tag{7}$$

If $\mathbf{t} = \mathbf{0}$ the motion of the point may be represented by a homology:

$$\mathbf{x}' = H_\infty \mathbf{x}. \tag{8}$$

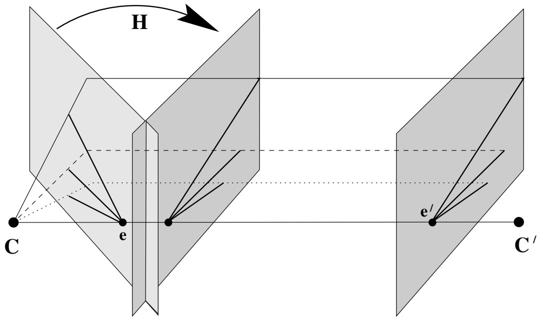

Figure 1: (Adapted from [9]) The effect of the camera motion on the image points may be viewed as a two-step process: mapping by a homography H_∞ followed by a motion along the corresponding epipolar lines.

In a general case the mapping of an image point $\mathbf{x}$ into $\mathbf{x}'$ may be viewed as a two step process: transformation by a homology (a specialization of homography which has two equal eigenvalues) H_∞ which simulates a pure rotational motion of the camera followed by an offset along the epipolar line which simulates a pure translational motion of the camera, see Figure 1.

3 Motion Estimation

Our strategy to attack the problem is to separate it into two smaller sub-problems: rotation estimation and translation estimation. The algorithm relies on the ability to partition the scene points into two sets: the distant and the near-by ones. The distant points are used for rotation estimation while the near-by ones take part in the translation estimation.

First, the stereo algorithm is presented, followed by the monocular one. The main difference is in the translation estimation part. While it is possible to implement a stereo-like algorithm in the monocular setting as well, it suffers from the scale drift. Thus, we propose a different technique.

3.1 Stereo

Partitioning the points To partition the points in the stereo case we hard-threshold their Z-coordinates (the threshold is a parameter of the algorithm). The depth of the points was computed by a stereo triangulation.

Rotation Estimation: We use distant points to estimate rotation R (i.e., near-by points do not take part in rotation estimation). As Eq. (6) states:

$$\mathbf{x}' = \mathrm{KRK}^{-1}\mathbf{x} + \mathrm{K}\mathbf{t}/Z. \tag{9}$$

The total motion of the feature point in the image plane may be viewed as a two-step process (the order is not important): transformation by homography $\mathrm{H}_\infty = \mathrm{KRK}^{-1}$ followed by displacement along the line defined by the epipole e' and the point $\mathrm{H}_\infty\mathbf{x}$. The magnitude of the displacement along the epipolar line depends on the the camera translation and the inverse depth of the point, see Figure 2.

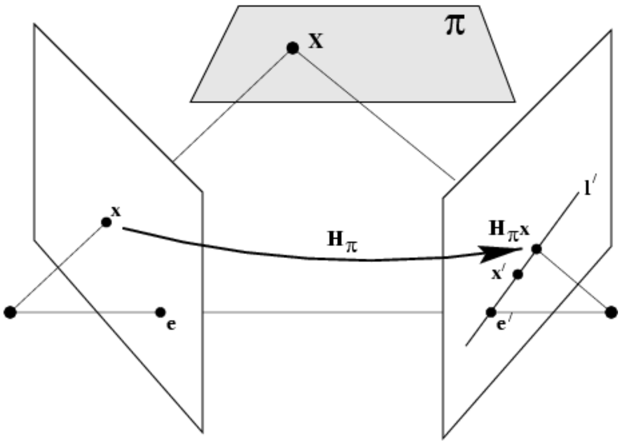

Figure 2: (Adapted from [9]) The homography H_π transfers the point onto a corresponding epipolar line. The displacement along the epipolar line depends on the inverse depth of the point and the camera translation magnitude.

Our estimation algorithm consists of initialization and non-linear refinement.

Initialization: to compute the initial estimate of the rotation parameters we assume that for the distant points (s.t., $\|\mathbf{t}\|/Z \ll \|\mathrm{H}_\infty\mathbf{x}\|$):

$$\mathbf{x}' \approx \mathrm{KRK}^{-1}\mathbf{x}. \tag{10}$$

This assumption is justified by the fact that for the distant points the displacement along the epipolar line is small. We multiply both sides of Eq. (10) by K^{-1} and denote $\mathbf{u}' = \mathrm{K}^{-1}\mathbf{x}$ and $\mathbf{u} = \mathrm{K}^{-1}\mathbf{x}$:

$$\mathbf{u}' = \mathrm{K}^{-1}\mathbf{x}' \approx \mathrm{RK}^{-1}\mathbf{x} = \mathrm{R}\mathbf{u}. \tag{11}$$

Since $\mathbf{u}$ and $\mathbf{u}'$ are projective quantities, only their directions are of importance, we normalize them to unit length and denote normalized quantities by $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{u}}'$ respectively. We choose a sample of n points ($n = 3$) and stack them as columns of matrices $\tilde{\mathrm{U}}$ and $\tilde{\mathrm{U}}'$ respectively. We search for R that solves the following minimization problem:

$$\underset{\mathrm{R}}{\operatorname{argmin}} \|\tilde{\mathrm{U}}' - \mathrm{R}\tilde{\mathrm{U}}\|_2. \tag{12}$$

Eq. (12) is known as the absolute orientation problem (see e.g., [11]) and its solution provides an initial estimate for the subsequent non-linear optimization problem.

Refinement: The idea of the refinement is this: the residual vector $H_\infty \mathbf{x} - \mathbf{x}'$ may be viewed as a sum of a vector orthogonal to the epipolar line and the vector parallel to it. We search for the camera rotation that ignores the parallel component (we view it as a "legal" bias) while trying to minimize the orthogonal one. We define the point residual as the orthogonal distance to the corresponding epipolar line and minimize the sum of squared residuals for all points. We do so, because, as Eq. (6) suggests, after we compensate for a rotation, the point is still allowed to move along the epipolar line.

Consider the objective:

$$\mathrm{R}(\mathrm{v},\theta) = \underset{v,\ \theta}{\operatorname{argmin}} \sum_{i=1}^{N} {r_i}^2 \quad \text{s.t.} \quad r_i = n_i \cdot (\mathbf{x}'_i - H_\infty(v,\theta)\mathbf{x}_i) \\ \text{where } n_i = (\mathrm{F}\mathbf{x}_i)_\perp \tag{13}$$

$\mathrm{H}_\infty(v,\theta) = \mathrm{KR}(\mathrm{v},\theta)\mathrm{K}^{-1}$ is the homography that transforms the image points in Eq. (6). It depends on a known camera intrinsic matrices K and a rotation matrix R. We choose to parameterize the rotation by an angle θ and an axis v. F denotes the fundamental matrix that corresponds to two subsequent stereo rig poses and is computed elsewhere. $\mathrm{F}\mathbf{x}_i$ denotes the epipolar line that corresponds to $\mathbf{x}_i$ in the second image and $(\mathrm{F}\mathbf{x}_i)_\perp$ is the normal to this line. We solve the minimization problem (13) by means of the Levenberg-Marquardt optimization algorithm.

To make the estimation robust we wrap the initialization procedure into the RANSAC iterations. We choose the strongest consensus estimate and its support set as an input for the solution of the Eq. (13).

Translation Estimation (1-point algorithm) To estimate the translation we use only the near-by points. First, we triangulate these points in the previous stereo pair to obtain their 3-D locations, and then iteratively minimize the sum of reprojection errors into the current frame.

The reprojection of point $\mathbf{X} = (X,Y,Z,1)^T$ into the current left image is given by:

$$\pi^{(l)}(\mathbf{X};\mathbf{t}) = K\Big[\mathrm{R} \mid \mathbf{t}\Big]\mathbf{X}. \tag{14}$$

and the reprojection of the same point into the current right image (b is the baseline of the stereo-rig) is given by:

$$\pi^{(r)}(\mathbf{X};\mathbf{t}) = K\Big[\mathrm{R} \mid \mathbf{t}\Big](\mathbf{X} - (b,0,0,0)^T), \tag{15}$$

We use the Levenberg-Marquardt algorithm to iteratively minimize the sum of squared reprojection errors (starting from $\mathbf{t} = \mathbf{0}$):

$$\|\mathbf{x}' - \pi^{(l)}(\mathbf{X};\mathbf{t})\|^2 + \|\mathbf{x}' - \pi^{(r)}(\mathbf{X};\mathbf{t})\|^2. \tag{16}$$

There are three unknown parameters, since $\mathbf{t} = (t_x,t_y,t_z)^T$, thus a single 3-D point provides enough constraints to determine $\mathbf{t}$.

3.2 Mono

The stereo setup has an advantage over the monocular one in the sense that it provides the algorithm with more information (e.g., the calibration and the additional image at each camera position). These advantages come at a price, e.g., the cameras need to be synchronized and the computational resource requirements climb. These make monocular setups and the related algorithms more attractive. In the following section we present the version of our algorithm, adapted for the monocular setup.

Given two sets of matching image points $\mathbf{x}_i, \mathbf{x}_i'$ from two subsequent frames I_1, I_2 respectively, we estimate the camera motion between these frames. Similar to the stereo algorithm in Section 3.1 the algorithm first partitions the points and then estimates the rotation followed by the estimation of the translation direction (it is well known that the magnitude of the camera translation is unavailable in the monocular setting).

Partitioning the points To estimate the rotation of the camera as in the Eq. (3), it is required to partition the set of the image points into the distant ones and the near-by ones. While in the stereo setting we may triangulate the points and threshold their depths, in the monocular setting this can not be done. This section proposes a method to perform the aforementioned partition in the monocular setting.

Consider the subsequent images I_1, I_2, I_3 taken by a moving camera at the locations O_1, O_2 and O_3 respectively. We assume the image points $\mathbf{x}_i$ in I_1 are known to be distant relative to the camera at O_1. The magnitude of the camera translation is small relative to the distant points depths, thus we assume that these points are distant w.r.t. the camera at O_2 and O_3 as well. Some of these points will be lost in I_3, thus it is desirable to known which of the points tracked from I_2 to I_3 (which are not part of $\mathbf{x}_i$'s) are distant (denote these by $\mathbf{y}_j$).

The real baselines $t_1 = \|O_1 - O_2\|$ and $t_2 = \|O_2 - O_3\|$ are unknown and thus we can not use them to obtain real depths of the points.

We use the following procedure to classify the newly tracked points in I_2 as distant:

1. Set $t_1 = 1$ and triangulate the distant points $\mathbf{x}_i$ to obtain the depths Z_i
2. Set $t_2 = 1$ and triangulate the points $\mathbf{y}_j$ to obtain the depths Z_j
3. Classify the point y_j to be distant if $Z_j > \min_i Z_i$.

While the assumption $t_1 \approx t_2$ is acceptable for the KITTI dataset, it may be improved on by computing the t_1/t_2 ratio (by minimizing the reprojection errors of $\mathbf{x}_i$ into I_3, similar to the translation estimation described in the Section 3.1).

To initialize the monocular algorithm we may further assume that the initial motion is a pure translation and thus the points with small disparity are the distant ones (disparity being the magnitude of the motion in the image plane).

Rotation Estimation is exactly as in the Section 3.1. Denote the estimated rotation by R and the corresponding homography by H.

Translation Estimation is as follows. Compensate the rotation by computing $\mathbf{y}_i = \mathrm{H}_\infty \mathbf{x}_i$. We optimize over the location of the epipole e and minimize the orthogonal distances of the points to their corresponding epipolar lines:

$$\underset{e}{\text{argmin}} \sum_i d(l_i, \mathbf{y}_i) + d(l_i, \mathbf{x}_i') \text{ s.t. } d(e, l_i) = 0 \text{ while } l_i = \underset{l}{\text{argmin}}\, d(l, \mathbf{x}_i') + d(l, \mathbf{y}_i) \tag{17}$$

We define the epipolar line l_i to be the line that passes through the epipole and its distance to $\mathbf{x}_i'$ and $\mathbf{y}_i$ is minimal. $d(l, \mathbf{x})$ denotes the distance from the line l to the point $\mathbf{x}$. The epipole provides us with the translation direction of the camera.

4 Experimental Results

4.1 The Choice of Features

We chose to evaluate our algorithm on the KITTI dataset [5], which is a de-facto standard for the visual odometry research works.

Feature Detector/Descriptor: We use Harris [8] corner detector. It is fast, well localized and (most important) Harris corners are abundant in urban scenes we work with. We detect corners in each new image and then match them to obtain putative matches. We tune sensitivity threshold of the detector in such a way, that we are left with about five hundred putative matches after matching and pruning. We extract a square patch of 7×7 pixels centered at the corner point and use this vector as feature descriptor.

We would like to point out that our method may be used with any feature detector that would allow to match features across images. The choice of feature detector should be viewed as a parameter to the algorithm and mainly depends on the images at test.

Feature Matching: We use sum-of-square differences (SSD) of feature descriptors as a metric function when matching features. For each feature we choose a single best match w.r.t. the metric function in the other image. We employ a number of heuristics to prune outliers:

- Reciprocity: features a and b match only if a matches b and b matches a
- Epipolar constraint: we work with calibrated stereo pair. When we match features across images of stereo pair, the search is one-dimensional, i.e., along the horizontal epipolar line. This heuristic is not used when matching features across subsequent frames.
- Chierality (visibility): also used when matching features across stereo pair images. We triangulate the features to obtain the 3-D point and keep the match only if the 3-D point is visible in both cameras.
- Circular match: similar to [4] we keep only those matches that form a circle.

4.2 Experimental Results

The Tables 1 and 2 present the results of the experiments for the KITTI dataset. The columns denote the number of the sequence. The rows denote the algorithm: SS is the baseline, HX is the stereo version of the algorithm, HG is the monocular version. The numbers are the mean error for the corresponding sequence with the last column is the mean error for the dataset. The error computation method is described in [5].

Table 1: Rotation errors for the KITTI sequences [deg/m]

	00	01	02	03	04	05	06	07	08	09	10	mean
SS	3.95e-04	1.98e-04	4.11e-04	1.07e-03	8.03e-04	3.49e-04	4.72e-04	2.96e-04	3.69e-04	3.44e-04	4.82e-04	4.71e-04
HX	**2.70e-04**	**1.75e-04**	**4.10e-04**	**6.51e-04**	**6.04e-04**	3.95e-04	**3.77e-04**	**2.37e-04**	**3.23e-04**	**3.22e-04**	5.66e-04	**3.93e-04**
HG	8.72e-04	3.89e-04	6.28e-04	1.07e-03	5.99e-04	6.96e-04	3.31e-04	8.12e-04	8.13e-04	6.82e-04	5.23e-04	6.74e-04

Table 2: Translation errors for the KITTI sequences %

	00	01	02	03	04	05	06	07	08	09	10	mean
SS	4.40e+00	9.25e+00	4.03e+00	1.22e+01	5.06e+00	2.80e+00	4.37e+00	2.21e+00	4.12e+00	5.25e+00	5.60e+00	5.39e+00
HX	3.07e+00	1.08e+01	3.80e+00	7.94e+00	3.82e+00	4.06e+00	3.99e+00	1.67e+00	3.28e+00	3.77e+00	5.65e+00	4.72e+00
HG	1.21e+01	1.48e+01	8.72e+00	1.33e+01	8.62e+00	8.37e+00	4.46e+00	7.93e+00	9.76e+00	1.16e+01	8.36e+00	9.82e+00

Stereo In this set of experiments we ran our algorithm in its stereo mode as described in the Section 3.1. Table 1 and 2 present the rotation and the translation errors respectively in the row HX. The columns marked bold are those that our algorithm outperforms the baseline (on 9 of 11 sequences). The results show that our algorithm improves the rotation results over the benchmark algorithm and successfully competes with it in the translation estimation.

Mono In an additional set of experiments we ran our algorithm in a monocular mode. Monocular motion estimation lacks a scale parameter. In order to compare the results we set the scale of the translation to be that of the stereo algorithm. Feature point selection/partition was done without using any stereo information and the motion estimation was done as explained in the Section 3.2.

5 Conclusions and Discussion

This paper presents a novel visual odometry algorithm. The novelty of the algorithm is in its rotation estimation method. The rotation is estimated by means of the infinite homography. The algorithm may be used both in the stereo and in the monocular setting.

The strengths of the presented algorithm are in its ability to split the motion estimation problem into two smaller problems and to operate directly on the image points instead of on the computed 3-D quantities. Splitting the problem helps because each sub-problem is easier to solve. The ability to partition the points into the distant and the near-by ones is what allows us to separate the rotation and the translation estimation.

The stereo version of the algorithm shows better performance, but the monocular version has the advantage of being a more practical one. Indeed the authors in [5] report that they recalibrate the cameras before each drive, which is hardly possible in real world installations.

References

[1] Parallel tracking and mapping for small AR workspaces. *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR*, 2007.

[2] Hernán Badino, Akihiro Yamamoto, and Takeo Kanade. Visual odometry by multi-frame feature integration. *Proceedings of the IEEE International Conference on Computer Vision*, (Cvad 13):222–229, 2013.

[3] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3951 LNCS:404–417, 2006.

[4] Andreas Geiger, Julius Ziegler, and Christoph Stiller. StereoScan: Dense 3d reconstruction in real-time. *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 963–968, 2011.

[5] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

[6] Natasha Govender. Evaluation of Feature Detection Algorithms for Structure from Motion. *Csir*, 2009.

[7] C G Harris and J M Pike. 3D Positional Integration from Image Sequences. *Procedings of the Alvey Vision Conference 1987*, pages 233–236, 1987.

[8] Chris Harris and Mike Stephens. A Combined Corner and Edge Detector. *Procedings of the Alvey Vision Conference 1988*, pages 147–151, 1988.

[9] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[10] Johan Hedborg, Pe Forssén, and Michael Felsberg. Fast and accurate structure and motion estimation. *5th International Symposium on Advances in Visual Computing: Part I*, pages 211–222, 2009.

[11] Berthold K P Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629, 1987.

[12] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. iSAM: Incremental smoothing and mapping. 24(6):1365–1378, 2008.

[13] Laurent Kneip, Davide Scaramuzza, and Roland Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2969–2976, 2011.

[14] Kurt Konolige, Motilal Agrawal, and Joan Solà. Large-scale visual odometry for rough terrain. *Springer Tracts in Advanced Robotics*, 66(STAR):201–212, 2010.

[15] David G Lowe. Distinctive image features from scale invariant keypoints. *Int'l Journal of Computer Vision*, 60:91–11020042, 2004.

[16] Annalisa Milella and Roland Siegwart. Stereo-based ego-motion estimation using pixel tracking and iterative closest point. *Proceedings of the Fourth IEEE International Conference on Computer Vision Systems, ICVS'06*, 2006(Icvs):21, 2006.

[17] Hans Peter Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. *tech. report CMU-RI-TR-80-03*, page 175, 1980.

[18] R A Newcombe, S J Lovegrove, and A J Davison. {DTAM}: Dense Tracking and Mapping in Real-Time. *Int. Conf. on Computer Vision (ICCV)*, pages 2320–2327, 2011.

[19] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, 2004.

[20] Clark F. Olson, Larry H. Matthies, Marcel Schoppers, and Mark W. Maimone. Rover navigation using stereo ego-motion. *Robotics and Autonomous Systems*, 43(4):215–229, 2003.

[21] Mikael Persson, Tommaso Piccini, Michael Felsberg, and Rudolf Mester. Robust stereo visual odometry from monocular techniques. *IEEE Intelligent Vehicles Symposium, Proceedings*, 2015-Augus:686–691, 2015.

[22] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3951 LNCS:430–443, 2006.

[23] Davide Scaramuzza and Friedrich Fraundorfer. Visual Odometry. (June), 2011.

[24] Jianbo Shi Jianbo Shi and Carlo Tomasi. Good features to track. *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, (December):593–600, 1994.

[25] Shiyu Song and Clark C Guest. Parallel , Real-Time Monocular Visual Odometry.

[26] Bill Triggs, Philip F. Mclauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle Adjustment - a Modern Synthesis. *Vision algorithms: theory and practice. S*, 34099: 298–372, 2000.