Available online at www.sciencedirect.com
ScienceDirect
Image and Vision Computing 26 (2008) 529–549
www.elsevier.com/locate/imavis

# 3D Motion from structures of points, lines and planes

Andrea Dell'Acqua *, Augusto Sarti, Stefano Tubaro

*Dipartimento di Elettronica e Informazione – Politecnico di Milano, Piazza Leonardo Da Vinci 32, 20133 Milano, Italy*

Received 10 November 2004; received in revised form 6 July 2007; accepted 6 July 2007

**Abstract**

In this article we propose a method for estimating the camera motion from a video-sequence acquired in the presence of general 3D structures. Solutions to this problem are commonly based on the tracking of point-like features, as they usually back-project onto viewpoint-invariant 3D features. In order to improve the robustness, the accuracy and the generality of the approach, we are interested in tracking and using a wider class of structures. In addition to points, in fact, we also simultaneously consider lines and planes. In order to be able to work on all such structures with a compact and unified formalism, we use here the Conformal Model of Geometric Algebra, which proved very powerful and flexible.

As an example of application of our approach, we propose a causal algorithm based on an Extended Kalman Filter, for the estimation of 3D structure and motion from 2D observations of points, lines and coplanar features, and we evaluate its performance on both synthetic and real sequences.


*Keywords:* Camera tracking; Geometric Algebra; Extended Kalman Filter

## 1. Introduction

Recovering camera parameters and 3D scene structure through the analysis of a set of acquired images is a classical Computer Vision problem that has attracted a great deal of attention of researchers from various fields in the past two decades. The literature is, in fact, rich with solutions to this problem, which exploit all sources of geometric and radiometric information.

If we focus just on the techniques based on the geometry of image features, we can roughly classify the solutions available in the literature in two broad categories: those that exploit the multi-view geometric constraints that are involved in the analysis of small subsets of images, and those that are based on the estimation of a dynamical system that captures the motion of the camera in the scene.

The former class of solutions is based on the analysis of pairs or triplets of views of the sequence, and recovers camera motion through the estimation of the two- or three-view Projective geometry (fundamental matrix or trifocal tensor [1–7]). These algorithms are known as batch techniques, and are characterised by a first step of motion and structure estimation from small subsets of views (typically 2–3) subsequently merged through bundle-adjustment or other approaches [8–10]. The most important results of this sort are nicely collected in [11,12] and, more recently, in [13]. The latter category of solutions concerns those causal and recursive algorithms that estimate camera motion and 3D structure through Extended Kalman Filtering (see [14–19]). Such solutions are particularly useful for real-time camera tracking applications, as they do not need to consider all views at the same time, but they can use only information on the past. The state vector is, in fact, upgraded every time a new measurement is available.

Although the theory behind structure and motion estimation methods is well-established, camera tracking is still considered a rather challenging problem in many practical situations. One major source of problems is the quality of feature localization and tracking (accuracy, lifespan, track-

* Corresponding author. Tel.: +39 339 5225679; fax: +39 02 2399 9611.
*E-mail address:* andrea.dellacqu@gmail.com (A. Dell'Acqua).

0262-8856/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.imavis.2007.07.001

ing stability, etc.), as features look different from different viewpoints and occlusions often prevent them from being visible at all. The problem of estimating structure and motion in the presence of missing data (occlusions) has been thoroughly analysed in the literature (see for example [20]). Furthermore, due to the large number of unknowns (structure of the scene, motion parameters as well as camera focal length), in order to overcome the ill-conditioning of the problem it is usually desirable to use as many features and exploit as many constraints as possible in the reconstruction algorithm. For this reason we are interested in considering not just point-like features but also lines and planes in 3D space.

Solutions based on multi-view Projective constraints are available for a variety of image features and structures. An example is [21], which proposes a batch approach that deals with points, lines and planes for the affine camera case. However, no contributions that we are aware of uses mixed features in an EKF-based method.

3D scenes (particularly those of human-made environments) are usually rich with locally straight edges and locally planar surfaces. Linear image features can be easily extracted using (subpixel) edge detectors, while planar surfaces can be identified by structures of points and/or lines or through the analysis of the surface texture over a number of frames [22].

One nice property of line features is that they are less 'local' than points. This means that a line is less likely to be totally occluded and, when it happens, it usually happens quite progressively over a number of frames. As we can expect, this is also true also for planar structures, as long as the plane is directly estimated, and is not just a coplanarity constraint that we force on a set of independently estimated features. We will discuss this again with more detail in Section 6 when describing the EKF-based camera tracking application.

Although the use of line features in camera motion estimation methods has been quite extensively addressed using Projective constraints [11,12,23], lines are usually seen as a source of difficulties. As pointed out in [24], linear algorithms for camera calibration based on line features tend to perform worse than point-based algorithms. Furthermore, using tools from linear algebra it is usually quite difficult to combine mixed features, therefore points and lines are seldom combined. More recently an iterative algorithm that is able to handle the hybrid case was proposed [24], based on the so-called Multiple View matrix. Another recent algorithm exploiting lines is [25], where lines are handled through the so-called Line Motion Matrix.

As far as causal recursive schemes are concerned, some solutions based on line features were recently proposed [26,27], but the representation of lines based on linear algebra causes the algorithm to be quite cumbersome (a more compact representation has been proposed in [25]). A recent algorithm exploiting lines in a real-time visual tracking application is [28].

As far as feature coplanarity is concerned, the literature is rich with solutions that exploit this constraint [29–32]. Such solutions are usually concerned with degeneracy aspects of the multi-view geometry that arise from the constraint itself. Here we are less interested in degeneracy and more focused in exploiting such constraints in a flexible fashion (e.g. several coplanar configurations of features). An effective algorithm for multi-view case exploiting the coplanarity constraint of points is [32], which relies on bundle adjustment techniques. In the case of causal motion estimation, modified versions of [14,15] were recently proposed, which are able to accommodate coplanar point structures [33–35].

In this article we propose a causal camera motion estimation method that is able to use points, lines and planes, and allows us to exploit coplanarity conditions where applicable. This level of flexibility, however, cannot be easily achieved using tools of Linear Algebra (LA), as they would not allow us to handle all the above features and constraints with the same formalism and with a unified framework. For example, rotating sets of points, lines and planes, would require different LA tools, and there would be problems in handling coplanarity conditions for mixed sets of features. One way to overcome these difficulties is to make use of tools of Geometric Algebra (GA) [36–39]. In Geometric Algebra, in fact, all sorts of linear subspaces (lines and planes in Projective space, and also circles and spheres in Conformal space) are easily handled using the same formalism. In particular, GA allows us to adopt a unique formalism for rotating such elements with a minimal parameterization. This is well known to be a crucial issue in minimization algorithms, where the management of constraints between mutually dependent parameters is always a problem. As we will see, GA also allows us to handle intersections between geometric primitives elegantly and compactly. This will allow us to embed coplanarity constraints for points and lines in an EKF structure through a direct estimation of the plane where the features lie. This last issue is quite important as it constitutes another step forward in improving the robustness and the efficiency of motion estimation. Thanks to its capability to easily deal with Euclidean transformations of geometric primitives such as lines or planes, GA has been recently applied with success to a number of Computer Vision tasks, for example the Pose Estimation problem from line observations [40].

## 2. A brief GA introduction

Geometric Algebra provides us with a unified framework that allows us to exploit all the algebraic properties that are known and commonly exploited in Linear Algebra and in Lie groups/algebras, while retaining a insightful geometric interpretation of all the operations. This is made possible by the fact that the elements in GA are not point coordinates but combinations of subspaces, and by the fact

that GA is equipped with an invertible geometric operation called *geometric product*.

This section has primarily a tutorial purpose, and it is aimed at keeping this article as self-contained as possible. For this reason, it can be skipped by those readers who are already familiar with the algebraic aspects of GA. More complete tutorials can be found in [41,42]. For those readers who are interested in more rigorous introduction to GA, we refer to [36]. A collection of applications of GA to Computer Vision can be found in [37].

### 2.1. The extended basis

While a basis of an $n$-dimensional vector space is made of $n$ linearly independent vectors, a basis of the $n$-dimensional Geometric Algebra is a linear space of dimension $2^n$. This basis is also called an *extended basis* as any vector or any higher-grade geometric object can always be expressed as a linear combination of the extended basis elements.

The GA built upon the 3D Euclidean space $\mathbb{E}^3$ is based on scalars (grade-0 objects), vectors (grade-1 objects), oriented area elements called 2-vectors or bivectors (grade-2 objects), and oriented volume elements called *pseudoscalars* or trivectors (grade-3 objects), which are usually denoted by $\mathbf{I}_v$ ($v$ being the dimension of the space). While a vector can be seen as a generator of a line in space and a bivector can be seen as the generator of a plane in space, quite clearly a pseudoscalar can only be seen as the generator of the whole space.

It is important to notice that an arbitrary linear combination of elements of the extended basis is generally a mixed-grade term, which is the reason why it is called *multivector*.

In the Projective case $\mathbb{P}^3$ points of the GA are grade-1 elements (vectors), lines are grade-2 (bivectors), planes are grade-3 (trivectors), while the pseudoscalar $\mathbf{I}_4$ is a 4-grade element that generates the whole space.

The subspaces of a Geometric Algebra are also called *blades*.

In what follows blades of any grade will be denoted by bold letters. In particular, for Euclidean vectors we will use lowercase letters, while for Conformal vectors and higher-grade blades of any space we will use capital letters. We will also use overlined bold letters for normalized Euclidean blades (i.e. vectors whose square is 1, or bivectors whose square is $-1$[1]), while regular vectors and matrices will be denoted by underlined bold letters.

### 2.2. The outer product

A $k$-vector $\mathbf{C}$ can be built with a $i$-vector $\mathbf{A}$ and a $j$-vector $\mathbf{B}$ (if $k = i + j$ and $\mathbf{A}$ and $\mathbf{B}$ have no subspace in common) using a product that implements the span of $\mathbf{A}$ and $\mathbf{B}$. This product is called *outer* or *wedge* product and is denoted by $\wedge$.

For example, given two vectors $\mathbf{a}$ and $\mathbf{b}$, the 2-blade $\mathbf{C} = \mathbf{a} \wedge \mathbf{b}$ in the Euclidean space can be seen as the generator of the linear manifold $S_{\mathbf{a}\wedge\mathbf{b}}$ spanned by $\mathbf{a}$ and $\mathbf{b}$, and its magnitude corresponds to the signed area of the parallelogram defined by such vectors. In the Projective space, $\mathbf{C}$ is the line that passes through the points $\mathbf{a}$ and $\mathbf{b}$.

Using the wedge product we can also test the incidence of two subspaces: given 2 bivectors $\mathbf{A}$ and $\mathbf{B}$, $\mathbf{A} \wedge \mathbf{B} = 0$ if they have a subspace in common.

The outer product is linear, associative and anticommutative:

- $\mathbf{a} \wedge (\mathbf{b} + \mathbf{c}) = \mathbf{a} \wedge \mathbf{b} + \mathbf{a} \wedge \mathbf{c}$;
- $\mathbf{a} \wedge (\mathbf{b} \wedge \mathbf{c}) = (\mathbf{a} \wedge \mathbf{b}) \wedge \mathbf{c}$;
- $\mathbf{a} \wedge \mathbf{b} = -\mathbf{b} \wedge \mathbf{a}$, which implies $\mathbf{b} \wedge \mathbf{b} = 0$.

### 2.3. The inner product

The *inner* product (denoted by $\cdot$) is an operator whose role is symmetrical to that of the outer product. While the outer product is a grade-increasing operator (it combines elements into a higher-grade one), the inner product is a grade-decreasing operator. The inner product between a blade of grade $r$ and a blade of grade $s$ is, in fact, a blade of grade $|r - s|$, therefore the inner product of two vectors is a scalar. When applied to vectors, the inner product coincides with the scalar product in Linear Algebra.

As the inner product of two $r$-blades is always a scalar, we can readily define a metric on the GA space

$$|\mathbf{A}| = \sqrt{\mathbf{A} \cdot \mathbf{A}}.$$

The inner product can be used to define orthogonality between blades

$$\mathbf{A} \perp \mathbf{B} \iff \mathbf{A} \cdot \mathbf{B} = 0.$$

### 2.4. The geometric product

Although the outer and the inner products are not invertible, the GA framework overtakes this issue by introducing an invertible product, the *geometric product*, which allows us to solve geometric equations. The geometric product of two GA elements always returns a mixed-grade *multivector*. In the case of vector elements $\mathbf{a}$ and $\mathbf{b}$, the geometric product takes on the following simple expression

$$\mathbf{ab} = \mathbf{a} \wedge \mathbf{b} + \mathbf{a} \cdot \mathbf{b}, \tag{1}$$

which is the combination of a bivector $\mathbf{a} \wedge \mathbf{b}$ and a scalar $\mathbf{a} \cdot \mathbf{b}$. Quite clearly, the geometric product becomes an outer product when the vectors are orthogonal, and becomes an inner product when the vectors are parallel. The geometric product is associative and distributive with respect to the sum, but it is neither commutative nor anticommutative.

[1] Due to properties of the geometric product that will be introduced later, the square of a bivector is a negative scalar.

One major property of the geometric product is that any $r$-vector has an inverse. Given an $r$-vector $\mathbf{A} = \mathbf{a}_1 \wedge \mathbf{a}_2 \wedge \ldots \wedge \mathbf{a}_r$, its *inverse* is defined as

$$\mathbf{A}^{-1} = \frac{\tilde{\mathbf{A}}}{\mathbf{A}\tilde{\mathbf{A}}}, \tag{2}$$

where

$$\tilde{\mathbf{A}} = \mathbf{a}_r \wedge \ldots \wedge \mathbf{a}_2 \wedge \mathbf{a}_1 \tag{3}$$

is the *reverse* of the blade $\mathbf{A}$. The term $\mathbf{A}\tilde{\mathbf{A}}$ is a non-zero scalar, therefore the inverse is always well defined.

The invertible product is a powerful tool not just because it allows us to solve geometric equations and define derivatives (geometric calculus), but also because it allows us to introduce the concept of duality.

### 2.5. Geometric operations

#### 2.5.1. Duality

Similarly to what happens in LA, the *dual* of a subspace in GA is its complementary space. However, unlike in LA, the fact that GA has an extended basis makes it closed with respect to the duality operator. In other words, the dual is not defined in a 'dual' space, but in the space itself.

In GA the dual of a subspace is computed by dividing the subspace itself by the pseudoscalar. The dual of a blade $\mathbf{A}$ is thus

$$\mathbf{A}^* = \mathbf{A}\mathbf{I}_v^{-1},$$

where the asterisk denotes the dual and $v$ is the dimension of the space.

Notice that the above definition of inverse implies that the inverse of the pseudoscalar is the pseudoscalar itself up to a sign change.[2] As far as the Euclidean space is concerned, we have $\mathbf{I}_3^{-1} = -\mathbf{I}_3$.

#### 2.5.2. Meet

Given two blades $\mathbf{A}$ and $\mathbf{B}$, their *meet* is the highest-grade factor that $\mathbf{A}$ and $\mathbf{B}$ have in common, i.e. their intersection. The meet operator is denoted by a $\vee$ and it can be related to the wedge product through duality

$$(\mathbf{A} \vee \mathbf{B})^* = (\mathbf{A}^* \wedge \mathbf{B}^*), \tag{4}$$

where the dual is evaluated on the lowest-grade space that contains both $\mathbf{A}$ and $\mathbf{B}$. Notice that the above is the geometric version of the De Morgan rule.

#### 2.5.3. Rotation

In GA rotations are geometrically encoded using the Euclidean rotation plane or, alternatively, by the rotation axis (which is the dual of the rotation plane). Such geometric objects are assigned a magnitude that is proportional to the rotation angle.

[2] We are not considering spaces whose basis contain vectors that square to 0. In this case, in fact, the inverse of the pseudoscalar is undefined.

Given a rotation of an angle $\varphi$ on the plane generated by the normalized bivector $\bar{\mathbf{B}}$, we define the *rotor* $\mathbf{R}$ as

$$\mathbf{R} = \mathrm{e}^{-\frac{\varphi}{2}\bar{\mathbf{B}}}.$$

As $\bar{\mathbf{B}}^2 = -1$ we can write

$$\mathbf{R} = \cos\varphi - \bar{\mathbf{B}}\sin\varphi,$$

therefore the rotor is a multivector made of a scalar and a bivector. A rotor satisfies the orthonormality constraint $\mathbf{R}\tilde{\mathbf{R}} = 1$, therefore the inverse (2) of a rotor coincides with its reverse (3). A rotor acts on a blade $\mathbf{X}$ as follows:

$$\mathbf{X} \mapsto \mathbf{R}\mathbf{X}\tilde{\mathbf{R}} = \mathbf{R}\mathbf{X}\mathbf{R}^{-1} = \mathrm{e}^{-\frac{\varphi}{2}\bar{\mathbf{B}}}\mathbf{X}\mathrm{e}^{\frac{\varphi}{2}\bar{\mathbf{B}}}.$$

Notice that two successive rotations operated by the rotors $\mathbf{R}_1$ and $\mathbf{R}_2$ are equivalent to a single rotation performed by the rotor $\mathbf{R} = \mathbf{R}_2\mathbf{R}_1$, which is the geometric product of $\mathbf{R}_1$ and $\mathbf{R}_2$. In fact, we have

$$\mathbf{R}_2\mathbf{R}_1\mathbf{x}\mathbf{R}_1^{-1}\mathbf{R}_2^{-1} = (\mathbf{R}_2\mathbf{R}_1)\mathbf{x}(\mathbf{R}_2\mathbf{R}_1)^{-1} = \mathbf{R}\mathbf{x}\mathbf{R}^{-1}.$$

Notice that any linear combination of the extended basis elements can be rotated by a rotor as shown above, therefore the formalism that we are adopting is of universal validity. For example, given $n$ points $\mathbf{a},\mathbf{b},\ldots\mathbf{z}$ and a generic rotor $\mathbf{R}$, we have

$$\mathbf{R}(\mathbf{a} \wedge \mathbf{b} \wedge \ldots \wedge \mathbf{z})\tilde{\mathbf{R}} = (\mathbf{R}\mathbf{a}\tilde{\mathbf{R}}) \wedge (\mathbf{R}\mathbf{b}\tilde{\mathbf{R}}) \wedge \ldots \wedge (\mathbf{R}\mathbf{z}\tilde{\mathbf{R}}), \tag{5}$$

as $(\mathbf{R}\mathbf{a}\tilde{\mathbf{R}}) \wedge (\mathbf{R}\mathbf{b}\tilde{\mathbf{R}}) = \frac{1}{2}(\mathbf{R}\mathbf{a}\tilde{\mathbf{R}}\mathbf{R}\mathbf{b}\tilde{\mathbf{R}} - \mathbf{R}\mathbf{b}\tilde{\mathbf{R}}\mathbf{R}\mathbf{a}\tilde{\mathbf{R}}) = \frac{1}{2}\mathbf{R}(\mathbf{a}\mathbf{b} - \mathbf{b}\mathbf{a})\tilde{\mathbf{R}} = \mathbf{R}(\mathbf{a} \wedge \mathbf{b})\mathbf{R}$.

The above is a very desirable property that applies to the rigid rotations of feature sets. The extension of this property to the more general case of rigid motions can be achieved by working in the Conformal space, where the translation is a linear operator that is, in fact, performed by a rotor.

## 3. The conformal model in GA

Although the Conformal model was initially introduced by Möbius in his study of the geometry of spheres, GA was applied to such model only recently [36]. For a more complete and rigorous account of Conformal GA, and its applications, we refer the reader to [37,43–46]. In this Section we will only provide a subset of results that are used in this article.

The Conformal model $\mathbb{C}^v$ of a Euclidean space $\mathbb{E}^v$ is obtained by enriching it with a two-dimensional manifold called the Minkowski plane. This procedure is somewhat similar to enriching a space of an additional vector to obtain the corresponding Projective space.

The orthonormal basis $\{\mathrm{e}_+, \mathrm{e}_-\}$ of the Minkowski plane is characterized by the following properties

$$\mathbf{e}_+^2 = +1,\ \mathbf{e}_-^2 = -1,\ \mathbf{e}_+ \cdot \mathbf{e}_- = 0,$$

therefore the Conformal model of the 3D Euclidean space is a 5D space whose basis contains four vectors ($\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$, and $\mathbf{e}_+$) that square to 1, and one vector ($\mathbf{e}_-$) whose square is $-1$.

The particular signature of the basis of the Minkowski plane attributes the Conformal space a number of interesting properties. The Conformal space, for example, contains a *null structure*, which means that the vectors that belong to this structure square to 0 *(null vectors)*. In the Conformal model of the 1D Euclidean space such structure becomes a 3D cone, therefore it takes the name of *null cone* (see Fig. 1) even in spaces of higher dimension.

The Conformal model maps Euclidean points onto null vectors through

$$F(\mathbf{x}) = \alpha(\mathbf{x}^2\mathbf{n} + 2\mathbf{x} - \mathbf{n}_0), \tag{6}$$

where $\mathbf{n} = \mathbf{e}_+ + \mathbf{e}_-$ and $\mathbf{n}_0 = \mathbf{e}_+ - \mathbf{e}_-$.

Notice that this mapping is 'Projective', which means that any Euclidean point is represented by a one-dimensional manifold (a generatrix of the cone). If we choose $\alpha = \frac{1}{2}$, then the mapping (6) becomes one-to-one. In geometric terms we are directing our map onto a slice of the null cone by cutting it with an hyperplane that is orthogonal to $\mathbf{n}_0$ (see Fig. 1). In algebraic terms we are forcing the following normalization condition for null vectors

$$\mathbf{X} \cdot \mathbf{n} = -1. \tag{7}$$

The origin of the Euclidean space is thus mapped onto $\frac{1}{2}\mathbf{n}_0$, and all the points 'at infinity' are mapped onto $\frac{1}{2}\mathbf{n}$.

A geometric interpretation of the Conformal model is provided by the *stereographic projection*, which is commonly used for flattening charts that are defined on a sphere. Let us consider a line representing the 1D Euclidean space $\mathbb{E}^1$ and the circle shown in Fig. 2. The stereographic projection of a point lying on the circle onto $\mathbb{E}^1$ is given by the intersection with $\mathbb{E}^1$ of the ray that originates from the circle's 'north-pole' and passes through the considered point. The inverse mapping, which is very similar to Eq. (6), maps the points at infinity onto the north pole.[3]

### 3.1. Geometric primitives in the Conformal GA

The basic geometric element of the Conformal space is the sphere. As the infinity of the Euclidean space is represented by a point, the Conformal model can be thought of as a 'curved' space in which circles and spheres are linear subspaces. Lines and planes are special instances of such subspaces as they all pass through the point at infinity $\mathbf{n}$.

The Conformal expression of a circle $\mathbf{C}$ passing through three Euclidean points $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ is the trivector

$$\mathbf{C} = F(\mathbf{x}_1) \wedge F(\mathbf{x}_2) \wedge F(\mathbf{x}_3) = \mathbf{X}_1 \wedge \mathbf{X}_2 \wedge \mathbf{X}_3$$

[3] To be precise, there is a difference between the Hestenes map (6) with $\alpha = \frac{1}{2}$ and the inverse stereographic projection. The slice of the cone defined by Eq. (7) is an hyper-paraboloid (a parabola in the $\mathbb{C}^1$ case, as shown in Fig. 1) with axis $\mathbf{n}$, called *horosphere*, while the inverse stereographic projection maps Euclidean points onto a hyper-circular slice (a circle in the $\mathbb{C}^1$ case) of the cone (which is perpendicular to the cone axis $\mathbf{e}_-$). As any Euclidean point is represented in the Conformal model by a generatrix of the cone, such difference is often neglected.

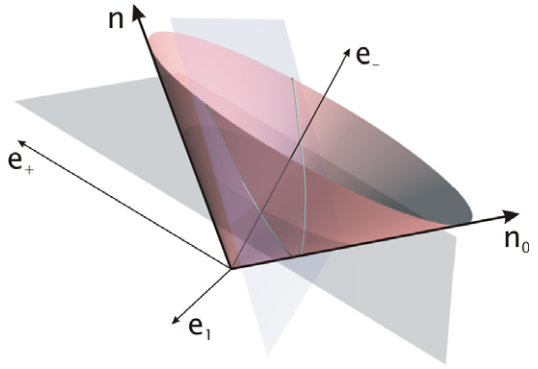


Fig. 1. The 3D Conformal space $\mathbb{C}^1$ is generated by the 1D Euclidean space $\mathbb{E}^1$ ($\mathbf{e}_1$) and the Minkowski plane ($\mathbf{e}_+, \mathbf{e}_-$). This space contains a *null structure*, the so called *null cone*. The vectors $\mathbf{n} = \mathbf{e}_+ + \mathbf{e}_-$ and $\mathbf{n}_0 = \mathbf{e}_+ - \mathbf{e}_-$ (that represents the Euclidean origin) lie on the cone, while the cone axis is the versor $\mathbf{e}_-$. The intersection of the cone with a plane orthogonal to $\mathbf{n}_0$ is a conic called *horosphere* (here a parabola).

while the line $\mathbf{L}$ passing through two Euclidean points $\mathbf{x}_1$ and $\mathbf{x}_2$ is the trivector

$$\mathbf{L} = F(\mathbf{x}_1) \wedge F(\mathbf{x}_2) \wedge \mathbf{n} = \mathbf{X}_1 \wedge \mathbf{X}_2 \wedge \mathbf{n}.$$

Similarly, spheres and planes are quadrivectors. The equation of a plane passing through the points $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ is

$$\Pi = F(\mathbf{x}_1) \wedge F(\mathbf{x}_2) \wedge F(\mathbf{x}_3) \wedge \mathbf{n} = \mathbf{X}_1 \wedge \mathbf{X}_2 \wedge \mathbf{X}_3 \wedge \mathbf{n}.$$

In GA the dual of a geometric primitive encodes its geometric properties [43]. The dual of a line $\mathbf{L}^*$ is the bivector

$$\mathbf{L}^* = \mathbf{L}\mathbf{I}_5^{-1} = \bar{\mathbf{n}}_L\mathbf{I}_3 + [(\bar{\mathbf{n}}_L \wedge \mathbf{o}_L)\mathbf{I}_3]\mathbf{n} = \bar{\mathbf{n}}_L\mathbf{I}_3 + \mathbf{Q}_L\mathbf{I}_3\mathbf{n} = \mathbf{\Omega}_L + \mathbf{q}_L\mathbf{n} \tag{8}$$

which is made of a 'Euclidean' term and a 'Conformal' term. The Euclidean term represents the dual (with respect to the Euclidean space $\mathbf{I}_3$) of the unitary direction $\bar{\mathbf{n}}_L$ (up to a sign change), i.e. the plane $\mathbf{\Omega}_l$ that is perpendicular to the line and passes through the origin (see Fig. 3). The Conformal part contains the dual (again with respect to the Euclidan space) of the *moment* of the line $\mathbf{Q}_L$, i.e. the vector $\mathbf{q}_L$. The moment of the line referred to the origin is the plane defined by the direction of the line and any of the points that lie on the line.

We also define $\mathbf{o}_L$ as the line's *offset*, i.e. the 3D vector that passes through the origin and is perpendicular to the line. Notice that $\mathbf{o}_L$, $\bar{\mathbf{n}}_L$ and $\mathbf{q}_L$ form an orthogonal frame, and that

$$\|\mathbf{q}_L\| = \|\mathbf{o}_L\| = d_L$$

i.e. $d_L$ is the scalar distance from the origin of the line.

The vectors $\bar{\mathbf{n}}_L$ and $\mathbf{q}_L$ are also called *Plücker Coordinates* of the line. The so-called *Plücker constraint* can be simply written as

$$\mathbf{n}_L \perp \mathbf{q}_L \iff \mathbf{n}_L \cdot \mathbf{q}_L = 0 \tag{9}$$

and, together with the constraint on the norm of $\bar{\mathbf{n}}_L$, reduces the number of degrees of freedom of the line parameterization from 6 (the three components of $\mathbf{n}_L$ and $\mathbf{q}_L$) to 4.

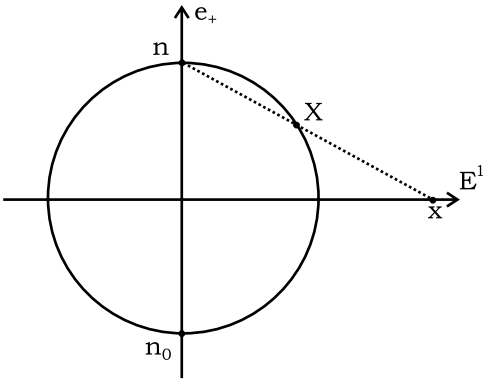


Fig. 2. By cutting the null-cone with an (hyper-)plane orthogonal to its axis $\mathbf{e}_-$ we obtain in general a sphere. In the $\mathbb{C}^1$ case the sphere becomes a circle in the $(\mathbf{e}_+\mathbf{e}_1)$ plane.

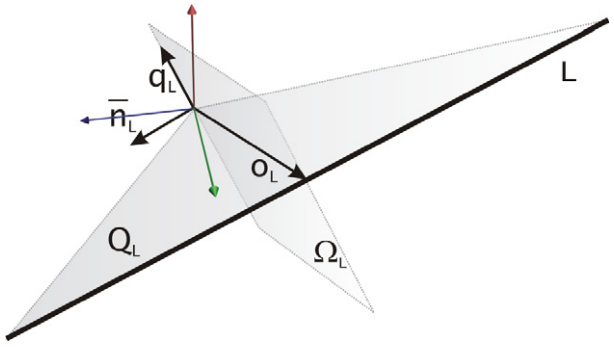


Fig. 3. A 3D line $L$ and the vectors and bivectors that characterize it.

The fact that the dual encodes all the geometric properties of the primitive holds true also for the plane case. Given a plane $\Pi$, its dual with respect to the whole space $\mathbf{I}_5$ is

$$\Pi^* = \bar{\mathbf{n}}_\Pi + d_\Pi \mathbf{n}, \tag{10}$$

where $\bar{\mathbf{n}}_\Pi$ is the unitary vector that is normal to the plane and $d_\Pi$ is the plane's scalar distance from the origin.

We can also define the *offset* of the plane as the vector

$$\mathbf{o}_\Pi = d_\Pi \bar{\mathbf{n}}_\Pi,$$

which can be seen as the translation that must be applied to a plane with normal $\bar{\mathbf{n}}_\Pi$ passing through the origin, in order to obtain the plane $\Pi$.

Notice that the dual (with respect to $\mathbf{I}_5$) of a plane in the Conformal space is a 5D 1-vector, unless the plane passes through the origin. In this case $d_\Pi = 0$ and its dual $\Pi^*$ gives us the normal Euclidean vector $\bar{\mathbf{n}}_\Pi$. Notice also that a plane has three degrees of freedom: two for the normal unitary vector $\bar{\mathbf{n}}_\Pi$ and one for the scalar distance from the origin $d_\Pi$.

### 3.2. Rigid motions in the conformal GA

Rotors act on the Euclidean and the Conformal representations of vectors in a consistent fashion. In fact, given a vector $\mathbf{x}$ in $\mathbb{E}^3$ undergoing a rotation $\mathbf{x} \mapsto \mathbf{R}\mathbf{x}\tilde{\mathbf{R}}$, we have

$$\mathbf{x} \mapsto \mathbf{R}\mathbf{x}\tilde{\mathbf{R}} \Rightarrow F(\mathbf{x}) \mapsto \mathbf{R}F(\mathbf{x})\tilde{\mathbf{R}} = F(\mathbf{R}\mathbf{x}\tilde{\mathbf{R}}), \tag{11}$$

which follows from the fact that rotations leave both the origin and the point at the infinity unchanged, so that $\mathbf{R}\mathbf{n}\tilde{\mathbf{R}} = \mathbf{R}\tilde{\mathbf{R}}\mathbf{n} = \mathbf{n}$ and $\mathbf{R}\mathbf{n}_0\tilde{\mathbf{R}} = \mathbf{n}_0$.

One interesting feature of the Conformal space is that the translation is a linear operation performed by particular rotors called *translators*

$$\mathbf{T}_t = e^{\frac{\mathbf{nt}}{2}} = 1 + \frac{\mathbf{nt}}{2} + \tfrac{1}{2}\left(\frac{\mathbf{nt}}{2}\right)^2 + \cdots = 1 + \frac{\mathbf{nt}}{2}, \tag{12}$$

where $\mathbf{t}$ is the Euclidean translation vector. Notice that the higher-order terms of the translator's expansion are zero as $\mathbf{n}^2 = 0$. The action of a translator on a Conformal vector is

$$\begin{aligned}\mathbf{T}\mathbf{X}\tilde{\mathbf{T}} = &\mathbf{T}F(\mathbf{x})\tilde{\mathbf{T}} = \left(1 + \frac{\mathbf{nt}}{2}\right)\tfrac{1}{2}(\mathbf{x}^2\mathbf{n} + 2\mathbf{x} - \mathbf{n}_0)\left(1 + \frac{\mathbf{tn}}{2}\right)\\ = &\tfrac{1}{2}\Big((\mathbf{x}+\mathbf{t})^2\mathbf{n} + 2(\mathbf{x}+\mathbf{t}) - \mathbf{n}_0\Big) = F(\mathbf{x}+\mathbf{t})\end{aligned}$$

which explains why the component of $\mathbf{X}$ along $\mathbf{n}_0$ is unchanged and, therefore, $\mathbf{X}$ remains on the hyperplane $\mathbf{n}_0 = -\frac{1}{2}$. The Euclidean component is translated by $\mathbf{t}$, while the component along $\mathbf{n}$ increases according to $\frac{1}{2}((\mathbf{x}+\mathbf{t})^2 - \mathbf{x}^2) = \mathbf{x}\mathbf{t} + \frac{\mathbf{t}^2}{2}$. As expected, a translation changes the component of $\mathbf{X}$ along $\mathbf{n}$, as it generally changes the magnitude of the vector.

Notice that, exactly like any other rotors, translators act on lines, circles, planes and spheres. Moreover, the actions of rotors can be combined through geometric multiplication, which returns another rotor. As a consequence, a generic rigid motion can always be expressed as a unique rotor, which is generally called *motor* $\mathbf{M}$. The motor $\mathbf{M}$ that performs a rotation $\mathbf{R}$ followed by a translation $\mathbf{t}$ is

$$\mathbf{M} = \mathbf{T}_t\mathbf{R},$$

where $\mathbf{T}_t = e^{\frac{\mathbf{nt}}{2}}$, and its action on a point $\mathbf{X}$ is

$$\mathbf{X} \to \mathbf{X}' = \mathbf{M}\mathbf{X}\mathbf{M}^{-1} = \mathbf{M}\mathbf{X}\tilde{\mathbf{M}}.$$

This expression holds true for more complex motions and primitives. For example, the rotation of a line $L$ about a point $P$ is performed by the motor

$$\mathbf{M}_P = \mathbf{T}_P\mathbf{R}\tilde{\mathbf{T}}_P$$

via

$$\mathbf{L} \to \mathbf{L}' = \mathbf{M}_P\mathbf{L}\tilde{\mathbf{M}}_P = \mathbf{T}_P\mathbf{R}\tilde{\mathbf{T}}_P\ \mathbf{L}\mathbf{T}_P\tilde{\mathbf{R}}\tilde{\mathbf{T}}_P,$$

where $\mathbf{T}_P = e^{\frac{\mathbf{nt_p}}{2}}$ and $\mathbf{t_p}$ is the point's offset from the origin.

## 4. Projections onto a moving camera

A projection relation is an equation that relates a 3D primitive to its projection onto the image plane of a camera. Given a pin-hole camera $(i)$, its optical center $\mathbf{C}^{(i)}$

and the directions of its three axes $\mathbf{X}_j^{(i)}$, $j=1,2,3$, it is possible to recover the expressions of the camera axes $\rho_j^{(i)} = \mathbf{C}^{(i)} \wedge \mathbf{X}_j^{(i)} \wedge \mathbf{n}$ and planes $\mathbf{\Pi}_{j_1}^{(i)} = \mathbf{C}^{(i)} \wedge \mathbf{X}_{j_2}^{(i)} \wedge \mathbf{X}_{j_3}^{(i)} \wedge \mathbf{n}$ with $\{j_1,j_2,j_3\} \in \{1,2,3\}, \{2,3,1\}, \{3,1,2\}$.

Let us consider a point $P$ and a line $L$ in the 3D Euclidean space $\mathbb{E}^3$ and their Conformal representations $\mathbf{P}$ and $\mathbf{L}$. Let $p_j^{(i)}$ and $l_j^{(i)}, i=0,\ldots,N-1$, $j=1,2,3$ be the image-plane coordinates of the projections respectively of the point $P$ and the line $L$ onto $N$ cameras. In this paper we consider a camera with focal length $f=1$. When $f\neq 1$, the equations that we will obtain will hold true by consistently rescaling $\underline{\mathbf{p}}^i$ and $\underline{\mathbf{l}}^i$.

The line (plane) passing through the camera center and the image plane coordinates of a point (line) is called 'back-projection ray (plane)'. A ray back-projected from the $i$th image plane can be expressed as a linear combination of camera-$i$ reference axes

$$\mathbf{\Psi}_P^{(i)} = \sum_{j=1}^{3} p_j^{(i)} \rho_j^{(i)}.$$

Similarly, a plane back-projected from the $i$th image plane can be expressed as a linear combination of camera-$i$ reference planes

$$\mathbf{\Phi}_L^{(i)} = \sum_{j=1}^{3} l_j^{(i)} \Pi_j^{(i)}.$$

These expressions can be written in terms of camera-$i$ rotation $\mathbf{R}^{(i)} = \mathrm{e}^{-\frac{\theta^{(i)}}{2}\mathbf{B}^{(i)}}$ and translation $\mathbf{T}^{(i)} = \mathrm{e}^{\frac{\mathbf{n}t^{(i)}}{2}}$ as follows

$$\mathbf{\Psi}_P^{(i)} = \sum_{j=1}^{3} p_j^{(i)} \left( \mathbf{T}^{(i)} \mathbf{R}^{(i)} \rho_j^{(0)} \tilde{\mathbf{R}}^{(i)} \tilde{\mathbf{T}}^{(i)} \right), \tag{13}$$

$$\mathbf{\Phi}_L^{(i)} = \sum_{j=1}^{3} l_j^{(i)} \left( \mathbf{T}^{(i)} \mathbf{R}^{(i)} \mathbf{\Pi}_j^{(0)} \tilde{\mathbf{R}}^{(i)} \tilde{\mathbf{T}}^{(i)} \right) \tag{14}$$

where $\rho_j^{(0)}$ and $\mathbf{\Pi}_j^{(0)}$ are the reference camera axes and planes, usually the first of the set. In what follows we will assume this camera to be placed in the origin of the world and oriented like the world reference system.

Back-projection rays and planes can also be expressed as a function of the 3D structure:

$$\begin{aligned} \breve{\mathbf{\Psi}}_P^{(i)} &= \mathbf{C}^{(i)} \wedge \mathbf{P} \wedge \mathbf{n} = (\mathbf{T}^{(i)} \mathbf{C}^{(0)} \tilde{\mathbf{T}}^{(i)}) \wedge \mathbf{P} \wedge \mathbf{n}, \\ \breve{\mathbf{\Phi}}_L^{(i)} &= \mathbf{C}^{(i)} \wedge \mathbf{L} = (\mathbf{T}^{(i)} \mathbf{C}^{(0)} \tilde{\mathbf{T}}^{(i)}) \wedge \mathbf{L}, \end{aligned} \tag{15}$$

i.e. as lines and planes joining the primitive and the center of the $i$th camera.

Camera motion, structure and projections are related by $\mathbf{\Psi}_P^{(i)} \simeq \breve{\mathbf{\Psi}}_P^{(i)}$ and $\mathbf{\Phi}_L^{(i)} \simeq \breve{\mathbf{\Phi}}_L^{(i)}$, where $\simeq$ means 'equal up to a scale factor'. Such 'Projective equality' constraints can be interpreted as parallelism constraints between Euclidean vectors, by applying the camera motion that takes the $i$th camera into the reference one to the back-projection rays and planes.

In fact, we have

$$\tilde{\mathbf{R}}^{(i)} \tilde{\mathbf{T}}^{(i)} \mathbf{\Psi}_P^{(i)} \mathbf{T}^{(i)} \mathbf{R}^{(i)} = \sum_{j=1}^{3} p_j^{(i)} \rho_j^{(0)},$$

$$\tilde{\mathbf{R}}^{(i)} \tilde{\mathbf{T}}^{(i)} \mathbf{\Phi}_L^{(i)} \mathbf{T}^{(i)} \mathbf{R}^{(i)} = \sum_{j=1}^{3} l_j^{(i)} \Pi_j^{(0)}$$

and

$$\begin{aligned} \tilde{\mathbf{R}}^{(i)} \tilde{\mathbf{T}}^{(i)} \breve{\mathbf{\Psi}}_P^{(i)} \mathbf{T}^{(i)} \mathbf{R}^{(i)} &= \tilde{\mathbf{R}}^{(i)} \left( \mathbf{C}^{(0)} \wedge (\tilde{\mathbf{T}}^{(i)} \mathbf{P} \mathbf{T}^{(i)}) \wedge \mathbf{n} \right) \mathbf{R}^{(i)}, \\ \tilde{\mathbf{R}}^{(i)} \tilde{\mathbf{T}}^{(i)} \breve{\mathbf{\Phi}}_L^{(i)} \mathbf{T}^{(i)} \mathbf{R}^{(i)} &= \tilde{\mathbf{R}}^{(i)} \left( \mathbf{C}^{(0)} \wedge (\tilde{\mathbf{T}}^{(i)} \mathbf{L} \mathbf{T}^{(i)}) \right) \mathbf{R}^{(i)} \end{aligned} \tag{16}$$

which are lines and planes that pass through the origin.

Lines and planes passing through the origin are completely specified by one Euclidean vector only: the direction in the line case and the normal in the plane case. Two lines are coincident if their directions coincide up to a scale factor, and the same is true for two planes and their normals.

The inner product of a line with $\mathbf{E} = \mathbf{e}_+ \mathbf{e}_-$ can be used for obtaining the line's direction from its Conformal expression. From the definition

$$\mathbf{p}^{(i)} = \sum_{j=1}^{3} p_j^{(i)} \mathbf{e}_j = \left( \sum_{j=1}^{3} p_j^{(i)} \rho_j^{(0)} \right) \cdot \mathbf{E}$$

we can write

$$\mathbf{\Psi}_P^{(i)} \simeq \breve{\mathbf{\Psi}}_P^{(i)} \Longleftrightarrow \mathbf{p}^{(i)} \wedge \left[ \tilde{\mathbf{R}}^{(i)} (\mathbf{C}^{(0)} \wedge \tilde{\mathbf{T}}^{(i)} \mathbf{P} \mathbf{T}^{(i)} \wedge \mathbf{n}) \mathbf{R}^{(i)} \right] \cdot \mathbf{E} = 0. \tag{17}$$

The Euclidean normal of two Conformal planes passing through the origin is simply their dual, therefore, by defining the vector

$$\mathbf{l}^{(i)} = \sum_{j=1}^{3} l_j^{(i)} \mathbf{e}_j = \left( \sum_{j=1}^{3} l_j^{(i)} \mathbf{\Pi}_j^{(0)} \right)^* \tag{18}$$

we obtain

$$\mathbf{\Phi}_L^{(i)} \simeq \breve{\mathbf{\Phi}}_L^{(i)} \Longleftrightarrow \mathbf{l}^{(i)} \wedge \left[ \tilde{\mathbf{R}}^{(i)} (\mathbf{C}^{(0)} \wedge \tilde{\mathbf{T}}^{(i)} \mathbf{L} \mathbf{T}^{(i)}) \mathbf{R}^{(i)} \right]^* = 0. \tag{19}$$

The projection Eqs. (17) and (19) are given in terms of the Conformal representations $\mathbf{P}$ and $\mathbf{L}$ of the point and the line, respectively. It is also possible to recover the corresponding equations as a function of the Euclidean vectors that characterize the primitive, i.e. the offset from the origin $\mathbf{o}_\mathrm{P}$ of the point $\mathbf{P}$ and the vectors $\mathbf{o}_L$, $\bar{\mathbf{n}}_L$ and $\mathbf{q}_L$ (Eq. (8)) of the line $\mathbf{L}$.

The point $\tilde{\mathbf{T}}^{(i)} \mathbf{P} \mathbf{T}^{(i)}$ is the Conformal representation of $\mathbf{o}_P - \mathbf{t}^{(i)}$, which is also the direction of the line $\mathbf{C}^{(0)} \wedge \tilde{\mathbf{T}}^{(i)} \mathbf{P} \mathbf{T}^{(i)} \wedge \mathbf{n}$, therefore Eq. (17) can be rewritten as follows

$$\mathbf{p}^{(i)} \wedge (\tilde{\mathbf{R}}^{(i)} (\mathbf{o}_P - \mathbf{t}^{(i)}) \mathbf{R}^{(i)}) = 0. \tag{20}$$

In the line case we have

$$\begin{aligned} (\mathbf{C}^{(0)} \wedge (\tilde{\mathbf{T}}^{(i)} \mathbf{L} \mathbf{T}^{(i)}))^* &= (\mathbf{n}_L \wedge -\mathbf{t}^{(i)}) \mathbf{I}_3 + \mathbf{q}_L \\ &= (\mathbf{n}_L \wedge (\mathbf{o}_L - \mathbf{t}^{(i)})) \mathbf{I}_3 \end{aligned} \tag{21}$$

which means that by applying the translation $-\mathbf{t}^{(i)}$ to a line $L$ we obtain a line $\mathbf{n}_L$ with the same direction and an offset of $\mathbf{o}_L - \mathbf{t}^{(i)}$.

At this point Eq. (21) can be rewritten in terms of just Euclidean vectors as follows

$$\mathbf{l}^{(i)} \wedge (\tilde{\mathbf{R}}^{(i)}((\mathbf{t}^{(i)} \wedge \mathbf{n}_L)\mathbf{I}_3 + \mathbf{q}_L)\mathbf{R}^{(i)}) = 0. \tag{22}$$

## 5. Defining a general cost function

Let us consider a structure of $H$ points and $K$ lines and the image coordinates of their projections onto $N$ views. The problem of estimating camera motion and structure can be generally expressed as the search of the minimum of the following cost function

$$\begin{aligned}
\bar{\hat{\mathbf{l}}}_{\mathbf{k}}^{(i)} &= [\tilde{\mathbf{R}}^{(i)}(\mathbf{C}^{(0)} \wedge \tilde{\mathbf{T}}^{(i)}\mathbf{L}_k\mathbf{T}^{(i)})\mathbf{R}^{(i)}]^*;\\
\bar{\hat{\mathbf{p}}}_{\mathbf{h}}^{(i)} &= [[\tilde{\mathbf{R}}^{(i)}(\mathbf{C}^{(0)} \wedge \tilde{\mathbf{T}}^{(i)}\mathbf{P}_h\mathbf{T}^{(i)} \wedge \mathbf{n})\mathbf{R}^{(i)}] \cdot \mathbf{E}];\\
F &= w_L F_L + w_P F_P\\
&= -w_L \sum_{k=1}^{K} \left\{ \bar{l}_k^{(i)} \wedge \bar{\hat{\mathbf{l}}}_{\mathbf{k}}^{(i)} \right\}^2 - w_P \sum_{h=1}^{H} \left\{ \bar{p}_h^{(i)} \wedge \bar{\hat{\mathbf{p}}}_{\mathbf{h}}^{(i)} \right\},
\end{aligned} \tag{23}$$

where $w_L$ and $w_P$ are scale factors designed to weigh differently the contribution of points and lines to the global cost, and the minus sign is required because the square of a bivector is always negative. Since the factors of the outer product have unit norm, this cost function encodes directly the sine of the angles between the back-projection rays (in the point case) and between the normals of the back-projection planes (in the line case). Notice also that the first part of Eq. (23) is equivalent to the cost function of *Lin_2D_1* alignment method proposed in [25]. As pointed out in [25] it is possible to devise cost functions with more physical meaning also in the line case (i.e. involving pixel coordinates and geometric distances between a line and points lying on another line), at the price of complicating the implementation.

This problem formulation is independent of the parameterizations chosen for points $\mathbf{P}_h$ and lines $\mathbf{L}_k$. For example, a point can be parameterized through its 3D coordinates (using Eq. (22)) or through its 2D projections onto a camera and the distance from the camera center. Moreover, geometric scene constraints can be embedded in the problem in a rather straightforward fashion. In fact, a line $\mathbf{L}$ can always be parameterized as the intersection of two planes, while a point $\mathbf{P}$ can be seen as the intersection of two lines, or a plane and a line, or three planes. This can be used for reducing the number of d.o.f. on constrained scenes. For example, if we are tracking the edges of an imaged cube, then we can decide to parameterize the scene with the cube's faces instead. This results in a reduction of the number of d.o.f. from 48 (12 lines) to 18 (6 planes).

More interesting is the possibility to represent generic coplanar features as the intersection between the back-projection of their view (a ray or a plane) and their coplanarity plane. Let us consider a point $\mathbf{P}$ and a line $\mathbf{L}$ lying on the plane $\Pi$. If the ray $\Psi_P^{(0)}$ and the plane $\Phi_L^{(0)}$ are the back-projections from the first view, then the 3D primitives can be parameterized as follows

$$\mathbf{P} = \Psi_P^{(0)} \vee \Pi, \tag{24}$$

$$\mathbf{L} = \Phi_L^{(0)} \vee \Pi. \tag{25}$$

The estimation of the plane $\Pi$ within a minimization scheme aimed at recovering the structure can be performed directly. Each plane of the scene contributes 3 d.o.f., thus the total number of d.o.f is independent of course from the number of observed features. As the estimates of the 3D locations of the primitives are affected by the localization error on the image-plane coordinates that define $\Psi_P^{(0)}$ and $\Phi_L^{(0)}$, such coordinates can be re-estimated, which adds two additional d.o.f. for each observed feature.

## 6. A causal scheme for structure and motion recovering

The cost function (23) can be minimized using a *causal scheme*. An optimization algorithms is causal when it only uses information from the 'past'. For example, for the estimation at time $i$ it uses the image location of the features extracted from the views at times $0,\ldots,i-1$. Such schemes are desirable in a variety of applications. For example, applications of motion control of mobile robots in cluttered environments require 3D scene and camera motion to be estimated in real time. Computational efficiency, however, is not desirable only in real-time applications, but in all applications of commercial interest. An example of this sort is the fast visualization of augmented reality, the pre-visualization mixed (real-virtual) realities for high-end visual applications (TV and cinema production), etc.

As pointed out in the introduction, several recursive schemes have been proposed in the literature in the past two decades. In this article we start from the approach of [14–18], which consists in modelling the projection process as a dynamical system whose state collects all the problem unknowns (camera motion and 3D feature coordinates). The state estimator is an Extended Kalman Filter [47,48], driven by the image features extracted from the acquired video sequence. The structure of this algorithm is basically quite simple: we start from the last prediction of the camera motion and the last estimation of the scene structure, available at time $i$, and we compute the 2D image feature projections on the current view. These projections are then compared with the observed ones. The difference between the predicted and the observed image coordinates, weighted by the Filter gain, are thus used for refining the estimates, and so on.

In this article we do not address the 2D feature tracking problem. We assume, in fact, that feature tracking is performed by a pre-processing module that detects and tracks both 2D points (corners) and lines (straight edges).

Our approach considers lines as global geometric entities as it does not use any information on the endpoints

of the visible segment. Moreover, we do not assume points and lines to be visible throughout the entire sequence as recursive algorithms can deal with occlusions by removing the no-longer-visible primitives and by including new ones in the EKF state vector [16–18].

Many causal schemes have been proposed in the literature, which address all sorts of features and constraints. The approach that we propose here, however, is characterized by several points of novelty. First of all we adopt a unified approach for point and line features, with compact formulations of intersection relations between geometric primitives. The tools of GA enable us to handle coplanarity constraints on points and lines in the same general framework. Such constraints are naturally enforced into an EKF structure by directly estimating the plane where the points and lines lie.

From an algebraic standpoint, our results for point coplanarity is similar to those proposed in [35]. Our approach, however, is more general as it can be naturally and quite straightforwardly extended to the case of line coplanarity. Furthermore, parameterizing rotations with rotors turns out to be a particularly effective choice as it is minimal and naturally embedded into the framework. This parameterization is crucial for the motion estimation problem because of the well-known difficulties in enforcing nonlinear constraints on state variables.

### 6.1. Camera motion parameterization

Camera motion is here described by six parameters: the components of the translation vector $\mathbf{t}$, which appear in the translator as $\mathbf{T} = e^{\frac{\mathbf{t}}{2}}$; and the rotation bivector $\mathbf{B} = \frac{\varphi}{2}\bar{\mathbf{B}}$, which represents the rotation plane scaled by the rotation angle.

A rotation is usually represented in linear algebra by a matrix $\underline{\mathbf{R}}$, which is bound to be orthogonal with unit determinant ($\underline{\mathbf{R}}^{\mathrm{T}}\underline{\mathbf{R}} = \underline{\mathbf{R}}\underline{\mathbf{R}}^{\mathrm{T}} = 1$). This parameterization, however, is redundant as the nine entries of $\underline{\mathbf{R}}$ are not independent (orthonormality constraint).

In order to overcome this difficulty [14,15] suggest to use quaternions, which are isomorphic to rotors, to parameterize camera rotations in the EKF. A quaternion $(q_0, q_1, q_2, q_3)$, however, needs to satisfy the nonlinear normalization constraint $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$, enforcing which is difficult in an algorithm such as the EKF, which is based on a step-by-step linearization. For this reason [14,15] adopt a mixed parameterization: quaternions to describe the global rotation (rotation between the first and the $(i-1)th$ camera), whereas in the state vector the incremental rotation between the $(i-1)th$ and the $ith$ camera is described by a triplet of euler angles.

Using GA we can adopt a unique parameterization for rotations without enforcing any additional constraint in the state estimation process, as the rotation is directly encoded by the bivector $\mathbf{B}$. Geometric Calculus in GA allows us to compute the derivatives of an expression containing rotors with respect to its exponent $\mathbf{B}$.

In [16–18] rotations are parameterized using exponential coordinates, which is typical in Lie Groups and Lie Algebras [49]. This parameterization is equivalent to GA's.

### 6.2. The prediction model

Motion parameters are to be considered as *dynamical* while the scene rigidity assumption suggests that the structure parameters are *static*. The estimation of static parameters is thus expected to converge to a constant value after a transient and their a-priori estimate at time $i$ is equal to the value estimated at time $i-1$.

In general, in the absence of any a-priori information on the camera motion (e.g. constant velocity, planar motion, constrained trajectory, etc.), it is impossible to predict the evolution of the dynamic parameters of the EKF state. The prediction model for motion parameters proposed in the literature [14–16,18] is thus the *random walk*. According to this model, our best guess for the current camera position and orientation are the previous camera position and orientation. As a consequence, the new measurements $\mathbf{y}^{(i)}$ must take care of correctly updating the camera viewpoint. As a consequence, static and dynamic parameters are treated exactly in the same way, although there is a big difference between them: for the static parameters the random walk model is adequate, whereas for the dynamic parameters that model is quite poor. Nonetheless, the performance of structure and motion estimators based on this model are remarkable, as shown in the literature.

A more sophisticated model, proposed in [16–18], uses a larger state vector that also includes the 3D velocity vectors $\mathbf{v}$ (translational velocity) and $\omega$ (rotational velocity). In this case the random walk prediction model is adopted for velocities instead of camera position and orientation. This means that, at time $i$, the camera is assumed to keep moving with the same velocity that it had between the time steps $i-2$ and $i-1$. This assumption guarantees a certain continuity in the prediction of the motion parameters, and seems more reasonable in the presence of smooth trajectories.

### 6.3. State-measurement relations

The EKF structure allows us to deal with state-measurements relations that are nonlinear and in implicit form, such as (17) and (19). This is possible through linearization of such relations with respect to the state variables and the measurements. However, it is advisable to write Eqs. (17) and (19) in explicit form, in order to avoid having to compute the derivatives with respect to the measurements and speed up the process. We follow this approach in the implementation of our algorithm.

In the point case this operation is quite straightforward as we can divide both terms of the wedge product by their

third component (which is zero only for the points at infinity on the image plane). By setting

$$\chi = [\tilde{\mathbf{R}}^{(i)}(\mathbf{C}^{(0)} \wedge \tilde{\mathbf{T}}^{(i)}\mathbf{P}\mathbf{T}^{(i)} \wedge \mathbf{n})\mathbf{R}^{(i)}] \cdot \mathbf{E}, \tag{26}$$

the predicted image-plane coordinates will be

$$\left[\hat{p}_1^{(i)} \quad \hat{p}_2^{(i)} \quad \hat{p}_3^{(i)}\right] = \left[\frac{\chi_1}{\chi_3} \quad \frac{\chi_2}{\chi_3} \quad 1\right]. \tag{27}$$

This is what will be compared with the observed coordinates $p_j^{(i)}, j = 1, 2, 3$.

In the line case any of the three coordinates could become zero, but we can still split the set of 3D lines of the structure into two subgroups: one including the lines whose image-plane coordinates satisfy $|l_1^{(i)}| \geqslant |l_2^{(i)}|$, the other including the remaining lines for which $|l_1^{(i)}| < |l_2^{(i)}|$ (see Fig. 4). This grouping must be re-evaluated at each frame.

By setting

$$\psi = [\tilde{\mathbf{R}}^{(i)}(\mathbf{C}^{(0)} \wedge \tilde{\mathbf{T}}^{(i)}\mathbf{L}\mathbf{T}^{(i)})\mathbf{R}^{(i)}]^*, \tag{28}$$

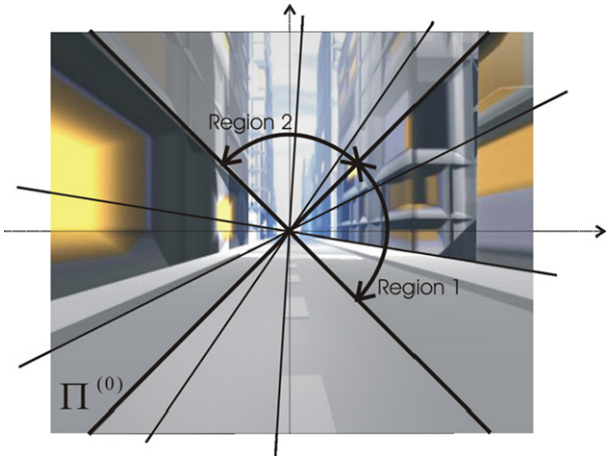


Fig. 4. The lines of the structure are divided in two regions. Region 1 includes lines whose image-plane coordinates satisfy $|l_1^{(0)}| \geqslant |l_2^{(0)}|$, region 2 is made of the remaining lines, for which $|l_1^{(0)}| < |l_2^{(0)}|$ (this picture shows only those lines that pass throught the origin ($l_3^{(0)} = 0$) but this subdivision applies to any line in the image plane).

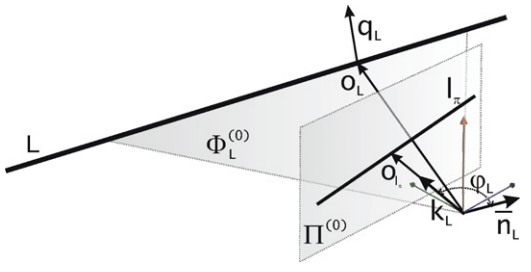


Fig. 5. A line $\mathbf{L}$ and its projection $\mathbf{l}_\Pi$ onto the image plane $\Pi^{(0)}$. The vectors $\mathbf{o}_L$, $\mathbf{o}_{l_\Pi}$ (the offset of $\mathbf{l}_\Pi$ from the origin) and $\bar{\mathbf{n}}_L$ lie on the back-projection plane $\Phi_L^{(0)}$. The angle between $\bar{\mathbf{n}}_L$ and $\mathbf{o}_{l_\Pi}$ (or its normalized version $\bar{\mathbf{k}}_L$) is $\varphi_L$.

the predicted image-plane coordinates will be

$$\left[\hat{l}_1^{(i)} \quad \hat{l}_2^{(i)} \quad \hat{l}_3^{(i)}\right] = \left[1 \quad \frac{\psi_2}{\psi_1} \quad \frac{\psi_2}{\psi_1}\right], \quad \text{if } |l_1^{(i)}| \geqslant |l_2^{(i)}|; \tag{29}$$
$$\left[\hat{l}_1^{(i)} \quad \hat{l}_2^{(i)} \quad \hat{l}_3^{(i)}\right] = \left[\frac{\psi_1}{\psi_2} \quad 1 \quad \frac{\psi_3}{\psi_2}\right], \quad \text{if } |l_1^{(i)}| < |l_2^{(i)}|.$$

Quite obviously, the image-plane coordinates provided by the tracking algorithms need to be consistently rescaled in order to enable a comparison with Eqs. (27) and (29).

### 6.4. Structure parameterization

A point or a line can be parameterized through their 3D coordinates or, in alternative, through the coordinates of their projections onto the image plane $\Pi^{(0)}$ of the reference camera plus some 3D information.

In the point case the 3D location of a point $P$ is a function of the image-plane coordinates $\mathbf{p}^{(0)}$ that describes the distance from the origin $d_P$ or the depth $\alpha_P$

$$\mathbf{o}_P = d_P\bar{\mathbf{p}}^{(0)} = \alpha_P\mathbf{p}^{(0)}, \tag{30}$$

where $\mathbf{p}^{(0)}$ lies on the image plane, i.e. $p_3^{(0)} = 1$ . In the line case, a line $L$ is specified by two vectors: the direction $\bar{\mathbf{n}}_L$ and the dual of the momentum $\mathbf{q}_L = \mathbf{Q}_L\mathbf{I}_3^{-1}$. Since the reference camera lies in the origin, the back-projection plane $\Phi_L^{(0)}$ and the momentum of the line $\mathbf{Q}_L$ represent the same plane. The normal to the back-projection plane $\Phi_L^{(0)}$ is $\mathbf{n}_{\Phi_L}^{(0)} = \mathbf{l}^{(0)} = \sum_{j=1}^3 l_j^{(0)}\mathbf{e}_j$ (Eq. (18)), therefore we have

$$\mathbf{q}_L = d_L\bar{\mathbf{l}}^{(0)}, \tag{31}$$

where $\bar{\mathbf{l}}^{(0)}$ is the normalized version of $\mathbf{l}^{(0)}$.

The strategy of parameterizing a 3D primitive through its projection, suggested in [14–16] for the point case, is a clever one as it enables the algorithm to take advantage of the a-priori information that is available on the location of the primitives. We know, in fact, that a 3D point (line) lies on its back-projection ray (plane).[4] The Kalman Filter allows us to take the a-priori information on a parameter $j$ into account by fine-tuning the initial variance of its estimate accordingly. Through this parameterization it is possible to concentrate most of the initial uncertainty along the direction of depth, rather than uniformly spreading it over all three components of the 3D point locations.

There is one more advantage to this parameterization: the number of parameters that describe a 3D primitive could be reduced by removing the image coordinates from the state vector. This solution considerably reduces the dimensionality of the problem, and significantly speeds up the algorithm. This choice [14,15], however, is heavily criticized in [16–18], since it considers the measurements on the first image of the video sequence as noise-free, and therefore it introduces a significant bias in the estimates.

As a matter of fact, 'freezing' the 2D coordinates of the projected points to their measured value violates the zero-mean assumption that is inherent in the Kalman Filter,

[4] Due to measurement noise this is only approximately true.

with the result that the Filter could easily diverge, as shown in [16]. It is important to notice, however, that allowing all the 2D coordinates on the reference image to vary makes the model unobservable, i.e. there are infinite configurations of structure and motion that explain the measurements at any time $i$. The model becomes observable by fixing the 2D coordinates of 3 points on the reference frame (plus one depth to remove the scale factor uncertainty in the reconstruction) [16–18].

*6.4.1. A closer look to the line case*

As pointed out in Section 3.1, lines are not generic trivectors in $\mathbb{C}^3$ as the line parameterization has only 4 d.o.f. The compact Conformal expression (19) can not be used for estimating the line structure, but it can only be used for predicting the new image-plane coordinates. In order to account for the non linear constraints on the Euclidean vectors $\bar{\mathbf{n}}_L$ and $\mathbf{q}_L$ ($\|\bar{\mathbf{n}}_L\| = 1, \bar{\mathbf{n}}_L \cdot \mathbf{q}_L = 0$) we need to switch to the Euclidean expression (22).

Nonetheless, in minimization algorithms such as the EKF it is preferable to adopt a minimal parameterization rather than forcing a constraint at each step. An effective one consists of expressing the direction $\bar{\mathbf{n}}_L$ as a rotated version of a known vector $\bar{\mathbf{k}}_L$ lying on the plane $\mathbf{Q}_L$ (Fig. 5)

$$\bar{\mathbf{n}}_L = \mathrm{e}^{-\frac{\varphi_L}{2}\mathbf{Q}_L}\bar{\mathbf{k}}_L\,\mathrm{e}^{\frac{\varphi_L}{2}\mathbf{Q}_L} = \mathrm{e}^{-\varphi_L\mathbf{Q}_L}\bar{\mathbf{k}}_L.$$

Notice that, when a vector lies on the rotation plane, the action of a rotor turns out to be greatly simplified. $\bar{\mathbf{k}}_L$ can be chosen to be the (normalized) vector that is perpendicular to the line projection $\bar{\mathbf{n}}_L$ onto the image plane $\Pi^{(0)}$

$$\mathbf{k}_L = -l_1^{(0)} l_3^{(0)}\mathbf{e}_1 - l_2^{(0)} l_3^{(0)}\mathbf{e}_2 + \left((l_1^{(0)})^2 + (l_2^{(0)})^2\right)\mathbf{e}_3. \tag{32}$$

As $\bar{\mathbf{k}}_L$ can be shown to be a function of just the image coordinates $\mathbf{l}^{(0)}$, $\bar{\mathbf{k}}_L$ is a function of just the angle $\varphi_L$. A line $L$ is thus parameterized by the scalars $\varphi_L$ and $d_L$. This information can be completed with the image-plane coordinates $\mathbf{l}^{(0)}$, if they are treated as problem unknowns.

This line parameterization is similar to the one proposed in [27].

It is important to remember that lines behave quite differently from points, as they do not allow us to define any geometric constraint on the camera motion from just one pair of views. In fact, given a pair of images and a set of stereo-corresponding line projections, the corresponding back-projections planes are always incident for any displacement and orientation of the cameras. Line constraints that are useful for EKF estimation and prediction require a minimum of three simultaneous views.

A possible solution, proposed in [27], is based on a three-step estimation scheme: given the current view and two new views, the interframe rotation is computed from the line observations; then the translation is recovered by keeping the rotation fixed; finally the structure is computed from the motion with the help of the a-priori estimate, if available.

As this multi-step estimation approach does not easily agree with an EKF structure, we propose in this article another approach, directly derived from the trifocal constraint. Three back-projection planes corresponding to the same line form a pencil (given the 3 planes, the line of intersection between any two of them must lie on the third one). One possibility is to model a line as the intersection of the back-projection planes from two cameras, for example the reference one (0) and an arbitrary one ($m$)

$$\mathbf{L} = \Phi_L^{(0)} \vee \Phi_L^{(m)}.$$

The line obtained from this expression is projected onto the current view ($i$), where $i > m$, in order to update the filter estimation (see Fig. 6).

A back-projection plane can always be expressed as a linear combination of the reference camera planes through the image-plane coordinates of the line. As the camera planes depend on the camera motion (see Eq. (14)), all the back-projection planes from the *mth* camera can be expressed only through the camera translation and rotation. A structure of $K$ lines is thus parameterized by only 6 parameters plus the image-plane coordinates $\bar{\mathbf{l}}_k^{(0)}$ and $\bar{\mathbf{l}}_k^{(m)}$, $k = 1, \ldots, K$, if they are seen as unknowns.

Like in the point case, in order to make the model observable, we need to fix a number of image-plane coordinates. The problem of determining the minimum number of line coordinates to be locked on the two reference images 0 and $m$ will be the subject of further investigation.

As far as frame $m$ is concerned, it should be chosen sufficiently far from the reference frame, in order to make the triangulation more robust.

The algorithm that we propose here uses the first few seconds of a video sequence to recover a rough estimate of the camera motion: a triplet with a sufficiently wide baseline is extracted and a classical algorithm based on trifocal tensor estimation [11] is run. The third image of the triplet is chosen as frame $m$, and the computed motion is used for initialization purposes.

The expression of a line $L$ defined as the intersection of two generic planes $\Pi_1$ and $\Pi_2$ is given in the appendix (see Eq. (39)). Here the planes $\Pi_1 = \Phi_L^{(0)}$ and $\Pi_2 = \Phi_L^{(0)}$ are

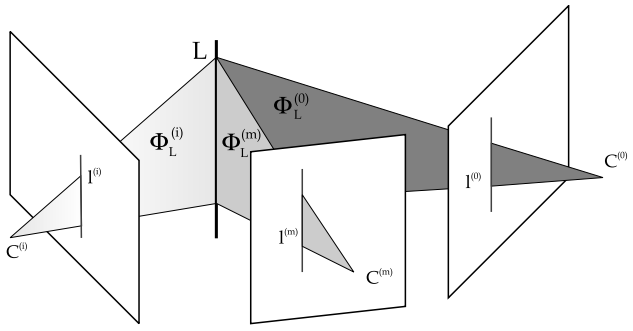


Fig. 6. A 3D line in the scene can be expressed as the intersection of two back-projection planes (from the cameras 0 and $m$).

written as functions of the image-plane coordinates $\bar{\mathbf{l}}^{(0)}$ and $\bar{\mathbf{l}}^{(m)}$ as follows:

$$\bar{\mathbf{n}}_{\Pi_1} = \bar{\mathbf{l}}^{(0)}, \quad \bar{\mathbf{n}}_{\Pi_2} = \mathbf{R}^{(m)}\bar{\mathbf{l}}^{(m)}\tilde{\mathbf{R}}^{(m)},$$

where $\bar{\mathbf{l}}^{(m)}$ is the normal to the back-projection plane in the reference system attached to the $m$th camera, therefore the rotor $\mathbf{R}^{(m)}$ is needed to give the correct orientation of $\Pi_2^*$. Since $d_{\Pi_1} = 0$ and $d_{\Pi_2} = \mathbf{t}^{(m)} \cdot \Pi_2^*$, the vectors $\mathbf{q}_L$, and $\bar{\mathbf{n}}_L$ defining the line $L$ can be written as

$$\mathbf{q}_L = \frac{(\mathbf{R}^{(m)}\bar{I}^{(m)}\tilde{\mathbf{R}}^{(m)}) \cdot t^{(m)}}{-\|(\mathbf{R}^{(m)}\bar{I}^{(m)}\tilde{\mathbf{R}}^{(m)}) \wedge \bar{I}^{(0)}\|}\bar{I}^{(0)}; \bar{\mathbf{n}}_L = \frac{[(\mathbf{R}^{(m)}\bar{I}^{(m)}\tilde{\mathbf{R}}^{(m)}) \wedge \bar{I}^{(0)}]I_3}{-\|(\mathbf{R}^{(m)}\bar{I}^{(m)}\tilde{\mathbf{R}}^{(m)}) \wedge \bar{I}^{(0)}\|}. \tag{33}$$

### 6.5. The coplanarity constraint

Coplanar primitives are quite frequent in video sequences. Typical examples are the inscriptions on a box, or the details on the facade of a building or on a painting. In this Section we show how it is possible to take advantage of this additional constraint on the scene in an EKF algorithm.

When dealing with coplanar features we can follow two possible approaches: the first consists of forcing the coplanarity constraint at each step in order to refine the structure estimation; another solution consists of describing the structure with a minimum number of parameters that automatically satisfy the constraint. Quite clearly, the second choice is more suitable and effective for an EKF structure. The solution that we propose here belongs to this category and is characterized by the fact that the algorithm parameterizes the coplanar primitives through the plane they belong to. In other words, what we include in the filter state are not the parameters of the coplanar primitives but the parameters of the planes of coplanarity.

Eqs. (24) and (25) can be used in Eqs. (17) and (19) when predicting the new image-plane coordinates, i.e. when the structure estimate at step $i$ is computed. However coplanar points and lines are parameterized in the state vector through the parameters of the plane where they lie and the image-plane coordinates of their projections onto the reference camera. Thus the knowledge of the complete state-measures relationship is needed in order to compute the Jacobian matrix used in the Filter.

The generic expression of the intersections between two planes or a plane and a line are reported in the Appendix A (Eqs. (39) and (42)). Let $\Pi$ be a plane in the scene, with normal $\bar{\mathbf{n}}_\Pi$ and distance from the origin $d_\Pi$. A point P lying on the plane $\Pi$ can be seen as the intersection of its optical ray $\Psi_{\mathbf{P}}^{(0)}$ from the first camera with $\Pi$(see Fig. 7). If we perform the substitutions $\bar{\mathbf{n}}_L = \bar{\mathbf{p}}^{(0)}$ and $\mathbf{q}_L = 0$ in Eq. (42), which are derived from the fact that the optical ray passes through the origin and its direction is the normalized vector pointing to the image-plane coordinates, then the offset of the point **P** is

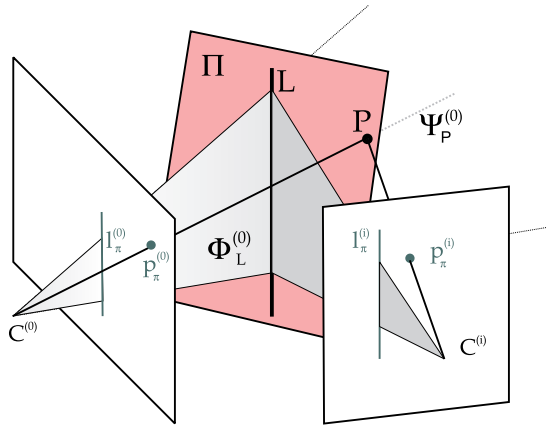


Fig. 7. A point P lying on a plane $\Pi$ can be expressed as the intersection of such plane with the back-projection ray $\Psi_L^{(0)}$ from the first camera. A 3D line **L** lying on $\Pi$ is the intersection between $\Pi$ and the back-projection plane $\Phi_L^{(0)}$.

$$\mathbf{o}_\mathrm{P} = \frac{d_\Pi \bar{\mathbf{p}}^{(0)}}{(\bar{\mathbf{n}}_\Pi \cdot \bar{\mathbf{p}}^{(0)})} = \frac{d_\Pi \bar{\mathbf{p}}^{(0)}}{(\bar{\mathbf{n}}_\Pi \cdot \bar{\mathbf{p}}^{(0)})}. \tag{34}$$

A 3D line L lying on II is the intersection between II and the back-projection plane $\Phi_L^{(0)}$ (see Fig. 7). By setting $\Pi_1 = \Pi$ and $\Pi_2 = \Phi_L^{(0)}$, we have $\bar{\mathbf{n}}_{\Pi 2} = \bar{\mathbf{l}}^{(0)}$ and $\mathrm{d}_{\Pi 2} = 0$. From Eq. (39) the vectors that define the line are

$$\mathbf{q}_L = \frac{d_\Pi}{||\bar{\mathbf{n}}_\Pi \wedge \bar{\mathbf{l}}^{(0)}||}\bar{\mathbf{l}}^{(0)}; \quad \bar{\mathbf{n}}_L = \frac{[\bar{\mathbf{n}}_\Pi \wedge (\bar{\mathbf{l}})^{(0)}]\bar{\mathbf{l}}_3}{||\bar{\mathbf{n}} \wedge \bar{\mathbf{l}}^{(0)}||}. \tag{35}$$

By comparing Eqs. (30) and (34), we have $d_p = \frac{d_\Pi}{(\bar{\mathbf{n}}_\Pi \cdot \bar{\mathbf{p}}^{(0)})}$, while by comparing Eqs. (31) with (35) we have $d_L = \frac{d_\Pi}{||\bar{\mathbf{n}}_\Pi \wedge \bar{\mathbf{l}}^{(0)}||}$.

Notice that a plane is described by four parameters in the Conformal space (see Eq. (10)), but number of d.o.f. is indeed only three, as the scale factor is redundant. It is possible to fix the scale factor by setting $||\bar{n}_\Pi|| = 1$, but in order to reduce the number of parameters and get rid of this nonlinear constraint we propose to estimate directly the following 3D vector

$$\mathbf{k}_\Pi = \frac{\bar{\mathbf{n}}_\Pi}{d_\Pi}; \tag{36}$$

this choice also simplifies the expressions of the primitives lying on $\Pi$ as follows

$$\mathbf{o}_\mathrm{P} = \frac{\mathbf{p}^{(0)}}{(\mathbf{k}_\Pi \cdot \mathbf{p}^{(0)})}; \tag{37}$$

$$\mathbf{q}_L = \frac{\bar{\mathbf{l}}^{(0)}}{||\mathbf{k}_\Pi \wedge \bar{\mathbf{l}}^{(0)}||}; \quad \bar{\mathbf{n}}_L = \frac{[\mathbf{k}_\Pi \wedge \bar{\mathbf{l}}^{(0)}]\bar{\mathbf{l}}_3}{||\mathbf{k}_\Pi \wedge \bar{\mathbf{l}}^{(0)}||}. \tag{38}$$

Notice that the proposed parameterization has an intrinsic limit: when the plane $\Pi$ where the primitives lie tends to pass through the origin, both the numerator and the denominator in Eqs. (34) and (35) tend to zero. In fact, while it is obvious that $\mathrm{d}_\Pi \to 0$, the fact that $\bar{\mathbf{n}}_\Pi \cdot \bar{\mathbf{p}}^{(0)} \to 0$ derives from the fact that the back-projection ray tends to lie on $\Pi$. Furthermore $\bar{\mathbf{n}}_\Pi \wedge \bar{\mathbf{l}}^{(0)} \to 0$ because the normals

to the planes $\Pi$ and $\Pi_2 = \Phi_L^{(0)}$ tend to be aligned. As a consequence, computing the 3D position of the primitives as the intersection of such planes constitutes an ill-conditioned approach. Anyway, this situation is not of interest in real applications as lines or points lying on a plane that pass through the center of the camera are not likely to be detected by a feature extraction algorithm.

### 6.6. The scale factor issue

If structure and motion are completely unknown, it is well known that structure parameters and translation vector are only recoverable up to a scale factor.[5] This implies that the solutions to the state estimation task are infinite, therefore the model is unobservable. In order to make it observable, the scale factor must be fixed in some way.

What is commonly done [14–16] is to fix to an arbitrary value a structure parameter, for instance the depth of a point or the distance of a line from the origin. In order to do so, the Kalman Filter is *saturated*, i.e. the model and initial variance of the parameter to be fixed are set to zero in order to keep its estimate fixed to the initial value. All the other parameters automatically scale themselves to accommodate this constraint.

A draw-back of this solution is its dependency from the lifespan of the reference feature. When the feature is no longer trackable, another primitive must be used as reference, and the choice is usually on the feature that exhibits the lowest estimation variance. Feature swapping causes a drift in the estimations [16–18], and particularly in the scale factor. In order to limit this problem, we associate the scale factor to a plane of the scene (if available) by locking one of its three parameters. This means that, as long as enough primitives belonging to this plane are visible throughout the sequence (not necessarily the same ones) the scale factor will keep being associated to the plane, thus minimizing the drift.

## 7. Simulations results

We tested our algorithm on both synthetic and real sequences, but since the quality of the estimates can only be assessed using ground-truth data, this Section will mainly consider experiments on synthetic data. In order to provide a more realistic view of the algorithm performances, we will not just consider completely synthetic feature sets (artificially generated image coordinates corrupted by localization noise (Section 7.2)) but also synthetically generated sequences (Section 7.3). In this second case we render a 3D scene viewed by a moving virtual camera and we process the sequence with a feature tracking algorithm. The tracked features are passed to the Kalman Filter and the tracked camera motion is compared with the known trajectory of the virtual camera (ground-truth data).

In Section 7.4 we also present some results relative to tests conducted on a real sequence acquired with a hand-held camera. In the case of rendered and of real video sequences, we tested the performance of the estimator by *augmenting* the scene with additional synthetic objects and checking whether such objects are, in fact, solidly attached to the scene in motion. The object location is, in fact, based on the scene estimate, and the camera motion allows us to create a new sequence containing the desired object viewed by a moving camera. The original sequence and the new one can now be merged, giving the illusion that the virtual objects is in the scene. Of course this process gives good results only if the camera motion is estimated with accuracy.

Except for Section 7.2, where the performance of the EKF algorithm are presented both in the point and line cases, we will focus our attention on the EKF line algorithm, which exhibits more novel aspects.

### 7.1. Error measurement

In order to measure the translation error we consider two types of measurements: the angle (in degrees)

$$E_{t_d} = \arccos\left(\frac{\mathbf{t}\cdot\hat{\mathbf{t}}}{\|\mathbf{t}\|\ \|\hat{\mathbf{t}}\|}\right)$$

and the distance (in meters)

$$E_{t_m} = (\mathbf{t}-\hat{\mathbf{t}})^2$$

between the actual translation vector ($\mathbf{t}$) and the estimated one ($\hat{\mathbf{t}}$). We derive both values for all the images of the sequence (we only discard the first 50 frames in order to skip the transient) and then we compute their average and standard deviation.

Error in meters are computed after aligning the reconstruction with the ground truth by rescaling the depth of a reconstructed point (or the distance of a line from the origin) to the correct value.

In order to have a scalar measurement of also the rotation error, we plot

$$E_R = \arccos\left(\frac{\mathrm{tr}(\underline{\mathbf{R}}\hat{\underline{\mathbf{R}}}^{\mathrm{T}})-1}{2}\right)$$

in degrees, where $\hat{\underline{\mathbf{R}}}$ is the estimated rotation matrix (computed from the estimated rotor $\hat{\mathbf{R}}$). We evaluate the rotation error for all the images of the sequence (skipping again the estimates related to the first 50 frames) and then we compute its average and standard deviation.

The error on the line structure is given by the average angle (in degrees) between the actual line orientation ($\mathbf{n}_{L_k}$) and the estimated one ($\hat{\mathbf{n}}_{L_k}$)

$$E_L = \frac{1}{K}\sum_{k=1}^{K}\arccos(\mathbf{n}_{L_k}\cdot\hat{\mathbf{n}}_{L_k}),$$

[5] Conversely, the camera rotation and the focal length (if unknown) can be recovered with no ambiguities.

while the error on the point structure is given by the average difference (in meters) between the actual position ($P_h$) and the estimated position ($\hat{P}_h$) of the points

$$E_P = \frac{1}{H}\sum_{h=1}^{H}(P_h - \hat{P}_h)^2.$$

As for the motion, we computed the standard deviation of the error also for the structure.

All the plotted error values have been averaged over 100 trials for different localization noise values.

### 7.2. Simulations on synthetic data

#### 7.2.1. Setup of the experiments

The scene viewed by the camera in the synthetic experiments is shown in Fig. 8. This structure is made of 21 points and 19 lines that join pairs of points, all lying on three planes. The depth of the points is from 200 to 400 times the focal length.

We used this structure to test point-based, line-based and joint point-line based algorithms, running the EKF both with and without information on feature coplanarity. We thus had five different cases:

1. Structure of independent points;
2. Point structure plus coplanarity information – the situation is similar to the independent point case, but the algorithm knows which points belong to which plane, therefore it directly estimates the parameters of the structure planes;
3. Structure of independent lines;
4. Line structure plus coplanarity information – similar to the independent line case, but the algorithm knows which lines belong to which plane, therefore it directly estimates the parameters of the structure planes;
5. Point and line structure plus coplanarity information.

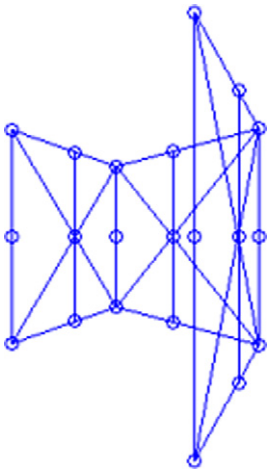

Fig. 8. A view of the synthetic structure used in the experiments, as seen from the camera (frame 60 of a 200 frame sequence). The scene has 21 points and 19 lines, lying on three planes. The depth of the points is from 200 to 400 focal lengths.

In all the experiments the camera follows an orbital motion around an axis that is parallel to the camera's $Y$ axis and passes approximative through the center of the scene (walk-around motion). The center of the scene stays approximatively at a distance of 3 m from the camera with a focal distance of 1 cm. The sequence is 200 frames long and the camera covers an angle of about 120° around the object. The image is $500 \times 500$ pixel, and the side of the image plane is set to 1 unit of focal length (the field of view is 53°).

The noise added to the image-plane coordinates of the points is gaussian with zero mean and standard deviation varying from 1 to 3 pixels, while the line measurements are perturbed by an additive noise on their angle. A gaussian noise with zero mean and standard deviation varying between 0.5° and 1.5° is added by rotating the 2D lines around the midpoint of the viewed segment.

As far as the experiments on mixed features are concerned, we chose sets of coplanar points and lines that correspond to comparable localization/orientation errors. For example, points that are affected by a localization error of 2 pixel and lines that are affected by an orientation error of half a degree are used together because half a degree of orientation error on the average viewed segment shifts its endpoints of about 2 pixels. Similarly, we paired points affected by 3 pixels of localization errors with lines affected with 1° of orientation error. We will refer to these two levels of noise as *level 1* and *level 2*.

#### 7.2.2. Evaluation of the algorithm

In Figs. 9–16 we report the estimation errors of the EKF algorithm in the four cases discussed above (independent/coplanar points and lines). While in the independent point case the algorithm does not need any a-priori information on the structure (usually the initial estimates of the points are set on a plane parallel to the image plane and placed at an arbitrary distance in front of the camera), a rough initialization is required both in the line case and in the coplanar features case. More specifically, in order to converge to a correct estimate, the algorithm needs to know the

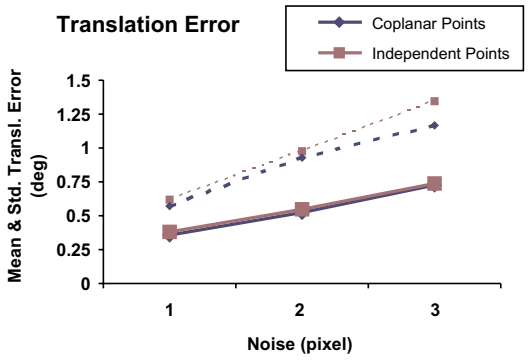


Fig. 9. Translational estimation error in degrees (point structure experiment) with a varying noise level. Solid lines, mean value; Dotted lines, mean value + 2 * std. dev.; Black lines, the filter uses coplanarity information; Gray lines, the lines are considered as independent.

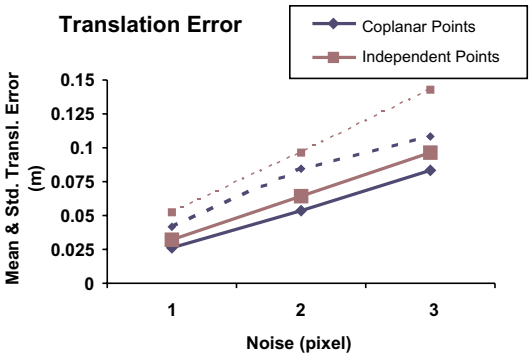


Fig. 10. Translational estimation error in meters (point structure experiment) with a varying noise level. Solid lines, mean value; Dotted lines, mean value + 2 * std. dev.; Black lines, the filter uses coplanarity information; Gray lines, the lines are considered as independent.

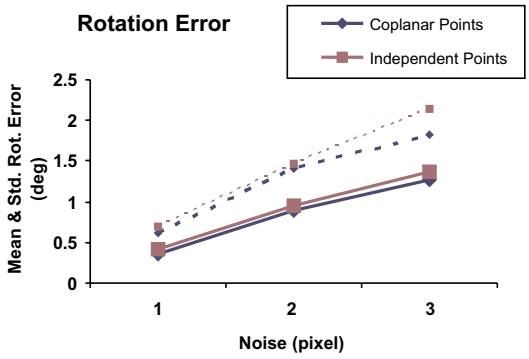


Fig. 11. Rotational estimation error in degrees (point structure experiment) with a varying noise level. Solid lines, mean value; Dotted lines, mean value + 2*std. dev.; Black lines, the filter uses coplanarity information; Gray lines, the lines are considered as independent.

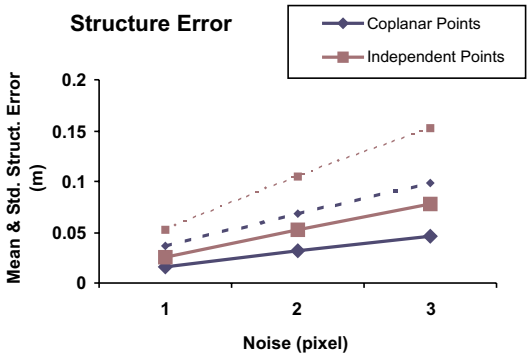


Fig. 12. Structure's estimation error in meters (point structure experiment) with a varying noise level. Solid lines, mean value; Dotted lines, mean value + 2 * std. dev.; Black lines, the filter uses coplanarity information; Gray lines, the lines are considered as independent.

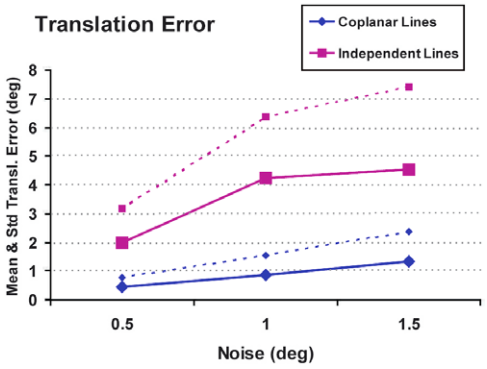


Fig. 13. Translational estimation error in degrees (line structure experiment) with a varying noise level. Solid lines, mean value; Dotted lines, mean value + 2 * std. dev.; Black lines, the filter uses coplanarity information; Gray lines, the lines are considered as independent.

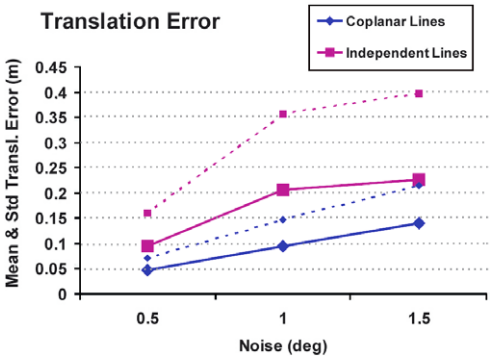


Fig. 14. Translational estimation error in meters (line structure experiment) with a varying noise level. Solid lines, mean value; Dotted lines, mean value + 2 * std. dev.; Black lines, the filter uses coplanarity information; Gray lines, the lines are considered as independent.

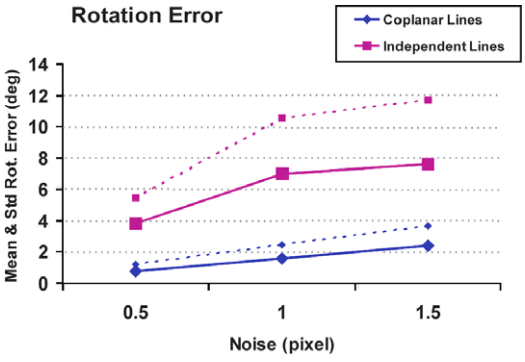


Fig. 15. Rotational estimation error in degrees (line structure experiment) with a varying noise level. Solid lines, mean value; Dotted lines, mean value + 2 * std. dev.; Black lines, the filter uses coplanarity information; Gray lines, the lines are considered as independent.

approximate orientation of lines and planes in the 3D space. This initialization is provided by a 2-view or 3-view estimation in the point or line case, respectively.

As far as concerns the coplanarity constraint, it should be exploited by the algorithm whenever this situation is encountered. Coplanarity information could be known in advance, but it is also possible to devise a run-time check on feature coplanarity which clusterizes coplanar features. This would allow us to switch the parameterization of features from 'independent' to 'coplanar' and add the common plane to the state vector. We ought to be careful, however, that this approach is expected to be fruitful only

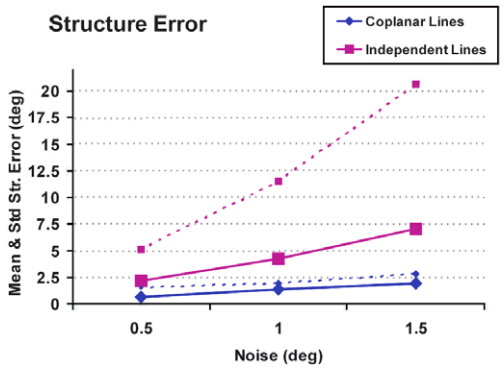


Fig. 16. Structure's estimation error (orientation of the lines) in degrees, with a varying noise level. Solid lines, mean value; Dotted lines, mean value + 2 * std. dev.; Black line, the filter uses coplanarity information; Gray lines, the lines are considered as independent.

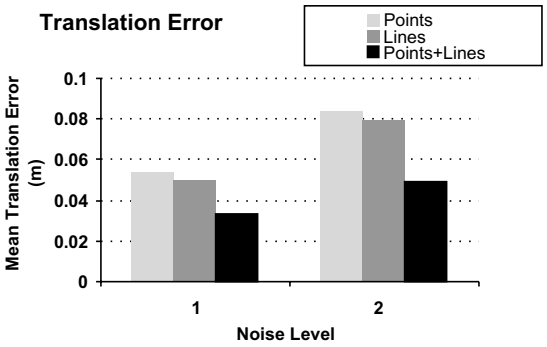


Fig. 17. Translational estimation error in meters for a structure of 19 lines and 21 points, with a varying noise level on feature position (level 1: std. dev. of 2 pixel (points) and 0.5° (lines); level 2: std. dev. of 3 pixel (points) and 1° (lines)).

if we can trust co-planarity judgement. If the coplanarity decision is incorrectly taken, this could impact negatively on the final result. In our experiments, the coplanarity constraint was imposed starting from a-priori information, but the problem of automatically clustering coplanar features is indeed worth investigating further in the future.

We would like to stress the fact that, in order to correctly steer the evolution of the Kalman Filter in the parameter space, the algorithm only requires a rough initialization: the structure estimate is refined during the global estimation process. This means that only a small motion around the object is required in order to recover a rough estimate of the structure: only one or two seconds of video sequence before starting the EKF algorithm is usually sufficient.

As expected, feature coplanarity is confirmed to improve the performance of the algorithm, as already mentioned in previous works [32]. This is particularly true in the line case.

We also performed some tests with mixed data (points and lines together). In these experiments, the three planes of the structures shown in Fig. 8 are estimated using observations of point and lines. The algorithm is provided with coplanarity information. We run two experiments. In the first one all the points (21) and lines (19) are available at the same time to the algorithm in the presence of two levels (1 and 2) of localization error. In Figs. 17, 19 and 21 results are shown in comparison with results obtained from only points (std. dev. of 2 and 3 pixels) or lines (std. dev. of 0.5° and 1°.). By using points and lines together, the quality of the estimates turns out to be greatly improved. This, however, could be easily attributed to the fact that the number of observed features is greater than before. For this reason, in the second experiment we wanted to compare the algorithm's performance in the presence of mixed structures while keeping the total number of observed features fixed. We run simulations with 21 points (case a), 15 points and 5 lines (case b), 10 points and 10 lines (case c), 5 points and 15 lines (case d) and 19 lines (case e), all picked from the structure of Fig. 8. Such simulations were run for both

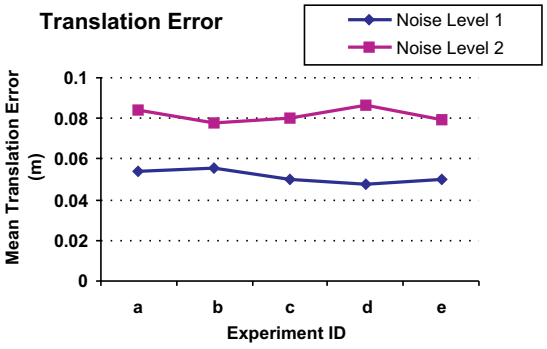


Fig. 18. Translational estimation error in meters, with five different structures, at two different localization error level. Structures are made of 21 points (a), 15 points and 5 lines (b), 10 points and 10 lines (c), 5 points and 15 lines (d) and 19 lines (e).

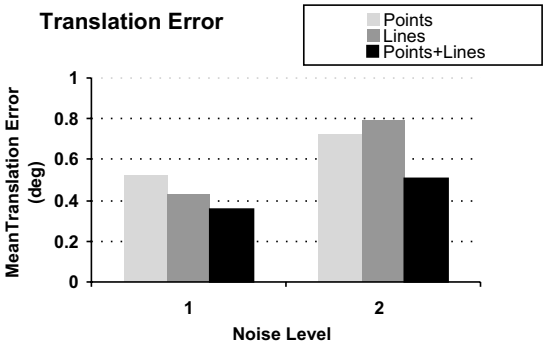


Fig. 19. Translational estimation error in degrees for a structure of 19 lines and 21 points, with a varying noise level on feature position (level 1: std. dev. of 2 pixel (points) and 0.5° (lines); level 2: std. dev. of 3 pixel (points) and 1° (lines)).

levels of localization error. The relative results are shown in Figs. 18, 20 and 22. As we can see, the algorithm exhibits a nearly constant performance for the various mixes of points and lines. Performance tends to slightly worsen for lines with increasing localization error. In conclusion, the information provided by point-like observations turns

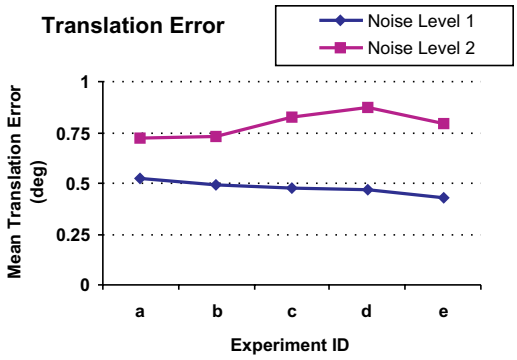


Fig. 20. Translational estimation error in degrees, with five different structures, at two different localization error level. Structures are made of 21 points (a), 15 points and 5 lines (b), 10 points and 10 lines (c), 5 points and 15 lines (d) and 19 lines (e).

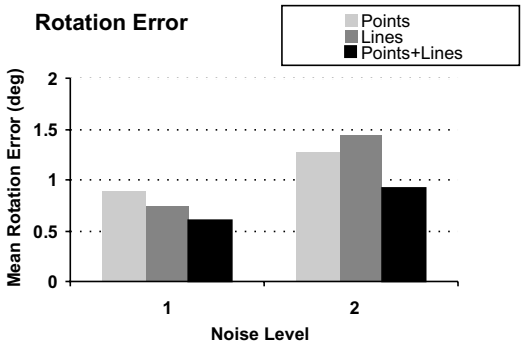


Fig. 21. Translational estimation error in degrees for a structure of 19 lines and 21 points, with a varying noise level on feature position (level 1: std. dev. of 2 pixel (points) and 0.5° (lines); level 2: std. dev. of 3 pixel (points) and 1° (lines)).

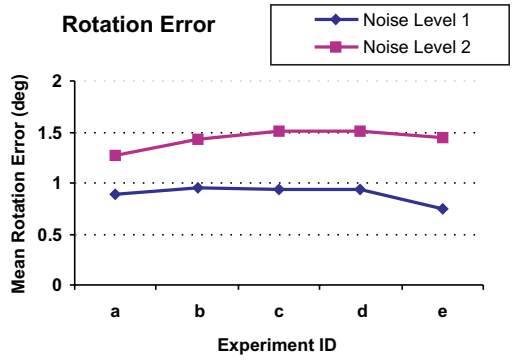


Fig. 22. Rotational estimation error in meters, with five different structures, at two different localization error level. Structures are made of 21 points (a), 15 points and 5 lines (b), 10 points and 10 lines (c), 5 points and 15 lines (d) and 19 lines (e).

out to be approximatively equivalent to that of line features for camera motion estimation purposes.

### 7.3. An experiment on a rendered sequence

In this experiment the EKF line algorithm was fed with trajectories extracted by a tracking algorithm. The sequence is synthetic and was generated with 3DSMax®. The virtual scene contains a textured cube, shown in Fig. 23.

Fig. 23. An image from the sequence generated with 3DSMax®. The lines on the cube sides are tracked by a line tracker and the EKF algorithm uses their coordinates as measurements.

The image-plane coordinates of the lines are tracked by a simple algorithm based on an edge extraction followed by a Hough transform. A global Hough transform is computed on the edge map of the first frame of the sequence and whenever new lines need to be extracted. Then local maxima of the Hough transform are extracted, and the visible segments of the corresponding lines are localized on the image. In the following frames a local Hough transform is computed for each segment in a bounding box centered on its position in the previous frame.

#### 7.3.1. Setup of the experiment

The side of the cube is 1 meter long. The animated camera moves around the cube keeping approximatively at a

Fig. 24. In this picture both the real and the estimated trajectories of the camera are visualized. The black curve is the real trajectory, while the gray curve is the trajectory estimated by the EKF algorithm using the coplanarity information for the lines on the cube sides.

Table 1
Estimation errors in the synthetic cube experiment

| | **TRANSL** (deg) $E_{t_d}$ | | **TRANSL** (m) $E_{t_m}$ | | **ROT** (deg) $E_R$ | |
|---|---|---|---|---|---|---|
| | Mean | Std. dev. | Mean | Std. dev. | Mean | Std. dev. |
| Copl. lines | 1.51 | 0.74 | 0.026 | 0.011 | 0.55 | 0.18 |
| Indep. lines | 2.63 | 0.81 | 0.065 | 0.015 | 1.07 | 0.64 |

distance of 3–3.5 m from the cube center and following the trajectory shown in Fig. 24. The sequence is 200 frames long, and the camera covers an angle of approximatively 28° around the cube. The field of view of the camera is set to 45°, and the size of the image plane is $320 \times 240$ pixel.

The rough quantization introduced both by the Hough transform (angle sampling step is 1°) and the low resolution of the image cause the trajectories of the tracked line to be very noisy, therefore we expect that the performance of the algorithm will improve significantly when coplanarity information on the lines is provided.

Table 1 shows the motion estimation errors for lines features with and without coplanarity information.

As we can see, the estimation errors are given for both direction (degrees) and magnitude (meters) of the translation vector. In Fig. 24 we can also see both the estimated and the real trajectories of the camera in the case of coplanar lines. A comparison between the two trajectories immediately gives us an idea of the quality of the estimates.

In order to visually assess the quality of our camera tracker we rendered a synthetic turtle on the cube sequence. As we can see from the frames shown in Fig. 25, the object moves solidly and consistently with the scene.

### 7.4. An experiment on a real sequence

With no ground-truth data available on the camera motion, the only way we have to quantify the estimation error is to compare the estimated 3D structure (if known) with the estimated one. The scene that we considered is a miniature ($40 \times 40 \times 20$ cm) portion of a crossing between a railroad and a road (see Fig. 26). The two ramps at the sides of the railroad are coplanar, therefore the structure can be considered as made of three planes. We measured the angles between any pair of them, in order to compare the true angles with the estimated ones.

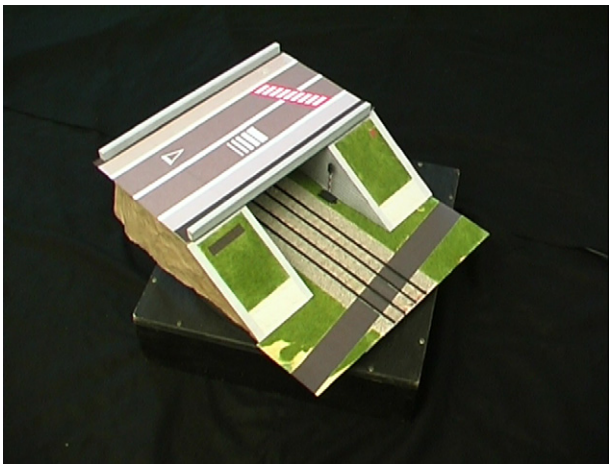

Fig. 26. One frame from the miniature crossrail intersection scene.

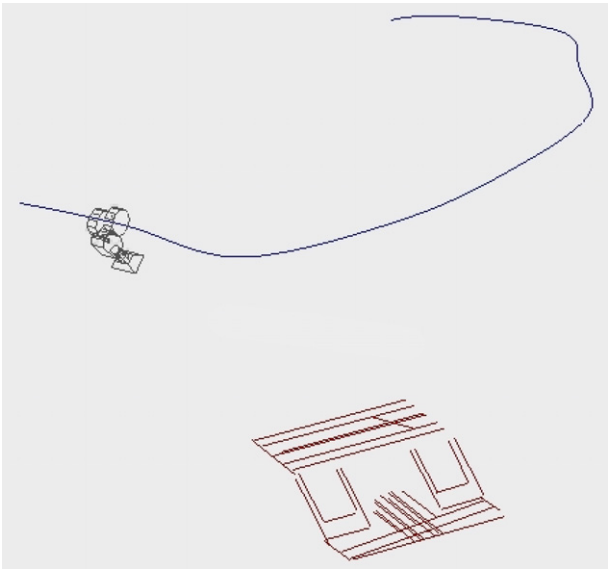

Fig. 27. Structure and camera trajectory of the crossrail intersection scene, estimated by the EKF-based algorithm.

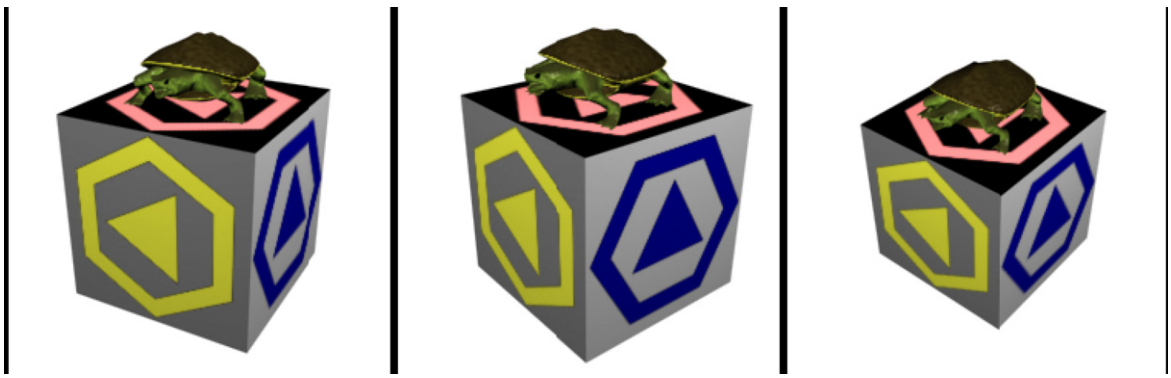

Fig. 25. Three frames of the augmented sequence where the original video of the synthetic cube is merged with the new sequence of the synthetic turtle. The turtle is rendered from a viewpoint that coincides with the motion estimated by the EKF algorithm.

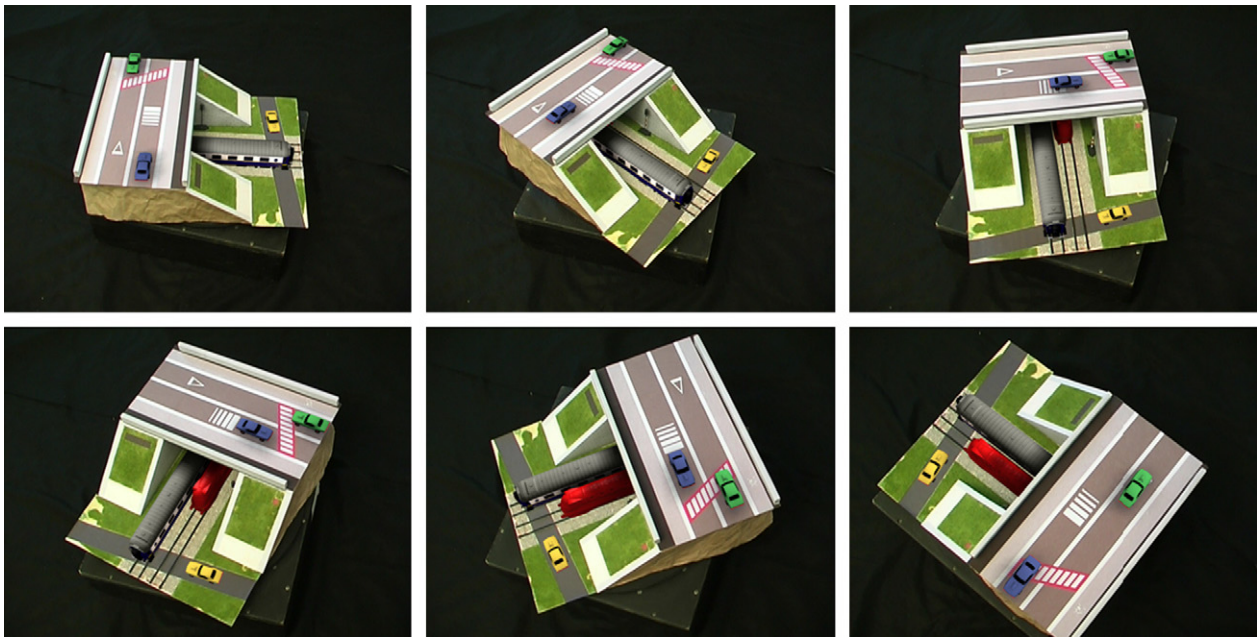

Fig. 28. Six frames from the augmented crossrail intersection sequence, where synthetic moving and static vehicles (train engine and carriage as well as cars) have been rendered and added to the scene. From a visual inspection of the resulting video sequence, it is clear how all vehicles are always correctly positioned within the scene.

In the estimation task we only use lines and coplanarity information. Feature extraction and tracking are performed by the simple algorithm described in the previous Section. The sequence is 400 frames long, and the number of tracked lines is 30.

The estimated trajectory and structure are shown in Fig. 27.

The structure is estimated with good precision: the error on the angles between any couple of planes is always below 0.5°.

In order to give a visual idea of the quality of the estimates, we rendered additional virtual objects (some static, some others moving along the road or on the railroad) and overlaid them on the original sequence (see Fig. 28).

## 8. Conclusions

In this article we discussed the problem of estimating 3D structure and camera motion from 2D observations of points, lines and coplanar features. By casting the problem into the framework of Geometric Algebra we could benefit from a unified formalism for dealing with structures of point, lines, planes and coplanarity information.

By adopting the Conformal Model of GA we could use a very compact notation and we could easily manipulate equations describing rigid motions. The EKF-based algorithm presented here is an example of application of this approach. This algorithm is a causal scheme that performs a 3D estimation from the observations (sequential or simultaneous) of point and line features, and its performance is thoroughly assessed and discussed in this article. The quality of the estimates is, in general, very good in the presence of a reasonable amount of noise, and the impact of coplanarity information is usually quite substantial, especially when the noise localization level is significant.

The independent (non-coplanar) lines case is the most critical one, as the estimation error is larger than in the other cases. Expanding the state vector with the 2D line coordinates will probably increase the quality of the estimation, since a localization error on the two reference images used for triangulation can deeply affect the recovered 3D structure and therefore the estimates of the camera motion at the frame $m$. For this reason, further investigations are needed on the minimum number of line coordinates to be locked on the reference images 0 and $m$ in order to make the model observable.

## Appendix A

In this Appendix we provide the expression of the intersections between two planes and a line and a plane as a function of their Euclidean parameters. Such expressions can be also computed using Maple® or Mathematica® packages of GA symbolic computation, available for free downloading in [50]. Two planes $\Pi_1$ and $\Pi_2$ always intersect in a line, unless they are parallel:

$$\mathbf{L} = \Pi_1 \vee \Pi_2 = -(\Pi_1^* \wedge \Pi_2^*)^*.$$

Let $\Pi_i^* = \bar{\mathbf{n}}_{\Pi_i}, +d_{\Pi_i}\mathbf{n}$ (Eq. (10)); we have

$$\begin{aligned}\mathbf{L}^* &= -(\bar{\mathbf{n}}_{\Pi_1} + d_{\Pi_1}\mathbf{n}) \wedge (\bar{\mathbf{n}}_{\Pi_2} + d_{\Pi_2}\mathbf{n}) \\ &= -(\bar{\mathbf{n}}_{\Pi_1} \wedge \bar{\mathbf{n}}_{\Pi_2}) - [(d_{\Pi_1}\mathbf{n} \wedge \bar{\mathbf{n}}_{\Pi_2}) + (\bar{\mathbf{n}}_{\Pi_2} \wedge d_{\Pi_2}\mathbf{n})] \\ &= -(\bar{\mathbf{n}}_{\Pi_1} \wedge \bar{\mathbf{n}}_{\Pi_2}) + [d_{\Pi_2}\bar{\mathbf{n}}_{\Pi_1} - d_{\Pi_1}\bar{\mathbf{n}}_{\Pi_2}]\mathbf{n}. \end{aligned} \quad (39)$$

By comparing Eqs. (8) and (39), we see that the intersection of two planes $\Pi_1$ and $\Pi_2$ is a line that is perpendicular to the plane spanned by the normals to the two planes $\Pi_1$ and $\Pi_2$ ($\mathbf{n}_L = -(\bar{\mathbf{n}}_{\Pi_1} \wedge \bar{\mathbf{n}}_{\Pi_2})^*$). The vector $q_L$ should lie on the plane that is perpendicular to the line and, in fact, it is a linear combination of $\bar{\mathbf{n}}_{\Pi_1}$ and $\bar{\mathbf{n}}_{\Pi_2}$ ($\mathbf{q}_L = [d_{\Pi_2}\bar{\mathbf{n}}_{\Pi_1} - d_{\Pi_2}\bar{\mathbf{n}}_{\Pi_2}]$).

A line **L** always intersects a plane II in one point, unless they are parallel. If the plane and the line intersect in the point X, the resulting bivector can be proven to be always of the form

$$\mathbf{B} = \Pi \vee \mathbf{L} = \mathbf{X} \wedge \mathbf{n}. \tag{40}$$

This fact is quite intuitive as the points at infinity in the Euclidean space are mapped onto the point **n** in the Conformal space. As a consequence, all the geometric objects passing through the infinity (lines and planes) intersect at least in **n**.

Eq. (40) can be easily proven:

$$\begin{aligned}\mathbf{B}^* = (\Pi \vee \mathbf{L})^* &= -(\Pi^* \wedge \mathbf{L}^*) = -(\bar{\mathbf{n}}_\Pi + d_\Pi \mathbf{n}) \wedge (\bar{\mathbf{n}}_L \mathbf{I}_3 + \mathbf{q}_L \mathbf{n}) \\ &= -(\bar{\mathbf{n}}_\Pi \wedge \mathbf{n}_L \mathbf{I}_3) - (d_\Pi \mathbf{n} \wedge \bar{\mathbf{n}}_L \mathbf{I}_3) - (\bar{\mathbf{n}}_\Pi \wedge \mathbf{q}_L \mathbf{n}),\end{aligned}$$

since $\mathbf{q}_L \mathbf{n} = \mathbf{q}_L \wedge \mathbf{n}$, and therefore $d_\Pi \mathbf{n} \wedge (\mathbf{q}_L \wedge \mathbf{n}) = 0$. We can rearrange the expression above as follows

$$\mathbf{B}^* = (\Pi \vee \mathbf{L})^* = -(\bar{\mathbf{n}}_\Pi \cdot \bar{\mathbf{n}}_L)\mathbf{I}_3 - [d_\Pi \bar{\mathbf{n}}_L \mathbf{I}_3 + (\bar{\mathbf{n}}_\Pi \wedge \mathbf{q}_L)]\mathbf{n},$$

therefore

$$\mathbf{B} = (\Pi \vee \mathbf{L}) = -(\bar{\mathbf{n}}_\Pi \cdot \bar{\mathbf{n}}_L)\mathbf{E} + [d_\Pi \bar{\mathbf{n}}_L - (\bar{\mathbf{n}}_\Pi \wedge \mathbf{q}_L)\mathbf{I}_3]\mathbf{n}. \tag{41}$$

Given a 3D point x, it is simple to verify that

$$\mathbf{X} \wedge \mathbf{n} = \mathbf{x}\mathbf{n} - \mathbf{E},$$

which is the same expression as Eq. (41) up to the scale factor $(\bar{\mathbf{n}}_\Pi \cdot \bar{\mathbf{n}}_L)$.

The Euclidean expression of the intersection point is therefore

$$\mathbf{x} = \frac{[d_\Pi \bar{\mathbf{n}}_L - (\bar{\mathbf{n}}_\Pi \wedge \mathbf{q}_L)\mathbf{I}_3]}{(\bar{\mathbf{n}}_\Pi \cdot \bar{\mathbf{n}}_L)}. \tag{42}$$

## References


[1] H. Longuet-Higgins, A computer algorithm for reconstructing a scene from two projections, Nature 293 (1981) 133–135.

[2] Q.T. Luong, T. Vieville, Canonical representations for the geometries of multiple projective views, Computer Vision and Understanding 64 (2) (1996) 193–229.

[3] B. Triggs, Autocalibration and the absolute quadric, Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (1997) 609–614.

[4] M. Pollefeys, R. Koch, L. Van Gool, Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In Proceedings of International Conference on Computer Vision, Bombay, 1998, pp. 90–96.

[5] O.D. Faugeras, Q.T. Luong, S.J. Maybank, Camera self-calibration: theory and experiments, ECCV'92, LNCS 588, Springer Verlag, Berlin, 1992, pp. 321–334.

[6] A. Heyden, K. Astrom, Euclidean reconstruction from image sequences with varying and unknown focal length and principal point, Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (1997) 438–443.

[7] R. Hartley, R. Gupta, T. Chang, Stereo from uncalibrated cameras, in: Computer Vision and Pattern Recognition, 1992, pp. 761–764.

[8] P. Beardsley, P. Torr, A. Zisserman, 3D model acquisition from extended image sequences, ECCV (1996) 683–695.

[9] A.W. Fitzgibbon, A. Zisserman, Automatic camera recovery for closed or open image sequences, ECCV (1998) 311–326.

[10] B. Triggs, P.F. McLauchlan, R.I. Hartley, A. Fitzgibbon, Bundle adjustment – a modern synthesis, vision algorithms: theory and practice, in: Lecture Notes in Computer Science, vol. 1883, Springer-Verlag, 2000, pp. 298–372.

[11] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, Cambridge, MA, 2000.

[12] O. Faugeras, Q. Luong, The Geometry of Multiple Images, The MIT Press, Cambridge, MA, 2001.

[13] Y. Ma, S. Soatto, J. Košecká, S.S. Sastry, An invitation to 3-D vision—from images to geometric models, Springer Verlag, Berlin, 2004.

[14] A. Azarbayejani, A.P. Pentland, Recursive estimation of motion, structure, and focal length, IEEE Transactions on Pattern Analysis and Machine Intelligence 17 (6) (1995) 562–575.

[15] T. Jebara, A. Azarbayejani, A. Pentland, 3D structure from 2D motion, IEEE Signal Processing Magazine 16 (3) (1998) 66–84.

[16] A. Chiuso, P. Favaro, H. Jin, S. Soatto, 3-D motion and structure causally integrated over time: theory (stability) and practice (occlusions), ESSRL Technical Report 99-003, October 1999.

[17] A. Chiuso, P. Favaro, H. Jin, S. Soatto, 3-D motion and structure from 2-D motion causally integrated over time: implementation, Proceedings of the European Conference on Computer Vision, Lecture Notes in Computer Science 1842 (2000) 735–750.

[18] A. Chiuso, P. Favaro, H. Jin, S. Soatto, Structure from motion causally integrated over time, IEEE Transactions on Pattern Analysis and Machince Intelligence 24 (4) (2002) 509–522.

[19] A.J. Davison, Real-time simultaneous localisation and mapping with a single camera, in: Proceedings of the Ninth International Conference on Computer Vision ICCV'03, October 2003, Nice, France, 2003.

[20] A. Gruber, Y. Weiss, Multibody factorization with uncertainty and missing data using the EM algorithm, International Conference on Computer Vision and Patern Recognition (2004) 707–714.

[21] D.D. Morris, T. Kanade, A unified factorization algorithm for points, line segments and planes with uncertainty models, ICCV (1998) 696–702.

[22] F. Dellaert, S. Thrun, C. Thorpe, Jacobian images of super-resolved texture maps for model-based motion estimation and tracking, in: IEEE Workshop on Applications of Computer Vision (WACV'98), October 1998, IEEE Computer Society, Princeton, NJ, 1998, pp. 2–7.

[23] C.J. Taylor, D. Kriegman, Structure and motion from line segments in multiple images, IEEE Transaction on Pattern Analysis and Machine Intelligence 17 (11) (1995) 1021–1033.

[24] Y. Ma, R. Vidal, J. Košecká, S. Sastry, Rank conditions on the multiple-view matrix, International Journal of Computer Vision 59 (2) (2004) 115–137.

[25] A. Bartoli, P. Sturm, The 3D line motion matrix and alignment of line reconstructions, International Journal of Computer Vision 57 (3) (2004) 159–178.

[26] J.L. Jezouin, N. Ayache, Three-dimensional structure from a monocular sequence of images, ICCV (1990) 441–444.

[27] T. Vieville, O. Faugeras, Feed-forward recovery of motion and structure from a sequence of 2D-lines matches, ICCV (1990) 517–520.

[28] P. Smith, I. Reid, A.J. Davison, Real-time monocular SLAM with straight lines, in: British Machine Vision Conference, 2006, pp. 17–26.

[29] R. Szeliski, P. Torr, Geometrically Constrained Structure from Motion: Points on Planes, MSR-ATR-98-64, Microsoft Research, 1998.

[30] O. Faugeras, F. Lustman, Motion and structure from motion in a piecewise planar environment, International Journal of Pattern Recognition and Artificial Intelligence 2 (3) (1988) 485–508.

[31] O. Shakernia, C.S. Sharp, R. Vidal, D. Shim, Y. Ma, S. Sastry, Multiple view motion estimation and control for landing an unmanned aerial vehicle, International Conference on Robotics and Automation (ICRA), 2002.
[32] A. Bartoli, P. Sturm, Constrained structure and motion from multiple uncalibrated views of a piece-wise planar scene, International Journal of Computer Vision 52 (1) (2003) 45–64.
[33] J. Strom, T. Jebara, S. Basu, A. Pentland, Real time tracking and modeling of faces: an EKF-based analysis by synthesis approach, in: Proceedings of the Modelling People Workshop, ICCV'99, August 1999.
[34] J. Strom, Structure from motion of planar surfaces (analysis of an EKF based approach), in: Proceedings of Symposium on Image Analysis, Swedish Society for Automated Image Analysis, March 2001, Norrkoping, Sweden, 2001, pp. 13–16.
[35] J. Alon, S. Sclaroff, Recursive estimation of motion and planar structure, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2, 2000, pp. 550–556.
[36] D. Hestenes, G. Sobczyk, Clifford Algebra to Geometric Calculus—A Unified Language for Mathematics and Physics, Kluwer Academic Publishers, Dordrecht, 1999.
[37] G. Sommer (Ed.), Geometric Computing with Clifford Algebras – Theoretical Foundations and Applications, in: Computer Vision and Robotics, Springer, Berlin, 2001, p. 7.
[38] W.E. Baylis (Ed.), Clifford (Geometric) Algebras with Applications in Physics, Mathematics and Engineering, Birkhauser, Boston, 1996.
[39] E. Baryo-Corrochano, J. Lasenby, G. Sommer, Geometric algebra: a framework for computing point and line correspondences and projective structure using *n* uncalibrated cameras, ICPR-96, August 1996, Vienna, 1996.
[40] Y. Zhang, G. Sommer, E. Bayro-Corrochano, The motor extended Kalman filter for dynamic rigid motion estimation from line observations, in [37], 2001, pp. 501–530.
[41] L. Dorst, S. Mann, Geometric Algebra: A Computational Framework for Geometrical Applications, Course N. 53. Siggraph 2001, Los Angeles, CA.
[42] L. Dorst, Geometric (Clifford) algebra: a practical tool for efficient geometrical representation, University of Amsterdam, Amsterdam, 1999.
[43] A.N. Lasenby, J. Lasenby, R. Wareham, A covariant approach to geometry and its applications in computer graphics, ACM Transactions on Computer Graphics, 2003, submitted for publication.
[44] C. Doran, A.N. Lasenby, J. Lasenby, Conformal geometry, euclidean space and geometric algebra, in: J. Winkler (Ed.), Uncertainty in Geometric Computations, Kluwer, Dordrecht, 2002.
[45] A.N. Lasenby, J. Lasenby, Surface evolution and representation using geometric algebra, in: Proceedings of The Mathematics of Surfaces IX, Proceedings of the Ninth IMA Conference on the Mathematics of Surfaces, London, 2000, pp.144–168.
[46] B. Rosenhahn, G. Sommer, Pose estimation in conformal geometric algebra. Part I: The stratification of mathematical spaces. Part II: Real-time pose estimation using extended feature concepts, Journal of Mathematical Imaging and Vision 22 (1) (2005) 27–70.
[47] S. Haykin, Adaptive Filter Theory, third ed., Prentice Hall, Englewood Cliffs, NJ, 1996.
[48] M.S. Grewal, A.P. Andrews, Kalman Filtering, Theory and Practice, Prentice Hall, Englewood Cliffs, NJ, 1993.
[49] B.C. Hall, Lie Groups, Lie Algebras, and Representations, Springer Verlag, Berlin, 2003.
[50] Available from: http://www.mrao.cam.ac.uk/~clifford/pages/software.htm.