Epipolar geometry in practice: 3D runtime odometry with a web-camera

George Terzakis[1], Phil Culverhouse[1], Guido Bugmann[1],Sanjay Sharma[2], and Robert Sutton[2]

[1]*Center for Robotics and Neural Systems, Plymouth University, Plymouth, UK*

[2]*School of Marine Science and Engineering, Plymouth University, Plymouth, UK*

{Georgios.Terzakis ,P.Culverhouse , G.Bugmann , Sanjay.Sharma , R.Sutton }@plymouth.ac.uk

Technical Report No.: MIDAS.SNMSE.2013.TR.007

Plymouth University, March 11, 2013

Abstract

This paper intends to present and analyze all the necessary steps one needs to take in order to build a simple application for structure from motion in practice using only the epipolar constraint. In other words, toillustratein practice (i.e,, cook-book style) about how the most fundamental of concepts in epipolar geometry can be combined into a complete and disambiguated series of steps for camera motion estimation and 3D reconstruction in a relatively simple computer program.In particular, the essential matrix is re-introduced in order to avoid misconceptions regarding the nature of the rotation matrix and translation between two camera positions during consequent computations. Strictly based on the formulation of the essential matrix in this paper, the most important of its properties are given with proofs, merely to provide necessary intuition into the series of derivations that ultimately lead to the recovery of the orientation and baseline vector (up-to-scale). Finally, an application that performs 3D position tracking of a simple web-camera is implemented (and source code made available) in order to demonstrate the practical value of the essential matrix with only a few of assumptions loosely in place (still background and relatively reliable point tracking).

1. Introduction

The problem of 3D reconstruction and odometry estimations has been under the spotlight of research in computer vision and robotics for the past two decades. Since the early paper by Longuett – Higgins [1] which built a start-up theoretical layer on the original photogrammetric equations by introducing the epipolar constraint, there have been numerous studies and proposed algorithms to estimate camera motion and to realize the world locations of a sparse cloud of points [2-5]. Such problems are now known as structure from motion (SFM) problems and fall in the broader field of multiple view geometry in computer vision.

Although the geometry of multiple images has been extensively researched, structure from motion remains a vastly unexplored territory, mainly due to great informational bandwidth that images carry. The pipeline of SFM starts with the detection of corresponding points in two distinct images of the same scene. There have been several algorithms that match points across multiple views, known as optical flow estimation methods [6-8]. Although such methods are being extensively employed with a great deal of success in many cases, they are nevertheless relying on a number of strong assumptions such as brightness constancy, no occlusions and relatively slow camera motion. Limitations in point will consequently propagate a fair amount of noise throughout geometrical computations, thereby introducing unstable results in many cases.

Structure from motion therefore still remains a very challenging problem in computer vision posed as the trade-off between numerical stability in geometrical computations, and the domestication of the highly complex content of real-world images.

2. Dissecting the epipolar constraint for use in practice

For the following analysis, it will be assumed that the projections of the tracked points are expressed in terms of a coordinate frame with its origin coinciding with the lower left corner of the screen. In other words, the 2D (uncalibrated) coordinates of a screen point (x, y) , are given by$[x\ y]^T = [x_s h_s - y_s]^T$, where x_s, y_s are the original screen coordinates, and h_s is the height of the image in pixels.

Although epipolar geometry applies to multiple motions in a scene by assigning multiple constraints[9], for the analysis that follows, the scene will be assumed to be static and containing no occluded regions. This is an acceptable assumption, especially in the case of footage of the estuarine taken from a moving boat.

2.1 Moving between coordinate frames

The goal is to infer a rigid transformation which "takes" the coordinate frame of the camera in the first view (let view #1) to the coordinates (in terms of the coordinate frame of view #1) and orientation of the camera frame in view #2. The definition of this transformation should be handled with care, since it can be the source of many misconceptions in the course of computations.

Let R be the rotation that aligns the three unit vectors of the first frame (first view) with the three unit vectors of the second frame (second view) and also let *b*bethe coordinate vector of the second camera center in terms of the first frame (also known as baseline). Then, it will be shown that the coordinate vectors of a world point M in the first frame,$M_1$and the second frame M_2, are related with the following relationship:

$$M_2 = R^T(M_1 - b) \qquad (\mathbf{1})$$

To derive the relationship of eq. (1), we consider the vectors U_1, U_2 and b as shown in figure 1.

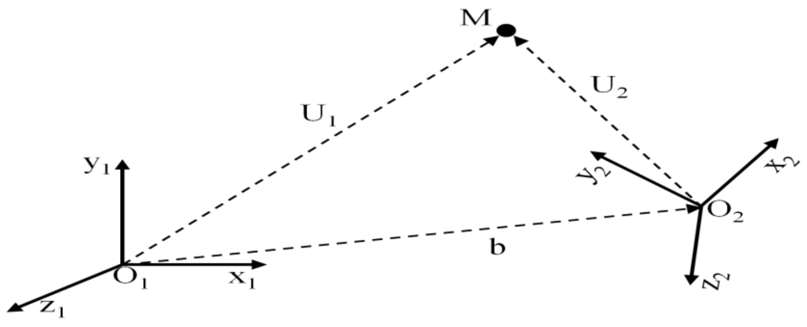

Fig. 1. The two camera frames in terms of the corresponding unit vectors x_1, y_1, z_1 and x_2, y_2, z_2 and the vectors U_1, U_2, b (notice that the coordinates of b are also the coordinates of O_2).

It is obvious that the coordinates of point M in the second coordinate frame will be simply the projections of U_2 onto the x_2, y_2 and z_2 respectively. From fig. 1, we see that U_2 can be easily expressed in terms of b and U_1, as $U_2 = U_1 - b$. We can then easily obtain the sought projections as follows:

$$M_2 = \begin{bmatrix} x_2 \cdot U_2 \\ y_2 \cdot U_2 \\ z_2 \cdot U_2 \end{bmatrix} = \begin{bmatrix} \left(R\begin{bmatrix}1\\0\\0\end{bmatrix}\right)^T \cdot U_2 \\ \left(R\begin{bmatrix}0\\1\\0\end{bmatrix}\right)^T \cdot U_2 \\ \left(R\begin{bmatrix}0\\0\\1\end{bmatrix}\right)^T \cdot U_2 \end{bmatrix} = R^T U_2 = R^T(U_1 - b) = R^T(M_1 - b) \qquad (\mathbf{2})$$

where·denotes the Euclidean inner product operator.

1.2 (Yet another) Derivation of the epipolar constraint

The epipolar constraint is simply a coplanarity constraint between the baseline b, and the two back-projectionrays defined by m_1, m_2 and the two camera centers (figure 2). It is clear that any world point with projections m_1 and m_2 on the two respective camera planes (in Euclidean space) forms a plane with the baseline vector. This can be indirectly expressed as an orthogonality constraint between the cross product of two vectors found on the epippolar plane and a third vector also situated in the epipolar plane. To express this constraint, one maysimply demand that the inner product of U_2 with the cross product of the b and $U_1$is zero:

$$U_2 \cdot (b \times U_1) = 0 \qquad (\mathbf{3})$$

Since $U_2 = U_1 - b = M_1 - b$and $U_1 = RU_2 + b = RM_2 + b$, substitution on eq. (3) yields:

$$(M_1 - b) \cdot \left([b]_\times (RM_2 + b)\right) = 0 \quad (\mathbf{4})$$

where $[b]_\times$ is the cross product skew symmetric matrix of b. Using matrix multiplication for the inner product of eq. (3) we get the following:

$$(M_1 - b)^T \left([b]_\times (RM_2 + b)\right) = 0 \quad (\mathbf{5})$$

By applying the distributive law and taking into consideration the fact that $b^T[b]_\times = [b]_\times b = 0$, the following constraint is obtained:

$$M_1{}^T [b]_\times R M_2 = 0 \quad \Leftrightarrow \quad M_2{}^T R^T [b]_\times M_1 \quad (\mathbf{6})$$

Since M_1 and M_2 are related to their projections the respective depth (scale) factors, the constraint can be re-expressed in terms of m_1 and m_2 (in homogenous coordinates):

$$m_1{}^T [b]_\times R m_2 = 0 \quad \Leftrightarrow \quad m_2{}^T R^T [b]_\times m_1 \quad (\mathbf{7})$$

If we dub the 3x3 matrix$R^T[b]_\times$ as the essential matrix E, then expression for the epipolar constraint is obtained in its widely recognized form:

$$m_2{}^T E m_1 = 0 \qquad (\mathbf{8})$$

where $E = \ R^T[b]_\times$.

It is clear that there can be several different manifestations of the constraint, depending on the ordering of the vectors in the product and the interpretation of the rotation matrix and this can cause of great misunderstandings, especially when trying to "stitch" different works in literature on the very same constraint. In most formulations, the second projection comes first in the product and this is the approach followed here. Also, the rotation matrix R denotes the rotation that rotates the three unit vectors of the first frame to align with the three unit vectors in the second frame.

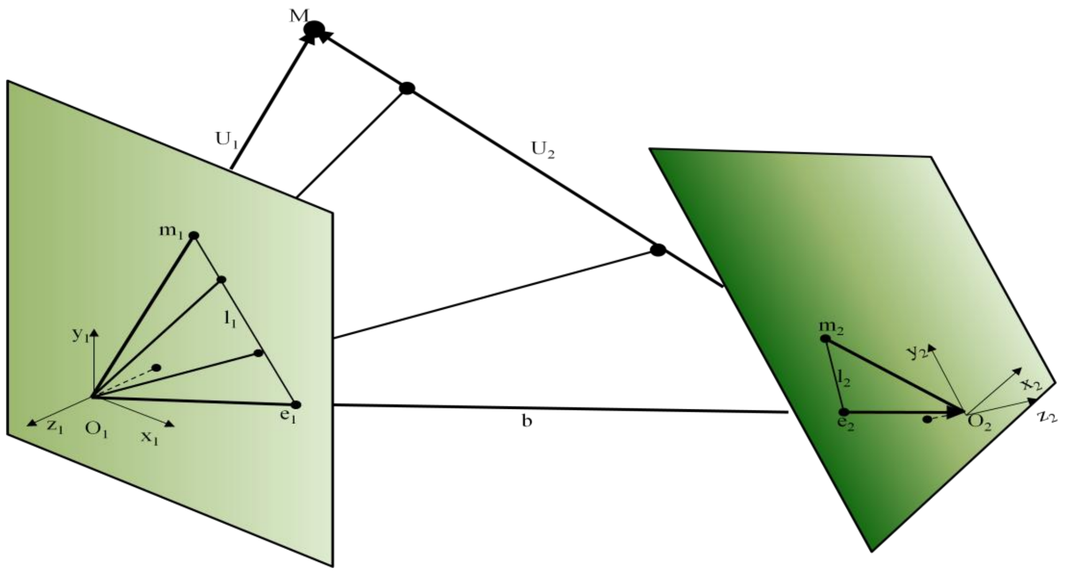

Fig. 2. The epipolar plane defined by a 3D point M and the two camera centers, O_1 and O_2 (in other words, the baseline b). The projections e_2 and e_1 of O_1 and O_2 onto the opposite camera planes, are calledepipoles.The lines l_1 and l_2 clearly are the intersection of the epipolar plane with the first and second camera plane respectively.The points on l_1, $l_2$back-project on points on the rays O_2M and O_1M respectively.

3. Computing the essential matrix from a set of correspondences

Typical image coordinate frames are inverted in the y-axis and, as mentioned earlier, a homogeneous location u in uncalibrated (pixel) space used in computations will have the following (homogeneous) coordinates:

$$u = \begin{bmatrix} x_s \\ h_s - y_s \\ 1 \end{bmatrix} \quad (\mathbf{9})$$

where x_s, y_s are the original image coordinates of the point and h_s is the height of the image.

3.1 Obtaining the essential matrix from the fundamental matrix

The essential matrix defines a bilinear relationship between the Euclidean coordinates of projections in the two views. In practice, an image is not a plane in Euclidean space, but it can be rather thought of as an affinely distorted version of one such plane. In other words, image coordinates contain a certain amount of scaling and cannot be directly used to compute the essential matrix. A simple solution would be to obtain the Euclidean coordinates of each point as $K^{-1}m_s$,where K is the intrinsic parameters matrix of the camera and thereafter compute the essential matrix. Unfortunately, this can introduce a great deal of numerical instability. However, if we substitute $m_1 = K^{-1}u_1$ and $m_2 = K^{-1}u_2$ (where u_1 and u_2 are the known uncalibrated coordinates introduced in eq. (9) corresponding to m_1 and m_2) in eq. (8), we get the following constraint:

$$(K^{-1}u_2)^T E K^{-1} u_1 = 0 \qquad (\mathbf{10})$$

which, with a few manipulations takes the following form:

$$u_2{}^T (K^{-T} E K^{-1}) u_1 = 0 \qquad (\mathbf{11})$$

The new form of the constraint implies that there exists a new matrix $F = K^{-T}EK^{-1}$ that links the uncalibrated homogenous coordinates of the correspondences. The matrix is known as the fundamental matrix [10] and it establishes the epipolar bilinear constraint in uncalibrated space.

$$u_2{}^T F u_1 = 0 \qquad (\mathbf{12})$$

It is therefore much easier to recover the fundamental matrix directly from the known uncalibrated coordinates and thereafter compute the essential as follows:

$$E = K^T F K \qquad (\mathbf{13})$$

There have been numerous algorithms proposed in literature for the computation of the fundamental matrix [2, 4]. The most popular implementation is however the 8-point algorithm [1], which, following Hartley's modification [11, 12]yields remarkably stable results.

4.Properties of the essential matrix

The most important properties of the essential matrix are listed below (trivial proofs are omitted):

1. *If E is an essential matrix, then the epipoles $e_1$and e_2 are the left and right null spaces of E respectively.*

   <u>Proof</u>

   Since the epipoles are the scaled images of the camera centers, then there exist $\lambda_1$and λ_2 such that $e_2 = \lambda_1 O_1$and $e_1 = \lambda_2 O_2$, where $O_1$and O_2 are expressed in terms of the coordinate frames of the second and first camera center respectively. By the definition of the fundamental matrix in eq. (8), $E = R^T[b]_\times$. Therefore:

$$e_2{}^T E = \lambda_1 O_1{}^T R^T [b]_\times = \lambda_1 (\underbrace{RO_1}_{-b})^T [b]_\times = -\lambda_1 b^T [b]_\times = 0$$
$$E e_1 = R^T [b]_\times \lambda_2 O_2 = \lambda_2 R^T [b]_\times \underbrace{O_2}_{b} = \lambda_2 R^T [b]_\times b = 0$$

2. *Any point m_1 (m_2)has an associated epipolar line l_2 (l_1) in the other view given by (points and lines are expressed in homogeneous coordinates):*

$$l_2 = E m_1 = e_2 \times m_2 \qquad (l_1 = E m_2 = e_1 \times m_1)$$

3. *(A theorem by Huang-Faugeras[13]) A 3x3 matrix E is an essential matrix if and only if it has a singular value decomposition (SVD)* $E = USV^T$ *such that:*

$$S = diag\{\sigma, \sigma, 0\}, \qquad \sigma \neq 0$$

andU, V are orthonormal transformations (rotations or reflections).

Proof

a. Let E be a fundamental matrix. Then, E is given by$E = R^T[b]_\times$. Taking E^TE,

$$E^TE = (R^T[b]_\times)^TR^T[b]_\times = [b]_\times{}^TRR^TR^T[b]_\times = [b]_\times{}^T[b]_\times(\mathbf{14})$$

We now resort to the following property of the cross product that holds for any linear transformation M and any pair of vectors a, b:

$$(Ma) \times (Mb) = M(a \times b) \quad (\mathbf{15})$$

If now M is an arbitraryorthonormal matrix, let R_0, then:

$$(R_0a) \times b = [R_0a]_x b$$

Multiplying with $R_0{}^T$ from the left,

$$R_0{}^T\big((R_0a) \times b\big) = R_0{}^T[R_0a]_x b \quad (\mathbf{16})$$

Making use of the cross product property in (15), equation (16) becomes:

$$\big(R_0{}^TR_0\big)a \times \big(R_0{}^Tb\big) = R_0{}^T[R_0a]_x b \quad \Leftrightarrow a \times \big(R_0{}^Tb\big) = R_0{}^T[R_0a]_x b$$

$$\Leftrightarrow [a]_\times R_0{}^Tb = R_0{}^T[R_0a]_x b \ \Leftrightarrow \big([a]_\times R_0{}^T - R_0{}^T[R_0a]_x\big)b = 0 \ \ \forall\, a, b, R_0$$

$$\Leftrightarrow [a]_\times = R_0{}^T[R_0a]_x R_0 \qquad (\mathbf{17})$$

If R_0 is chosen to be the rotation that aligns the baseline vector with the z-axis, that is, $R_0b = [0 \quad 0 \quad ||b||]^T$, then by substituting (17) into (14),

$$E^TE = \big(R_0{}^T[R_0a]_xR_0\big)^T R_0{}^T[R_0a]_xR_0 = R_0{}^T[R_0a]_x{}^T[R_0a]_x[R_0a]_xR_0$$

$$\Leftrightarrow E^TE = R_0{}^T\begin{bmatrix} 0 & ||b|| & 0 \\ -||b|| & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 0 & -||b|| & 0 \\ ||b|| & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}R_0$$

$$\Leftrightarrow E^TE = R_0{}^T\begin{bmatrix} ||b||^2 & 0 & 0 \\ 0 & ||b||^2 & 0 \\ 0 & 0 & 0 \end{bmatrix}R_0 \quad (\mathbf{18})$$

The decomposition of eq. (18) is a SVD (one out many) of E^TE. Hence, E will also decompose as follows:

$$E = R_0{}^T\begin{bmatrix} ||b|| & 0 & 0 \\ 0 & ||b|| & 0 \\ 0 & 0 & 0 \end{bmatrix}R_0 \quad (\mathbf{19})$$

b. In the light of the proof in (a), given that there exists a SVD of E in which the first and second singular value are equal (and non-zero), it would be reasonable to attempt to construct a skew symmetric matrix $T_\times$and a rotation matrix Λ, using$U$, S, V such that, $E = \Lambda T_\times$.

The construction of $T_\times$ and Λ that follows is based on the observation that it possible to express $S = diag\{\sigma, \sigma, 0\}$ in the following ways:

$$S = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = R_z\left(-\frac{\pi}{2}\right) S R_z\left(-\frac{\pi}{2}\right)^T \quad \mathbf{(20)}$$

or,

$$S = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = R_z\left(\frac{\pi}{2}\right) S R_z\left(\frac{\pi}{2}\right)^T \quad \mathbf{(21)}$$

where, $R_z(\varphi)$ is the rotation matrix of angle φabout the z-axis.

It is a highly convenient coincidence that the product of the two last matrices in the triplets is a skew symmetric matrix. Specifically:

$$S = R_z\left(\frac{\pi}{2}\right) S R_z\left(\frac{\pi}{2}\right)^T = \begin{bmatrix} 0 & -\sigma & 0 \\ \sigma & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} R_z\left(\frac{\pi}{2}\right)^T \quad \mathbf{(22)}$$

or,

$$S = R_z\left(-\frac{\pi}{2}\right) S R_z\left(-\frac{\pi}{2}\right)^T = \begin{bmatrix} 0 & \sigma & 0 \\ -\sigma & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} R_z\left(-\frac{\pi}{2}\right)^T \quad \mathbf{(23)}$$

Now, the property of skew symmetric matrices in eq. (17) can be of great use in a heuristic sense. In particular, it guarantees that for any rotation matrix $U$and for any skew symmetric matrix $S_\times$, the matrix $U S_\times U^T$ is also a skew symmetric matrix. In the light of this consequence and using (22), the SVD of E can be expressed as follows:

$$E = U R_z\left(-\frac{\pi}{2}\right) \begin{bmatrix} 0 & -\sigma & 0 \\ \sigma & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T = U \underbrace{R_z\left(-\frac{\pi}{2}\right)^T V^T V R_z\left(-\frac{\pi}{2}\right)}_{equal\ to\ identity} \begin{bmatrix} 0 & -\sigma & 0 \\ \sigma & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} R_z\left(-\frac{\pi}{2}\right)^T V^T$$

$$\Leftrightarrow E = \underbrace{\left(U R_z\left(-\frac{\pi}{2}\right)^T V^T\right)}_{a\ rotation\ matrix} \underbrace{\left(V R_z\left(-\frac{\pi}{2}\right)\right) \begin{bmatrix} 0 & -\sigma & 0 \\ \sigma & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \left(V R_z\left(-\frac{\pi}{2}\right)\right)^T}_{also\ a\ skew\ symmetric\ matrix\ by\ virtue\ of\ eq.(17)} \quad \mathbf{(24)}$$

In quite the same, a second rotation-skew symmetric matrix is obtained from eq. (23):

$$E = \underbrace{\left(UR_z\left(\frac{\pi}{2}\right)^T V^T\right)}_{a\ rotation\ matrix}\underbrace{\left(VR_z\left(\frac{\pi}{2}\right)\right)\begin{bmatrix} 0 & \sigma & 0 \\ -\sigma & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\left(VR_z\left(\frac{\pi}{2}\right)\right)^T}_{also\ a\ skew\ symmetric\ matrix\ by\ virtue\ of\ eq.(17)} \qquad \mathbf{(25)}$$

The only remaining "loose end" is to show that matrices $UR_z\left(-\frac{\pi}{2}\right)^T V^T$ and $UR_z\left(\frac{\pi}{2}\right)^T V^T$ are indeed rotation matrices. Since the product involves both U and V, then the sign of the product of their determinants should be positivedue to the fact that these matrices participate in the SVD of $E^T E$ which is always has a positive determinant. And since the determinant of R_z is also positive, it follows that the two aforementioned products are orthonormal matrices with positive determinants, hence rotations.

To summarize, there exist at least two rotation-skew symmetric matrix pairs that correspond to an SVD of 3x3 matrix in which the first two singular values are equal and non-zero, are:

$$E = \underbrace{\left(UR_z\left(-\frac{\pi}{2}\right)^T V^T\right)}_{rotation\ matrix}\underbrace{\left(VR_z\left(-\frac{\pi}{2}\right)\right)R_z\left(-\frac{\pi}{2}\right)S\left(VR_z\left(-\frac{\pi}{2}\right)\right)^T}_{skew\ symmetric\ matrix} \qquad \mathbf{(26)}$$

and,

$$E = \underbrace{\left(UR_z\left(\frac{\pi}{2}\right)^T V^T\right)}_{rotationmatrix}\underbrace{\left(VR_z\left(\frac{\pi}{2}\right)\right)R_z\left(\frac{\pi}{2}\right)S\left(VR_z\left(\frac{\pi}{2}\right)\right)^T}_{skewsymmetricmatrix} \qquad \mathbf{(27)}$$

4. *If E is an essential matrix such that $E = R^T[b]_\times$, then:*

$$Tr(EE^T) = Tr(E^T E) = 2\|b\|^2$$

<u>*Proof*</u>

The product of $E^T E$ yields:

$$E^T E = (R^T[b]_\times)^T R^T[b]_\times = -[b]_\times{}^2 \qquad \mathbf{(28)}$$

If $b = [b_x \quad b_y \quad b_z]^T$, then:

$$E^T E = -\begin{bmatrix} 0 & -b_z & b_y \\ b_z & 0 & -b_x \\ -b_y & b_x & 0 \end{bmatrix}^2 = \begin{bmatrix} {b_y}^2 + {b_z}^2 & -b_x b_y & -b_x b_z \\ -b_x b_y & {b_x}^2 + {b_z}^2 & -b_y b_z \\ -b_x b_z & -b_y b_z & {b_x}^2 + {b_y}^2 \end{bmatrix} \quad \mathbf{(29)}$$

From (29), it can be easily seen that $Tr(E^T E) = 2\|b\|^2$.

Taking now the product EE^T the following relationship arises:

$$EE^T = R^T[b]_\times (R^T[b]_\times)^T = -R^T{[b]_\times}^2 R$$

$$\Leftrightarrow EE^T = (R^T[b]_\times R)(R^T[b]_\times R) \quad \mathbf{(30)}$$

But eq. (17) states that $[a]_\times = {R_0}^T[R_0 a]_x R_0$ for any orthogonal matrix R_0 and for any vector a. Hence, (30) becomes:

$$EE^T = -{[R^T b]_\times}^2 \quad \mathbf{(31)}$$

Clearly, $\|R^T b\| = \|b\|$; it therefore follows from (31) and (29) that $Tr(EE^T) = 2\|R^T b\|^2 = 2\|b\|^2$.

5.Recovering baseline and orientation

With the essential matrix in place, the next step is to obtain the rotation matrix R and the baseline vector *b*¸ up-to-scale. In other words, one needs to assign a value to a global scale factor for the scene (which typically is the norm of the baseline) in order to recover the depth of the 3D locations of the tracked 2D features, thereby recovering the scaled baseline vector, the rotation and the 3D locations of all tracked features from the essential matrix. There are essentially two approaches to extracting the rigid transformation up to a given scale factor λ from the essential matrix[3, 13]. A simple algebraic method proposed by B. Horn [14] appears to be very numerically stable in practice and will be explained here in detail.

Let λbe a scale factor such that $\|b\|^2 = \lambda$ and let $b' = \frac{b}{\|b\|}$ the unit-length baseline vector. We can then solve for b' and R instead of b and R.By considering the product $E^T E$ and by substituting $E = R^T[b]_\times$ we obtain,

$$E^T E = (R^T[b]_\times)^T R^T[b]_\times = -{[b]_\times}^2 = \lambda I_3 - bb^T = \lambda I_3 - \lambda b' b'^T \quad \mathbf{(32)}$$

From eq. (14), it follows that $\frac{Tr(E^T E)}{2} = \lambda$. We may therefore obtain a new (projectively equivalent) essential matrix for which the baseline is b' (i.e., it has unit length):

$$E' = \frac{1}{\sqrt{Tr(E^T E)/2}} E \quad \mathbf{(33)}$$

5.1 Baseline

It is now easier to extract the baseline from *E'*. In particular, taking eq. (14) and solving for one of the three components of *b'* yields two solutions corresponding to a negative and positive sign for that particular component. In other words, there will always be two solutions for the baseline, let b'_1 and b'_2, such that, $b'_1 = -b'_2$.

Without constraining generality, let the *x*-component of the baseline b'_x be non-zero. Since $\|b'\|^2 = 1$ it follows from (29) that,

$$b_x = +\sqrt{1-([E^T E]_{11})^2} \qquad (\mathbf{34})$$

where $[E^T E]_{ij}$ is the element of $E^T E$ in the *i*th row and *j*th column. In quite the same way, one can calculate the absolute value of b_y and b_z:

$$|b_y| = \sqrt{1-([E^T E]_{22})^2} \qquad (\mathbf{35})$$

and,

$$|b_z| = \sqrt{1-([E^T E]_{33})^2} \qquad (\mathbf{36})$$

To discover the signs of b_y and b_z one simply needs to consider the products $b_x b_y = -[E^T E]_{12}$ and $b_x b_z = -[E^T E]_{13}$.

5.2 Rotation

The first step to orientation recovery comes with a brilliant observation which states that if $E' = [e_{ij}] = [(e^1)^T \quad (e^2)^T \quad (e^3)^T]^T$ where e^i_j is the element in the *i*th row and *j*th column of E' and $e_j{}^T$ is the *i*th row of E' (employing tensorial index notation only when denoting row or column vectors, so that matrix rows are not confused with columns in the process of computations) in vector form, then the matrix of cofactors of *E*, $C(E')$ can be expressed as follows:

$$C(E') = \begin{bmatrix} \begin{vmatrix} e_{22} & e_{23} \\ e_{32} & e_{33} \end{vmatrix} & -\begin{vmatrix} e_{21} & e_{23} \\ e_{31} & e_{33} \end{vmatrix} & \begin{vmatrix} e_{21} & e_{22} \\ e_{31} & e_{32} \end{vmatrix} \\ -\begin{vmatrix} e_{12} & e_{13} \\ e_{32} & e_{33} \end{vmatrix} & \begin{vmatrix} e_{11} & e_{13} \\ e_{31} & e_{33} \end{vmatrix} & -\begin{vmatrix} e_{11} & e_{12} \\ e_{31} & e_{32} \end{vmatrix} \\ \begin{vmatrix} e_{12} & e_{13} \\ e_{22} & e_{23} \end{vmatrix} & -\begin{vmatrix} e_{11} & e_{13} \\ e_{21} & e_{23} \end{vmatrix} & \begin{vmatrix} e_{11} & e_{12} \\ e_{21} & e_{22} \end{vmatrix} \end{bmatrix} = \begin{bmatrix} ((e^2)^T \times (e^3)^T)^T \\ ((e^3)^T \times (e^1)^T)^T \\ ((e^1)^T \times (e^2)^T)^T \end{bmatrix} \qquad (\mathbf{37})$$

The scaled essential matrix E' can now be expressed in terms of the columns of *R* ,r_1, r_2 and r_3 and the skew symmetric cross product matrix $[b']_\times$ as follows:

$$E' = R^T [b']_\times = \begin{bmatrix} r_1{}^T \\ r_2{}^T \\ r_3{}^T \end{bmatrix} [b']_\times = \begin{bmatrix} r_1{}^T [b']_\times \\ r_2{}^T [b']_\times \\ r_3{}^T [b']_\times \end{bmatrix} = \begin{bmatrix} -([b']_\times r_1)^T \\ -([b']_\times r_2)^T \\ -([b']_\times r_3)^T \end{bmatrix} = \begin{bmatrix} -(b' \times r_1)^T \\ -(b' \times r_2)^T \\ -(b' \times r_3)^T \end{bmatrix} \qquad (\mathbf{38})$$

From equations (15) and (16), the classic formulation of the scalar triple product identity of cross-products is now clearly recognizable in the rows of $C(E')$:

$$C(E') = \begin{bmatrix} \left((b' \times r_2) \times (b' \times r_3)\right)^T \\ \left((b' \times r_3) \times (b' \times r_1)\right)^T \\ \left((b' \times r_1) \times (b' \times r_2)\right)^T \end{bmatrix} = \begin{bmatrix} \left(b' \cdot \underbrace{(r_2 \times r_3)}_{r_1}\right) b'^T \\ \left(b' \cdot \underbrace{(r_3 \times r_1)}_{r_2}\right) b'^T \\ \left(b' \cdot \underbrace{(r_1 \times r_2)}_{r_3}\right) b'^T \end{bmatrix} = \begin{bmatrix} (b' \cdot r_1) b'^T \\ (b' \cdot r_2) b'^T \\ (b' \cdot r_3) b'^T \end{bmatrix} \quad \mathbf{(39)}$$

From (17), the matrix of cofactors can be re-written in terms of matrix multiplications:

$$C(E') = \begin{bmatrix} r_1{}^T(b'b'^T) \\ r_2{}^T(b'b'^T) \\ r_3{}^T(b'b'^T) \end{bmatrix} = R^T(b'b'^T) \qquad \mathbf{(40)}$$

Equation (40) is partially the final solution. There still exists one last observation to make in order to conclude the derivation:

$$E'[b']_\times = R^T[b']_\times{}^2 = R^T(b'b'^T - I) = R^T(b'b'^T) - R^T \qquad \mathbf{(41)}$$

Substituting (40) in (41), yields:

$$E'[b']_\times = C(E') - R^T$$

$$\Leftrightarrow R^T = -E'[b']_\times + C(E')$$

$$\Leftrightarrow R = \left(-E'[b']_\times + C(E')\right)^T = C(E')^T + [b']_\times E'^T$$

$$\boxed{R = C(E')^T + [b']_\times E'^T} \quad \mathbf{(42)}$$

5.3 Resolving ambiguities in the baseline-rotation solution

From an algebraic point of view, it is clear that, given a normalized essential matrix E', a very same baseline vector can be obtained either from E', or from $-E'$, since $E'^T E = (-E)^T(-E)$. Clearly, this ambiguity reflects the direction of the baseline, since the negative sign can only originate from $[b']_\times$.

Equation (42) introduces a second ambiguity related to the sign of the normalized essential matrix. In particular, since the baseline b'can be obtainedfrom eitherE'or $-E'$, then there exist two possible solutions for the rotation matrix per baseline solution. The latter implies that for a given normalized essential matrix E', there exist four solutions with respect to the sign of E' and b'. Assuming that b' is the "positive" (i.e., the solution in which the first non-zero coordinate in *x*, *y*, *z* ordering is positive) baseline solution then the possible solutions for the rotation are given in table 1.

	$\boldsymbol{b'}$	$\boldsymbol{-b'}$
$\boldsymbol{E'}$	$R = C(E')^T + [b']_\times {E'}^T$	$R = C(E')^T - [b']_\times {E'}^T$
$\boldsymbol{-E'}$	$R = C(E')^T - [b']_\times {E'}^T$	$R = C(E')^T + [b']_\times {E'}^T$

Table 1. Possible rotation solutions for the four sign combinations of $E'$and b'.

The four possible solutions for baseline and orientation have geometric interpretationswith respect to the observed 3D points. In effect, there are two possible directions for the baseline vector. For each direction, there exist two possible orientations of the second camera frame (figure 3).

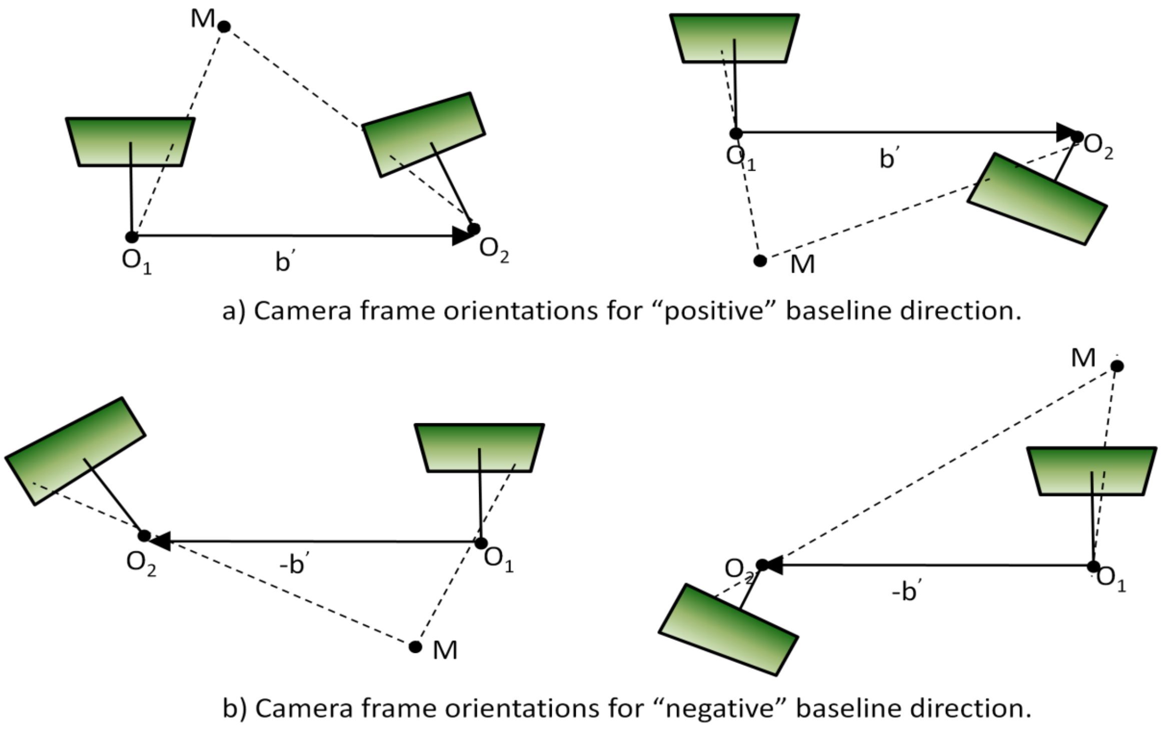
M
O1
b′
O2
M
O1
b′
O2
a) Camera frame orientations for "positive" baseline direction.
O2
-b′
O1
M
M
O2
-b′
O1
b) Camera frame orientations for "negative" baseline direction.

Fig. 3. The four possible combinations of the two camera frame orientations and the two baseline directions (positive direction is chosen arbitrarily) with respect to the location of an observed 3D point M: a)The two possible orientations of the second camera center (*O*$_2$)coordinate frame for positive baseline direction.b) The two possible orientations of the second camera center coordinate frame for negative baseline direction.

Clearly, in only one configuration will all of the observed 3D points lie in front of both camera planes. The correct baseline-orientation therefore corresponds to the configuration that places all points in front of the camera (i.e., all reconstructed 3D points should have a negative (or positive) *Z*-coordinate in both the first and second camera frame. In practice however, the configuration that yields the greatest number of points with negative sign in both coordinate frames should be chosen.

5.4 Retrieving depth (and 3D locations)

Let $M_1 = [X_1 \quad Y_1 \quad Z_1]^T$ and $M_2 = [X_2 \quad Y_2 \quad Z_2]^T$ be the coordinates of M expressed in terms of the first and second camera coordinate frame respectively.Also, let

$m_1 = [x_1 \quad y_1 \quad 1]^T$ and $m_1 = [x_2 \quad y_2 \quad 1]^T$ be the projections of M on the first and second camera plane respectively, in normalized homogeneous coordinates (in uncalibrated space). The projective relationships between *m1* and *M1* is:

$$m_1 = \frac{1}{Z_1} K M_1 \Leftrightarrow M_1 = Z_1 K^{-1} m_1 \qquad (\mathbf{43})$$

where*K* is the matrix of intrinsic parameters of the camera. A similar expression can be obtained for *m2*:

$$m_2 = \frac{1}{Z_2} K M_2 \qquad (\mathbf{44})$$

Substituting eq. (1) into (44) yields:

$$m_2 = \frac{1}{Z_2} K R^T (M_1 - b) \Leftrightarrow \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \frac{1}{Z_2} \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {r_1}^T \\ {r_2}^T \\ {r_3}^T \end{bmatrix} \begin{bmatrix} X_1 - b_x \\ Y_1 - b_y \\ Z_1 - b_z \end{bmatrix} \qquad (\mathbf{45})$$

wheref_x and f_y are the horizontal and vertical pixel ratios, (c_x, c_y) is the center of the screen in pixels, *r1*, *r2*, *r3* are the columns of *R* and $b = [b_x \quad b_y \quad b_z]^T$ is the baseline vector.

Equation (45) yields two equations corresponding to the *x* and *y* coordinate of *m2*:

$$x_2 = \frac{1}{Z_2} f_x {r_1}^T (M_1 - b) \qquad (\mathbf{46})$$

and,

$$y_2 = \frac{1}{Z_2} f_y {r_2}^T (M_1 - b) \qquad (\mathbf{47})$$

From equation (1), *Z2* can be expressed in terms of *r3*, *X* and *b* as:

$$Z_2 = {r_3}^T (M_1 - b) \qquad (\mathbf{48})$$

Substituting (48) into (46) and (47) yields the following:

$$x_2 = f_x \frac{{r_1}^T (M_1 - b)}{{r_3}^T (M_1 - b)} \qquad (\mathbf{49})$$

and,

$$y_2 = f_y \frac{{r_2}^T (M_1 - b)}{{r_3}^T (M_1 - b)} \qquad (\mathbf{50})$$

Finally, substituting (43) into (49) and (50) yields:

$$x_2 = f_x \frac{{r_1}^T (Z_1 K^{-1} m_1 - b)}{{r_3}^T (Z_1 K^{-1} m_1 - b)} \qquad (\mathbf{51})$$

and,

$$y_2 = f_y \frac{{r_2}^T (Z_1 K^{-1} m_1 - b)}{{r_3}^T (Z_1 K^{-1} m_1 - b)} \qquad (\mathbf{52})$$

Equations (51) and (52) yield one solution for Z_1 respectively:

$$Z_1 = \frac{(f_x r_1 - x_2 r_3)^T b}{(f_x r_1 - x_2 r_3)^T K^{-1} m_1} \qquad (\mathbf{53})$$

and,

$$Z_1 = \frac{\left(f_y r_2 - y_2 r_3\right)^T b}{\left(f_y r_2 - y_2 r_3\right)^T K^{-1} m_1} \qquad (\mathbf{54})$$

Either solution provided by (53) or (54) can now be used to recover the 3D location of *M* (eq. (1)). And after M_1 has been recovered, the depth Z_2 can be easily calculated by$Z_2 = {r_3}^T (M_1 - b)$.

6. The algorithm

The method for 3D reconstruction and baseline-orientation recovery for a moving calibrated camera analyzed in the previous sections can be summarized in the algorithm that follows.

3D Reconstruction and baseline-orientation recovery

Input: a) Set of correspondences ${m_1}^{(i)}$ *and* ${m_2}^{(i)}$ *in two respective views of a scene, b) Matrix of camera intrinsic parameters, K.*

Output: a) Rotation matrix R that aligns the first camera frame with the second, b) Baseline vector expressed in terms of the first camera frame, c) 3D coordinates of all points $M^{(i)}$*expressed in with respect to the first camera coordinate frame.*

```
1.  Compute the fundamental matrix F from the two-view correspondences using the modified 8-
    point algorithm.
2.  Compute the essential matrix E:
        $\boldsymbol{E \leftarrow K^T F K}$
3.  Normalize the essential matrix:
        $\boldsymbol{E \leftarrow \frac{1}{\sqrt{Tr(E^T E)/2}} E}$
4.  $\boldsymbol{AcceptReconstruction \leftarrow False}$.
5.  Repeat while $\boldsymbol{AcceptReconstruction == False}$:
6.      Compute baseline b from the elements of $E^T E$,
            $\boldsymbol{E^T E = -[b]_\times^{\,2} = b b^T - I_3}$
        assuming that the first non-zero   component in the order $b_x$, $b_y$, $b_z$ is positive:
7.      $\boldsymbol{Counter \leftarrow 1}$.
8.      Repeat while (AcceptReconstruction == False) and (Counter < 2):
9.          $\boldsymbol{C(E) \leftarrow Matrix\ of\ cofactors\ of\ E}$.
10.         Compute the rotation matrix R:
                $\boldsymbol{R \leftarrow C(E)^T + [b]_\times E^T}$
```

```
11.             For each pair of correspondences $m_1^{(i)}$ and $m_2^{(i)}$:
12.                 Compute the depth of $M^{(i)}$ in terms of the first camera frame:
                        $\boldsymbol{Z_1}^{(i)} \leftarrow \frac{\left(\boldsymbol{f_x r_1} - \boldsymbol{x_2}^{(i)}\boldsymbol{r_3}\right)^{\boldsymbol{T}}\boldsymbol{b}}{(\boldsymbol{f_x r_1} - \boldsymbol{x_2}^{(i)}\boldsymbol{r_3})^{\boldsymbol{T}}\boldsymbol{K}^{-1}\boldsymbol{m_1}^{(i)}}$
                    or,
                        $\boldsymbol{Z_1}^{(i)} \leftarrow \frac{\left(\boldsymbol{f_y r_2} - \boldsymbol{y_2}^{(i)}\boldsymbol{r_3}\right)^{\boldsymbol{T}}\boldsymbol{b}}{\left(\boldsymbol{f_y r_2} - \boldsymbol{y_2}^{(i)}\boldsymbol{r_3}\right)^{\boldsymbol{T}}\boldsymbol{K}^{-1}\boldsymbol{m_1}^{(i)}}$
13.                 Compute the depth $M^{(i)}$:
                        $\boldsymbol{M}^{(i)} \leftarrow \boldsymbol{Z_1 K^{-1} m_1}^{(i)}$
14.                 Compute the depth of $M^{(i)}$ in terms of the second camera frame:
                        $\boldsymbol{Z_2}^{(i)} \leftarrow \boldsymbol{r_3}^{\boldsymbol{T}}\left(\boldsymbol{M}^{(i)} - \boldsymbol{b}\right)$
15.                 If $(\boldsymbol{Z_1}^{(i)}\boldsymbol{Z_2}^{(i)} > 0\ \forall i)$ then $\boldsymbol{AcceptReconstruction} \leftarrow \boldsymbol{True}$.
16.                 $\boldsymbol{Counter} \leftarrow \boldsymbol{Counter} + \mathbf{1}$.
17.                 $\boldsymbol{b} \leftarrow -\boldsymbol{b}$.
18.         $\boldsymbol{E} \leftarrow -\boldsymbol{E}$.
```

Note here, that in practice there may not be any rotation-baseline combination with all depth products $Z_1^{(i)}Z_2^{(i)}$ positive. Usually such cases will rise with the existence of noise in the correspondences. A typical solution to the issue would be to choose the combination of orientation and baseline that minimizes the number of negative depth products.

7. Results

To demonstrate the value of the aforementioned algorithm in terms of odometry, a simple application that resolves rigid transformations up-to-scale (the baseline is always assumed to be of unit length) and changes the viewpoint matrix of an OpenGL [15] window has been created (figure 4). A colored rectangle is placed at the origin and as the camera view changes according to the orientation and position of the webcamera, the effects of perspective become evident in the display window (e.g., the square turns into a trapezium).

Odometry is generally robust, given that brightness does not change significantly (hence optical flow is reliable) and that the background in the sequence remains approximately still. Another important factor that affects the orientation and baseline recovered from the essential matrix is degenerate configurations [16] which typically rise when the camera is not really moving or it is performing pure rotational motion (i.e., baseline is zero).In these cases, epipolar geometry is degenerate and the results obtained from it can be misleading. As a result, the square makes abrupt “jumps” to random directions when the camera is actually still.

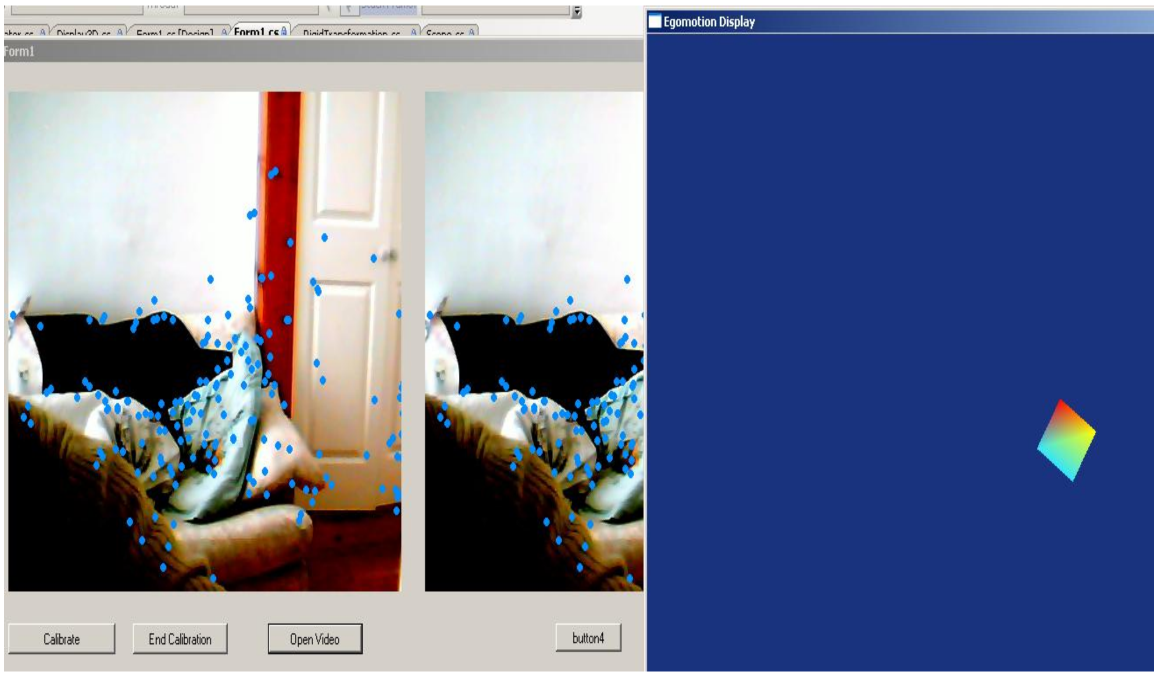

Fig. 4.A simple application that captures odometry up to an unknown scale factor with a webcamera. On the right, previous and current frame in the sequence are displayed; the tracked point-features are indicated as blue dots. The blue window on the right is an OpenGL pseudo 3D environment in which a colored square lies at the origin and the viewpoint changes its orientation and position according to the rotation and baseline vector recovered from the essential matrix (notice that, although the perspective distortion is minor, the square is no longer appearing as a square, but as a trapezium).

The application was implemented in *C#* [17] using the .NET [18] library Emgu 2.2.1 which is a wrapper for the open-source computer vision library (OpenCV) [19]. To draw a rectangle in a pseudo 3D environment, the .NET library OpenTK was used, which is also a wrapper for open-source graphics library OpenGL [15]. The code can be downloaded here.

8. Conclusion

The use of epipolar geometry for runtime odometry and 3D reconstruction has been significantly underestimated to date in the robotics community. This is primarily due the fact that -so far- there have not been any robust formulations for sparse point tracking. Another reason is the numerical stability of the overall process literally “hangs from a thread” mainly when extracting rotation and translation from the essential matrix. Finally, one of the most important reasons for which the epipolar constraint is still unpopular for runtime applications in robotics is that image contents may involve complex configurations such as multiple motions, occlusions, or degenerate arrangements of the point cloud such as coplanar or very distant features.

The point-tracking in this applications relies on the increasingly popular pyramidal LK-tracker [6], while fundamental matrix estimation is done using the RANSAC [20] based method implemented in OpenCV. The quality of the odometry has been tested by simply moving the web-camera and observing the motion of the colored rectangle shown on figure 4. The results are promising, in the sense that indeed the rectangle follows the camera motion for quite some time with remarkable accuracy. The latter implies that the method described here has a lot of potential if the results are fused with extra feedback such as information from an inertia measurement unit (IMU) or any other sensory information of the sort in a generative manner. In other words, if one "injects" prior information about the camera motion, then the quality of the recoverted odometry and 3D structure should increase significantly.

References

[1] H. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Readings in computer vision: issues, problems, principles, and paradigms,* p. 61, 1987.

[2] R. Hartley*, et al.*, *Multiple view geometry in computer vision* vol. 2: Cambridge Univ Press, 2003.

[3] E. Trucco and A. Verri, *Introductory techniques for 3-D computer vision* vol. 93: Prentice Hall New Jersey, 1998.

[4] O. Faugeras, *Three-dimensional computer vision: a geometric viewpoint*: the MIT Press, 1993.

[5] O. Faugeras*, et al.*, *The geometry of multiple images: the laws that govern the formation of multiple images of a scene and some of their applications*: the MIT Press, 2004.

[6] J. Y. Bouguet, "Pyramidal implementation of the affine lucas kanade feature tracker—description of the algorithm," Technical report). Intel Corporation2001.

[7] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial intelligence,* vol. 17, pp. 185-203, 1981.

[8] M. Z. Brown*, et al.*, "Advances in computational stereo," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 25, pp. 993-1008, 2003.

[9] R. Vidal*, et al.*, "Two-view multibody structure from motion," *International Journal of Computer Vision,* vol. 68, pp. 7-25, 2006.

[10] Q. T. Luong and O. D. Faugeras, "The fundamental matrix: Theory, algorithms, and stability analysis," *International Journal of Computer Vision,* vol. 17, pp. 43-75, 1996.

[11] W. Chojnacki and M. J. Brooks, "On the consistency of the normalized eight-point algorithm," *Journal of Mathematical Imaging and Vision,* vol. 28, pp. 19-27, 2007.

[12] R. I. Hartley, "In defense of the eight-point algorithm," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 19, pp. 580-593, 1997.

[13] Y. Ma*, et al.*, *An invitation to 3-d vision: from images to geometric models* vol. 26: springer, 2003.

[14] B. K. P. Horn, "Recovering baseline and orientation from essential matrix," *J. Optical Society of America,* 1990.

[15] A. OpenGL*, et al.*, "OpenGL programming guide," ed: Addison-Wesley, 1999.

[16] R. Hartley, "Ambiguous configurations for 3-view projective reconstruction," *Computer Vision-ECCV 2000,* pp. 922-935, 2000.

[17] A. Hejlsberg*, et al.*, *C# language specification*: Addison-Wesley Longman Publishing Co., Inc., 2003.

[18] D. Fay, "An architecture for distributed applications on the Internet: overview of Microsoft's. NET platform," in *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, 2003, p. 7 pp.

[19] G. Bradski, "The opencv library," *Doctor Dobbs Journal,* vol. 25, pp. 120-126, 2000.

[20] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM,* vol. 24, pp. 381-395, 1981.