Author Guidelines for the British Machine Vision Conference

BMVC 2016 Submission # ??

Abstract

In this work we revisit a 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 scene structure typical for that seen by a moving car and is suitable for use in stereo setting. We recover rotation and translation separately, thus dealing with two separate (smaller) problems. The rotation is estimated by means of infinite homography. We start with an initial estimate and then refine it using iterative procedure. After the rotation is compensated for the translation is found by means of 1-point algorithm in stereo setting and epipole computation for pure translational motion in monocular setting. We evaluate our algorithm on the KITTI [**?**] data-set.

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 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 wheel odometry, visual odometry is subject to error accumulation over time. Contrary to wheel odometry, visual odometry is not affected by wheel slip in 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 stereo setup, the scene must be static or the images taken at the same time. Also, video processing incurs computational burden.

The focus of this work is on a car scenario in a stereo setup. We evaluate our algorithm on a KITTI data-set [**?**].

Typical visual odometry pipeline is shown in Figure 1

1.1 Related Work

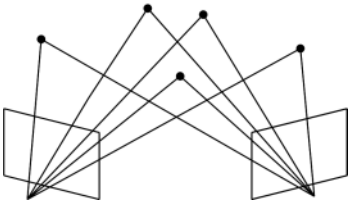

Figure 2: Matching features in two images are projections of the same world point

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

Similar to [**?**] 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

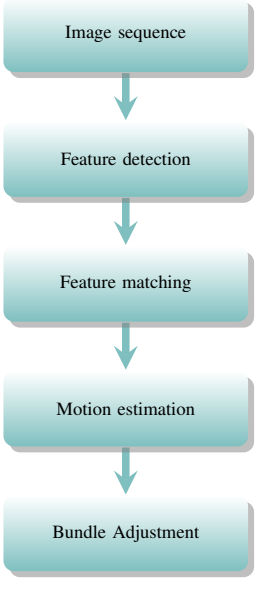

Figure 1: Common visual odometry pipeline

Visual odometry algorithms use large number of corner detectors (e.g. Moravec [**?**], Forstner [**?**], Harris [**?**], Shi-Tomasi [**?**], Fast [**?**]) and blob detectors (e.g., SIFT [**?**], SURF [**?**]). 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 [**?**]. In this work we choose Harris [**?**] corners, but this choice is not crucial. We view feature point choice as a parameter, which needs to be determined from data (e.g., by cross-validation).

Features are either tracked [**?**] or matched [**?**] (i.e., freshly detected in each new frame) between subsequent images. While early works chose to track features, most of the current works detect and match them. The output of this stage are pairs of image features, which are (hopefully) projections of the same 3D point in the real world, as in Figure 2.

Matched features are used as an input for motion estimation procedure. By whether the features are specified in 2-D or 3-D, the estimation procedures, may be classified into 3-D-to-3-D ([**?**], 3-D-to-2-D ([**?**]) and 2-D-to-2-D ([**?**]). Most of the early work was 3D-to-3D. More recent works [**?**] 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 [**?**], perspective 3-point algorithm [**?**], re-projection error minimization [**?**].

Global methods [**?**], [**?**] 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 [**?**] also keep track of (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 computational price). Note that accuracy does not imply robustness, since the outlier that made it into the map may greatly skew subsequent pose estimates.

Methods that explicitly system state uncertainty tend to use filtering mathematical machinery, e.g., [**?**], [**?**], [**?**]. Another alternative to maintain map/pose estimate consistency is to use bundle adjustment approach [**?**]

Monocular systems [**?**] make use of a single camera, while stereo systems [**?**] rely on a calibrated stereo rig. In monocular setup the translation of the camera may only be estimated up to scale, while in stereo all six motion parameters may be recovered. Additional advantage of the stereo setup is more information at each step, which may be one of the reasons why stereo algorithms perform better.

1.2 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 a 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 near-by points.

Current work presents the algorithm in stereo setting, however it may also be used in monocular setting. We use stereo to partition the scene points, in **??** we discuss how this may be done in mono.

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 for two stereo image pairs:

1. Feature detection. We use Harris [**?**] features.
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 [**?**] to reject outliers.
3. Partition scene points into two sets: distant and near-by.
4. Estimate camera rotation from distant points.
5. Estimate camera translation from near-by points.

1.3 Results Outline

We show that on KITTI data-set our method improves upon StereoScan[**?**] results and competes with state of the art methods.

2 Preliminaries and Notation

2.1 Camera model

We consider central projection of points in space onto a plane. Let the center of projection be the origin of the euclidean frame and let plane $Z = f$ be the image plane. World point $\mathbf{X} = [X,Y,Z]^T$ is projected onto the image plane by means of connecting $\mathbf{X}$ and the center of projection and taking the intersection point of the image plane with this line to be a projected point $\mathbf{x} = [fX/Z, fY/Z, f]^T$

If the world point is represented by a homogeneous 4-vectors, e.g., $\mathbf{X} = [X,Y,Z,1]^T$, then:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{pmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{1}$$

Equation 1 assumes that the origin of coordinates in the image plane is at principal point and that the center of projection is at the origin of the world coordinate system.

Let the coordinates of the principal point be $c = [p_x, p_y]^T$ in the image plane frame. Also, let the camera frame orientation be described by $R \in SO(3)$ and its origin be the inhomogeneous vector $C \in \mathbf{R}^3$ as seen from the world frame. Thus, Equation 1 becomes:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} \begin{bmatrix} R & -RC \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2}$$

Denote the *intrinsic parameters* matrix:

$$K = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} \tag{3}$$

Denote $\mathbf{t} = -RC$ and let the *extrinsic parameters* matrix be:

$$\begin{bmatrix} R & \mathbf{t} \end{bmatrix} \tag{4}$$

Putting this together and denoting *camera matrix* $P = K[R\ \mathbf{t}]$, we may write:

$$\mathbf{x} = K[R\ \mathbf{t}]\mathbf{X} = P\mathbf{X} \tag{5}$$

Note that, $\mathbf{X}_{cam} = [\mathrm{R}\ \mathbf{t}]\mathbf{X}$ is the representation of $\mathbf{X}$ in the camera reference frame.

2.2 Image Point Mapping Related to Camera Motion

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

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

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

$$\mathbf{x} = \mathrm{P}\mathbf{X} \quad \mathbf{x}' = \mathrm{P}'\mathbf{X}$$

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:

$$\begin{aligned} \mathbf{x}' &= \mathrm{K}'[\mathrm{R} \mid \mathbf{t}](X,Y,Z,1)^T && (6)\\ &= \mathrm{K}'\mathrm{R}(X,Y,Z)^T + \mathrm{K}'\mathbf{t} && (7)\\ &= \mathrm{K}'\mathrm{R}\mathrm{K}^{-1}\mathbf{x}Z + \mathrm{K}'\mathbf{t} && (8)\\ & && (9) \end{aligned}$$

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

$$\mathbf{x}' = \mathrm{K}'\mathrm{R}\mathrm{K}^{-1}\mathbf{x} + \mathrm{K}'\mathbf{t}/Z = \mathrm{H}\mathbf{x} + \mathrm{K}'\mathbf{t}/Z = \mathrm{H}\mathbf{x} + \mathbf{e}'/Z \quad (10)$$

If $\mathrm{R} = \mathrm{I}$ (e.g. pure translation) the point $\mathbf{x}$ will undergo a motion along a corresponding epipolar line:

$$\mathbf{x}' = \mathbf{x} + \mathrm{K}'\mathbf{t}/Z = x + \mathbf{e}'/Z \quad (11)$$

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

$$\mathbf{x}' = \mathrm{H}\mathbf{x}$$

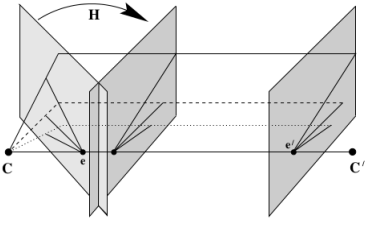

Figure 3: General camera motion may be viewed as a two step process

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.

3 Motion Estimation Pipeline

3.1 Choice of Features

Feature Detector/Descriptor: We use Harris [**?**] 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 [**?**] we keep only those matches that form a circle.

3.2 Motion Estimation

First we partition all features into two sets: distant points and near-by points. We do so by applying a hard threshold to the Z-coordinate of the triangulated point. This threshold is a parameter to the algorithm.

Rotation Estimation: We use distant points to estimate rotation R (i.e., near-by points do not influence this part of the algorithm). The assumption is that for these points:

$$\mathbf{x}' \approx \mathrm{H}_\infty \mathbf{x} = \mathrm{K}'\mathrm{R}\mathrm{K}^{-1}\mathbf{x} \tag{12}$$

The estimation algorithm has two parts: initialization and refinement. To initialize the rotation, we multiply both sides of 12 by K'^{-1} and denote $\mathbf{u}' = \mathrm{K}'^{-1}\mathbf{x}$ and $\mathbf{u} = \mathrm{K}^{-1}\mathbf{x}$:

$$\mathbf{u}' = \mathrm{K}'^{-1}\mathbf{x}' \approx \mathrm{R}\mathrm{K}^{-1}\mathbf{x} = \mathrm{R}\mathbf{u} \tag{13}$$

Since u and u' are projective quantities, only their directions are of importance, we normalize them to unit length and solve absolute orientation [**?**] to obtain initial estimate of R.

Both cameras of the stereo rig undergo the same rigid motion (e.g., the same rotation R and the same translation **t**), therefore features from both images of the stereo pair comply with 12. Thus we unify feature points from both images for sake of estimation (this augmentation only is possible in stereo setting).

To refine the initial estimate of R we use non-linear optimization procedure. Consider the objective:

$$\mathrm{R}(\mathrm{v},\theta) = \underset{v,\theta}{\operatorname{argmin}} \sum_{i=1}^{N} {r_i}^2 \quad \text{s.t.} \quad n_i = (\mathrm{F}x_i)_\perp \qquad r_i = n_i \cdot (x_i' - H(v,\theta)x_i) \tag{14}$$

Here $\mathrm{H}(v,\theta) = \mathrm{K}'\mathrm{R}(\mathrm{v},\theta)\mathrm{K}^{-1}$ and $(\mathrm{F}x_i)_\perp$ is the normal to the epipolar line. In other words, we define the error to be orthogonal distance to the corresponding epipolar line. We do so, because as 10 suggests that after we compensate for a known rotation R we are still left with (small for distant points) displacements along corresponding epipolar lines.

We minimize the sum of squared errors by means of Levenberg-Marquardt optimization algorithm. Rotations are parameterized by axis/angle representation and the optimization is over $v \cdot \theta$ which is simple to implement and fair **??** . To make this procedure robust to outliers, we wrap it into RANSAC iterations. Finally, we choose a rotation that has the largest support set.

Translation Estimation We apply translation estimation only to 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:

$$\pi^{(l)}(\mathbf{X};\mathbf{t}) = K\Big[\mathrm{R} \mid \mathbf{t}\Big]\mathbf{X} \tag{15}$$

and the reprojection of the same point into the current right image is given by:

$$\pi^{(r)}(\mathbf{X};\mathbf{t}) = K\Big[\mathrm{R} \mid \mathbf{t}\Big](\mathbf{X} - (b,0,0,0)^T) \tag{16}$$

Where

- $\mathrm{R} \in SO(3)$ is a rotation matrix
- $\mathbf{t} \in \mathbf{R}^3$ is an inhomogeneous camera translation vector
- K is a camera intrinsic matrix, as in 3

We use Levenberg-Marquardt algorithm to iteratively minimize the sum of squared reprojection errors:

$$\|\mathbf{x}' - \pi^{(l)}(\mathbf{X};\mathbf{t})\|^2 + \|\mathbf{x}' - \pi^{(r)}(\mathbf{X};\mathbf{t})\|^2 \tag{17}$$

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 **t** (this minimization procedure is referred to as 1-point algorithm in the literature [ref?]).

We do not search for an initial estimate of **t** since the optimization procedure usually converges to a close enough solution.

Table 1: Rotation errors for KITTI sequences

	00	01	02	03	04	05	06	07	08	09	10	mean
SS	3.04e-04	1.15e-04	2.77e-04	1.04e-03	4.52e-04	2.43e-04	4.28e-04	2.52e-04	4.99e-04	3.12e-04	3.91e-04	3.92e-04
IO-S	5.57e-04	9.17e-05	4.51e-04	3.63e-04	4.11e-04	4.11e-04	1.25e-04	5.53e-04	3.24e-04	2.17e-04	7.43e-04	3.86e-04
IO-M	3.26e-04	1.36e-04	2.99e-04	2.51e-04	1.57e-04	3.29e-04	2.18e-04	3.47e-04	3.11e-04	2.21e-04	4.32e-04	2.75e-04

Table 2: Translation error for KITTI sequences

	00	01	02	03	04	05	06	07	08	09	10	mean
SS	3.19	5.61	3.35	12.4	2.75	2.14	4.67	2.41	6.2	4.59	5.06	4.76
IO-S	5.73	5.75	4.12	2.52	1.36	3.67	1.63	4.35	3.36	2.89	4.29	3.61
IO-M	3.53	6.22	3.09	3.54	1.5	2.29	2.55	2.75	3.17	3	5.97	3.42

3.3 Monocular setting

In this section we show how our algorithm may be adapted to monocular setting.

The dependence of the algorithm as described in previous sections on stereo is in the scene point partition stage. We are interested in points, s.t. in 10 the $\mathbf{t}/Z$ is small ($\mathbf{t}$ is the translation of the camera or "baseline" and Z is the depth of the 3D-point). This causes the desired property of homography mapping to hold.

When we threshold the distance to the 3-D point (e.g., Z) we do it in units of baseline, e.g.,

$$Z < \alpha \|\mathbf{t}\| \tag{18}$$

which is equivalent to

$$\frac{\|\mathbf{t}\|}{Z} > \frac{1}{\alpha} \tag{19}$$

Note that in the monocular setting the baseline is known only up to scale, however, since Z is known also up to the same scale factor it cancels out and the threshold is the same threshold.

4 Experiments

4.1 Stereo

We provide results for our algorithm on a KITTI dataset [**?**].

4.2 Mono

References

[1] A. Alpher. Frobnication. *Journal of Foo*, 12(1):234–778, 2002.

[2] A. Alpher and J. P. N. Fotheringham-Smythe. Frobnication revisited. *Journal of Foo*, 13 (1):234–778, 2003.

[3] A. Alpher, J. P. N. Fotheringham-Smythe, and G. Gamow. Can a machine frobnicate? *Journal of Foo*, 14(1):234–778, 2004.

[4] Authors. The frobnicatable foo filter, 2006. ECCV06 submission ID 324. Supplied as additional material `eccv06.pdf`.

[5] Authors. Frobnication tutorial, 2006. Supplied as additional material `tr.pdf`.

[6] N. David Mermin. What's wrong with these equations? *Physics Today*, October 1989. http://www.cvpr.org/doc/mermin.pdf.