Image Sensor Model Using Geometric Algebra: From Calibration to Motion Estimation

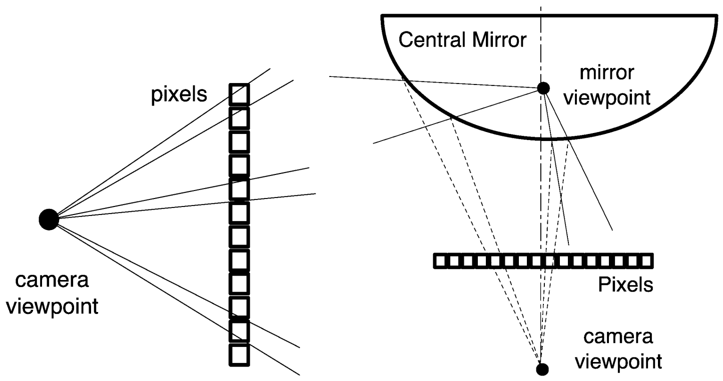
Thibaud Debaecker, Ryad Benosman,
and Sio H. Ieng

Abstract In computer vision image sensors have universally been defined as the nonparametric association of projection rays in the 3D world to pixels in the images. If the pixels' physical topology can be often neglected in the case of perspective cameras, this approximation is no longer valid in the case of variant scale sensors, which are now widely used in robotics. Neglecting the nonnull pixel area and then the pixel volumic field of view implies that geometric reconstruction problems are solved by minimizing a cost function that combines the reprojection errors in the 2D images. This paper provides a complete and realistic cone-pixel camera model that equally fits constant or variant scale resolution together with a protocol to calibrate such a sensor. The proposed model involves a new characterization of pixel correspondences with 3D-cone intersections computed using convex hull and twists in Conformal Geometric Algebra. Simulated experiments show that standard methods and especially Bundle Adjustment are sometimes unable to reach the correct motion, because of their ray-pixel approach and the choice of reprojection error as a cost function which does not particularly fit the physical reality. This problem can be solved using a nonprojective cone intersection cost function as introduced below.

1 Introduction

A large amount of work has been carried out on perspective cameras introducing the pinhole model and the use of projective geometry. This model turns out to be very efficient in most cases, and it is still widely used within the computer vision community. Several computation improvements have been introduced [5]; nevertheless, this model has limitations, notably that it can only be applied to projective sensors. The pixel sampling and slight fluctuations in the calibration process lead to the fact

T. Debaecker (✉)
Institut des Systèmes Intelligents et de Robotique (ISIR), 4 place Jussieu, Paris, France
e-mail: thibaud.debaecker@isir.jussieu.fr



(a) Pixel-cones of a perspective camera. (b) Catadioptric pixel-cones after reflection on a central mirror

Fig. 1 Pixel-cones in the case of perspective cameras and variant scale sensors (here a central catadioptric sensor). All cones are almost the same in (**a**), whereas in (**b**) the pixel-cones vary drastically according to the position of the pixel within the perspective camera observing the mirror

that two rays of view of pixels representing the same point in the scene never exactly intersect across the same 3D point they should represent. Finally, this model fails to introduce the reality of the sensor, since the approximation of the field of view of the pixel is restricted to a ray instead of a small surface in the image plane.

The limitations pointed out become drastically problematic with the appearance of new kinds of nonlinear visual sensors like foveolar retinas [3] or panoramic sensors (see [2] for an overview). As shown in Fig. 1(a), in the case of perspective cameras, all pixels produce a similar cone of view. Cones being barely the same, it is easily understandable why cones can be approximated using lines in this case, but as an illustration of the consequences on a pinhole case, let us consider two identical 640×480 cameras with a 30-centimeter space between each of them, and watching a scene from a five-meter distance. Following the formulation we will describe in the following sections, for a pair of matched points, it is possible to compute in 3D space the real volume which represents the solution set of the triangulation problem. It appears that the mere pixel sampling leads to a 12-cm^3 volume of solutions. It becomes obvious that this approximation using rays will lead to significant imprecision for a catadioptric sensor (combination of a perspective camera and a hyperboloid mirror), especially in the computation of intersections, as shown in Fig. 1(b). Cones then become an absolute necessity.

Different methods have been developed to address the motion estimation problem despite the issue of nonperfect intersection of rays. It is important to notice two particular points in these methods. First, a cost function is chosen to evaluate the accuracy of a solution, and then a seeking strategy is chosen to reach the optimal solution according to this cost function. If the panel of seeking strategies is wide (Branch and Bound Algorithm [11], Levenberg-Marquardt and other least-squared minimization used in the classic Bundle Adjustment (BA) [18], Second Order Cone Programming [10]), the cost function is in most cases the reprojection error. Because of the 5 DOF resolution of the motion estimation problem, this cost function is logically chosen instead of distances of rays in 3D space.

The aim of this paper is to show that introducing cones gives an opportunity to be closer to the real physics of pixel correspondences and thus generates more accurate situations. Euclidean spaces do not allow an easy-manipulating cone expression, especially if intersections of such cones have to be computed. Conformal Geometric Algebra is introduced to enable a simple formulation of cones using twists [17]. A simple line is used as the twist axis to rotate a second line used as the cone directrix. A wide variety of shapes can be generated with twists combination, constructing cones with different kinds of basis. Motion estimation has been chosen here as an application of this cone-pixel camera model to show its reliability. The use of this model enables us to introduce a new cost function as the intersection of cones in space. We show here through experiments that BA is unable to estimate the correct motion because the solution does not correspond to a minimum of its cost function. However it can be found with the cone intersection criterion using a stochastic optimization method like Simulated Annealing [12].

Recently, cones have been introduced to modelize the uncertainties of ray directions rather than the pixel field of view. The most related work have been done by Perwass et al. [14] by showing how the uncertainty of all elements of the Geometric Algebra of conformal space can be appropriately described by covariance matrices. Giving an uncertain expression of the projection point, this approach can modelize noncentral sensors. In [11], cone aperture is set as an arbitrary error parameter of matched points. Other approaches which do not use non-least-square minimization methods have been used to correct these problems in multiple view geometry problematics. All these methods still remain mathematical instead of physical approaches [8, 9] and consider that nonintersections reflect necessarily an imprecise calibration result despite that it corresponds to the real geometry.

This chapter is structured as follows. After introducing the basis element of Conformal Geometric Algebra in Sect. 2, we describe in Sect. 3 the mathematical formulation of the general pixel-cones model using twists [17]. An experimental protocol to find the pixel cones of light is presented in Sect. 4, and results obtained from this protocol are applied on a pinhole camera and a catadioptric sensor. In Sect. 5, we introduce a cone intersection score function to address the motion estimation problem and present results in simulation experiments. Conclusions and future works are included in Sect. 6.

2 Introduction to Conformal Geometric Algebra

This section has been widely inspired by [15], which presents a good understanding and more detailed introduction to Geometric Algebra. The reader unfamiliar with CGA should refer to [6, 7] for an overview, and examples of its use in computer vision can be found in [13, 16].

2.1 Geometric Algebras

Geometric Algebra can be seen as a Clifford Algebra with its focus on a suited geometric interpretation. A Geometric Algebra $\mathcal{G}_{p,q,r}$ is a nonlinear space of dimension 2^n, $n = p + q + r$, with a space structure, called blades, to represent so-called multivectors as higher-grade algebraic entities in comparison to vectors of a vector space as first grade entities. A geometric Algebra $\mathcal{G}_{p,q,r}$ is constructed from a vector space $\mathbb{R}^{p,q,r}$, endowed with the signature (p, q, r), by application of a *geometric product*. This means that the generating vector space is always an element of its generated geometric algebra, and therefore the vectors of the vector space can be found as elements in each geometric algebra.

The product defining a geometric algebra is called *geometric product* and is denoted by juxtaposition, e.g., **AB** for two algebraic elements **A** and **B** called multivectors. The geometric product of vectors consists of an outer ($\wedge$) product and an inner (.) product. Their effect is to increase or to decrease the grade of the algebraic entities, respectively. Let $\mathbf{e}_i, \mathbf{e}_j \in \mathbb{R}^{p,q,r}$ be two orthonormal basis vectors of the vector space. Then the geometric product for these vectors of the geometric algebra $\mathcal{G}_{p,q,r}$ is defined as

$$\mathbf{e}_i\mathbf{e}_j := \begin{cases} 1 & \text{for } i = j \in \{1, \ldots, p\}, \\ -1 & \text{for } i = j \in \{p+1, \ldots, p+q\}, \\ 0 & \text{for } i = j \in \{p+q+1, \ldots, n\}, \\ \mathbf{e}_{ij} = \mathbf{e}_i \wedge \mathbf{e}_j = -\mathbf{e}_j \wedge \mathbf{e}_i & \text{for } i \neq j. \end{cases}$$

The geometric product of the same two basis vectors leads to a scalar, whereas the geometric product of two different basis vectors leads to a new entity, which is called a *bivector*. Geometric algebras can be expressed on the basis of graded elements. Scalars are of grade zero, vectors of grade one, bivectors of grade two, etc. A linear combination of elements of different grades is called a multivector **M**, which can be expressed as

$$\mathbf{M} = \sum_{i=0}^{n} \langle \mathbf{M} \rangle_i,$$

where the operator $\langle . \rangle$ denotes the projection of a general mutlivector to the entities of grade s. A multivector **A** of grade i can be written as $\mathbf{A}_{\langle i \rangle}$.

The inner (.) and outer ($\wedge$) products of two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^{4,1}$ are defined as

$$\mathbf{u}.\mathbf{v} := \frac{1}{2}(\mathbf{uv} + \mathbf{vu}), \tag{1}$$

$$\mathbf{u} \wedge \mathbf{v} := \frac{1}{2}(\mathbf{uv} - \mathbf{vu}). \tag{2}$$

Two blades of highest grade are called pseudoscalars and noted as $\pm\mathbf{I}$. The *dual* $\mathbf{X}^*$ of a blade $\mathbf{X}$ is defined by

$$\mathbf{X}^* := \mathbf{X}\mathbf{I}^{-1}.$$

It follows that the dual of an r-blade is an $(n-r)$-blade. The reverse $\tilde{\mathbf{A}}_{\langle S\rangle}$ of an s-blade $\mathbf{A}_{\langle S\rangle} = \mathbf{a}_1 \wedge \cdots \wedge \mathbf{a}_s$ is defined as the reverse outer product of the vectors $\mathbf{a}_i$,

$$\tilde{\mathbf{A}}_{\langle S\rangle} = (\mathbf{a}_1 \wedge \mathbf{a}_2 \wedge \cdots \wedge \mathbf{a}_{s-1} \wedge \mathbf{a}_s)^{\tilde{}},$$

$$\tilde{\mathbf{A}}_{\langle S\rangle} = (\mathbf{a}_s \wedge \mathbf{a}_{s-1} \wedge \cdots \wedge \mathbf{a}_2 \wedge \mathbf{a}_1).$$

The *join* $\mathbf{A}\dot{\wedge}\mathbf{B}$ is the pseudoscalar of the space given by the sum of spaces spanned by $\mathbf{A}$ and $\mathbf{B}$. For blades $\mathbf{A}$ and $\mathbf{B}$, the *dual shuffle* product $\mathbf{A} \vee \mathbf{B}$ is defined by the DeMorgan rule

$$(\mathbf{A} \vee \mathbf{B})^* := \mathbf{A}^* \dot{\wedge} \mathbf{B}^*.$$

For blades $\mathbf{A}$ and $\mathbf{B}$, it is possible to use the join to express *meet* operations:

$$\mathbf{A} \vee \mathbf{B} := \left(\mathbf{A}J^{-1} \wedge \mathbf{B}J^{-1}\right)J \tag{3}$$

with $J = \mathbf{A}\dot{\wedge}\mathbf{B}$.

2.2 Conformal Geometric Algebra (CGA)

A *Minkowski* plane is used to introduce CGA. Its vector space $\mathbb{R}^{1,1}$ has the orthonormal basis $\{\mathbf{e}_+, \mathbf{e}_-\}$ defined by the properties

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

In addition, a *null basis* can now be introduced by the vectors

$$\mathbf{e}_0 = \frac{1}{2}(\mathbf{e}_- - \mathbf{e}_+) \quad \text{and} \quad \mathbf{e} = \mathbf{e}_- + \mathbf{e}_+.$$

These vectors can be interpreted as the origin $\mathbf{e}_0$ of the coordinate system and the point at infinity $\mathbf{e}$, respectively. Furthermore, $\mathbf{E}$ is defined as $\mathbf{E} := \mathbf{e} \wedge \mathbf{e}_0 = \mathbf{e}_+ \wedge \mathbf{e}_-$.

The role of the Minkowski plane is to generate null vectors, and so to extend an Euclidean vector space $\mathbb{R}^n$ to $\mathbb{R}^{n+1,1} = \mathbb{R}^n \oplus \mathbb{R}^{1,1}$. The conformal vector space derived from $\mathbb{R}^3$ is thus denoted as $\mathbb{R}^{4,1}$, and a basis is given by $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_+, \mathbf{e}_-\}$. The conformal unit pseudoscalar is denoted as

$$\mathbf{I}_C = \mathbf{e}_{+-123} = \mathbf{E}\mathbf{I}_E,$$

where $\mathbf{I}_E$ is the unit pseudoscalar in the Euclidean Geometric Algebra, i.e., $\mathbf{I}_E = \mathbf{e}_{123}$. The points in CGA are related to those of Euclidean space by

$$\underline{\mathbf{x}} = \mathbf{x} + \frac{1}{2}\mathbf{x}^2\mathbf{e} + \mathbf{e}_0.$$

In CGA, spheres can be interpreted as the basis entities from which the other entities are derived. A sphere with center $\mathbf{p} \in \mathcal{G}_3$ and radius $\rho \in \mathbb{R}^3$ can be written as

$$(\mathbf{x} - \mathbf{p})^2 = \rho^2. \tag{4}$$

It turns out that a point $\mathbf{x}$ is nothing more than a degenerate sphere with radius $\rho = 0$. Equation (4) can therefore be represented more compactly as

$$(\mathbf{x} - \mathbf{p})^2 = \rho^2 \quad \Longleftrightarrow \quad \underline{\mathbf{x}}.\underline{\mathbf{s}} = 0.$$

A sphere can be defined with its dual form which can be calculated directly from at least four points on it,

$$\underline{\mathbf{s}}^* = \underline{\mathbf{a}} \wedge \underline{\mathbf{b}} \wedge \underline{\mathbf{c}} \wedge \underline{\mathbf{d}},$$

and a point $\underline{\mathbf{x}}$ is on this sphere if and only if

$$\underline{\mathbf{x}}.\underline{\mathbf{s}} = 0 \quad \Longleftrightarrow \quad \underline{\mathbf{x}} \wedge \underline{\mathbf{s}}^* = 0.$$

Geometrically, a circle $\underline{\mathbf{z}}$ can be described by the intersection $\underline{\mathbf{z}} = \underline{\mathbf{s}_1} \wedge \underline{\mathbf{s}_2}$ of two linearly independent spheres $\underline{\mathbf{s}_1}$ and $\underline{\mathbf{s}_1}$. This means that

$$\underline{\mathbf{x}} \in \underline{\mathbf{z}} \quad \Longleftrightarrow \quad \underline{\mathbf{x}}.\underline{\mathbf{z}} = 0.$$

In the same way, the dual form of a circle is geometrically defined by three points on it,

$$\underline{\mathbf{z}}^* = \underline{\mathbf{a}} \wedge \underline{\mathbf{b}} \wedge \underline{\mathbf{c}},$$

and a line is a degenerate case of a circle passing through a point at infinity:

$$\underline{\mathbf{L}}^* = \underline{\mathbf{e}} \wedge \underline{\mathbf{a}} \wedge \underline{\mathbf{b}}.$$

The bivectors of the geometric algebra can be used to represent rotations of points in the 3D space. A *rotor* $\mathbf{R}$ is an even grade element of the algebra which satisfies $\mathbf{R}\tilde{\mathbf{R}} = 1$. By using the Euler representation of a rotor, we have

$$\mathbf{R} = \exp\left(-\frac{\theta}{2}n\right).$$

The rotation of a point represented by its vector $\mathbf{x}$ can be carried out by multiplying the rotor $\mathbf{R}$ from the left and its reverse from the right to the point such as

$$\mathbf{x}' = \mathbf{R}\mathbf{x}\tilde{\mathbf{R}}.$$

CGA enables one a multiplicative expression of translation $\mathbf{t}$ as a special rotation acting at infinity by using the null vector $\mathbf{e}$:

$$\underline{\mathbf{x}}' = \mathbf{T}\underline{\mathbf{x}}\tilde{\mathbf{T}} \quad \text{with } \mathbf{T} = \exp\left(-\frac{\mathbf{et}}{2}\right).$$

It is possible in GA to generate kinematic shapes which result from the orbit effect of points under the action of a set of coupled operators. The nice idea is that the operators are what describes the curve (or shape). To model a rotation of a point $\underline{\mathbf{X}}$ around an arbitrary line $\underline{\mathbf{L}}$ in the space, the general idea is to translate the point $\underline{\mathbf{X}}$ with the distance vector between the line $\underline{\mathbf{L}}$ and the origin, to perform a rotation and to translate the transformed point back. So a motor $\mathcal{M}$ describing a general rotation has the form

$$\mathcal{M} = \mathbf{T}\mathbf{R}\tilde{\mathbf{T}}.$$

Screw motions can be used to describe rigid motions by combining a rotation around an axis with a translation parallel to that axis $\mathbf{T}_{d\mathbf{n}}$. The resulting motor is

$$\mathbf{M} = \mathbf{T}_{d\mathbf{n}}\mathbf{T}\mathbf{R}\tilde{\mathbf{T}}.$$

As introduced in [17], these operators are the motors which are the representation of $SE(3)$ in $\mathcal{R}_{4,1}$. The use of twists gives a compact representation of cones and brings the heavy computation of the intersection of two general cones to a simple intersection of lines.

3 General Model of a Cone-Pixels Camera

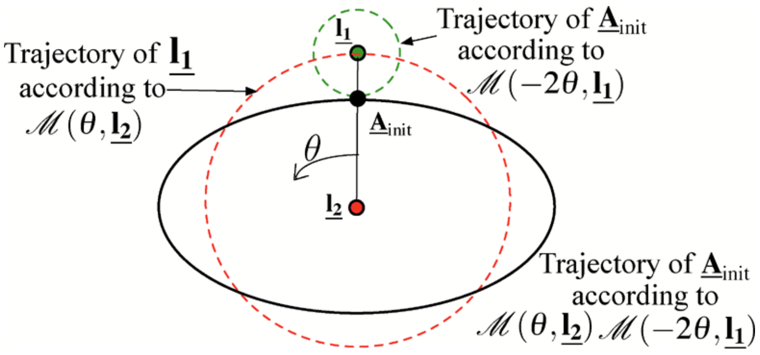
This section will present a general model of a camera using cone-pixels. There are several possible ways to write the equation of a cone. A single-sided cone with vertex V, axis ray with origin at V, unit-length direction A, and cone angle $\alpha \in (0, \pi/2)$ is defined by the set of points X such that vector $X - V$ forms an angle α with A. The algebraic condition is $A \cdot (X - V) = |X - V|\cos(\alpha)$. The solid cone is the cone plus the region it bounds, specified as $A \cdot (X - V) \geq |X - V|\cos(\alpha)$. It is somewhat painful to compute the intersection of two cones, and this can become even more complicated integrating rigid motion parameters between cones. CGA is used to enable us a simple formulation of cones using twists, as introduced in Sect. 2.

3.1 Geometric Settings

As shown in Fig. 3 in the case of a perspective camera, the image plane here represented by I contains several rectangular pixels $p(i, j)$, where i, j corresponds to the position of the pixel. Considering $p(i, j)$, its surface is represented by a rectangle defined by points $A_0 \ldots A_8$, with A_0 corresponding to the center of the rectangle.

Given a line $\underline{\mathbf{l}}$ (with unit direction) in space, the corresponding motor describing a general rotation around this line is given by $\mathcal{M}(\theta, \underline{\mathbf{l}}) = \exp(-\frac{\theta}{2}\underline{\mathbf{l}})$. The general rotation of a point $\underline{\mathbf{x}}$ around any arbitrary line $\underline{\mathbf{l}}$ is

$$\underline{\mathbf{x}}' = \mathcal{M}(\theta, \underline{\mathbf{l}})\underline{\mathbf{x}}\tilde{\mathcal{M}}(\theta, \underline{\mathbf{l}}). \tag{5}$$



Trajectory of $\underline{\mathbf{l_1}}$ according to $\mathcal{M}(\theta,\underline{\mathbf{l_2}})$
$\underline{\mathbf{l_1}}$
Trajectory of $\underline{\mathbf{A}}_{\text{init}}$ according to $\mathcal{M}(-2\theta,\underline{\mathbf{l_1}})$
$\underline{\mathbf{A}}_{\text{init}}$
θ
$\underline{\mathbf{l_2}}$
Trajectory of $\underline{\mathbf{A}}_{\text{init}}$ according to $\mathcal{M}(\theta,\underline{\mathbf{l_2}})\mathcal{M}(-2\theta,\underline{\mathbf{l_1}})$

Fig. 2 Generating an ellipse with a *2twist* combination

The general form of the *2twist* generated curve is the set of points $\underline{\mathbf{x}}'$ defined as

$$\underline{\mathbf{x}}' = \mathcal{M}(\lambda_2\theta,\underline{\mathbf{l}_2})\mathcal{M}(\lambda_1\theta,\underline{\mathbf{l}_1})\underline{\mathbf{x}}\tilde{\mathcal{M}}(\lambda_1\theta,\underline{\mathbf{l}_1})\tilde{\mathcal{M}}(\lambda_2\theta,\underline{\mathbf{l}_2}). \tag{6}$$

In the following, we are interested in generating ellipses to approximate the form of the pixel rather than squares. Ellipses are indeed expressible mathematically in such compact form as it will be shown in what follows, while using square instead will involve discontinuities in the expression of the cones. In the previous equation, an ellipse corresponds to the values $\lambda_1 = -2$ and $\lambda_2 = 1$. $\underline{\mathbf{l}_1}$ and $\underline{\mathbf{l}_2}$ are the two rotation axes needed to define the ellipses [17]. An ellipse generated with twists is illustrated in Fig. 2.

Considering a single pixel $p_{i,j}$ (see Fig. 3), its surface can be approximated by the ellipse generated by a point A that rotates around point A_0 with a rotation axis corresponding to $\mathbf{e}_3$ normal to the plane I. The ellipse $\mathcal{E}^{i,j}$ generated corresponding to the pixel $p_{i,j}$ is the set of all the positions of $A(\theta)$:

$$\forall\theta \in [0,\ldots,2\pi],$$
$$\mathcal{E}^{i,j} = \left\{\underline{\mathbf{A}}(\theta) = \mathcal{M}(\theta,\underline{\mathbf{l}_2})\mathcal{M}(-2\theta,\underline{\mathbf{l}_1})\underline{\mathbf{A}}_{\text{init}}\tilde{\mathcal{M}}(-2\theta,\underline{\mathbf{l}_1})\tilde{\mathcal{M}}(\theta,\underline{\mathbf{l}_2})\,|\right\}.$$

The initial position of $\underline{\mathbf{A}}$ is $\underline{\mathbf{A}}_{\text{init}}$. The elliptic curve is generated by setting the two connected twists in order to obtain an ellipse with principal axes $(\overline{A_8A_0}, \overline{A_6A_0})$ in order to fit the rectangular surface of the projection of the pixel as shown in Fig. 3. It is now possible to generate the cone corresponding to the field of view of the pixel. To set the different lines, we use the *dual* expression of geometric objects in CGA. We set the line $\underline{\mathbf{l}}^*_{0_{i,j}}$, the cone axis corresponding to $p_{i,j}$, as

$$\underline{\mathbf{l}}^*_{0_{i,j}} = e \wedge \underline{\mathbf{O}} \wedge \underline{\mathbf{A}}_0.$$

The generatrix of the cone is the line joining O to A. Since the position of A depends on the cone aperture α, the generatrix is noted as $\underline{\mathbf{l}_{\mathbf{OA}(\alpha)}}^{i,j}$.

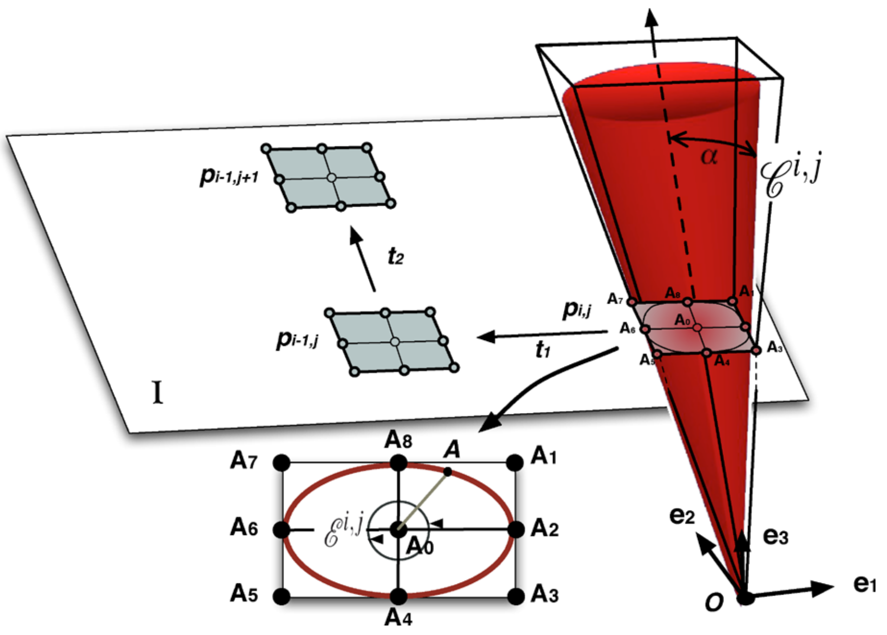


Fig. 3 Cones Geometric settings

A priori, one can think that the pixel-cone of view of $p_{i,j}$ is the cone $\mathcal{C}^{i,j}(\alpha)$ defined by the projection point and the surface of the pixel,

$$\mathcal{C}^{i,j}(\alpha) = \mathcal{M}(\theta, \underline{\mathbf{l}_{0_{i,j}}}) \underline{\mathbf{l}_{\mathbf{OA}(\alpha)}}^{i,j} \tilde{\mathcal{M}}(\theta, \underline{\mathbf{l}_{0_{i,j}}}), \tag{7}$$

with $\mathcal{M}(\theta, \underline{\mathbf{l}_{0_{i,j}}}) = \exp(-\frac{\theta}{2}\underline{\mathbf{l}_{0_{i,j}}})$, but it will appear that this first intuition is not true because of the optical refraction combined with proper camera design. However, this cone has to be computed and used as a preliminary guess to find the real cone of view which can only be wider. In the following, this overlapping will be pointed out experimentally, and overlapping cones of view of neighboring pixels are shown in Fig. 9. Two criterions will be introduced taking into account an eventual overlapping.

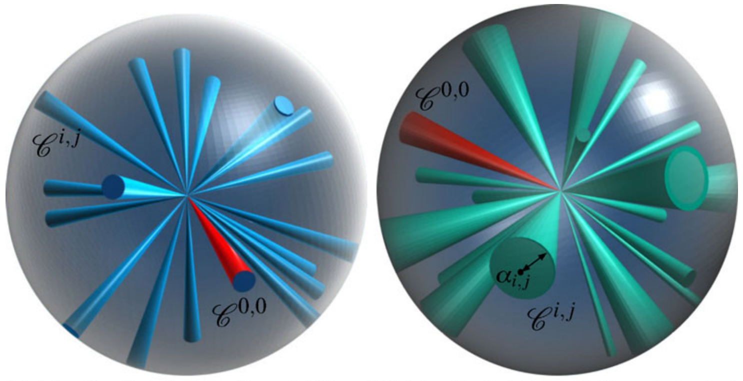
Note the equivalent expression of the cone using the outer product generating a line after having generated an ellipse from the point $A(\alpha)$:

$$\mathcal{C}^{i,j}(\alpha) = e \wedge \underline{\mathbf{O}} \wedge \left(\mathcal{M}(\theta, \underline{\mathbf{l}_{0_{i,j}}}) \underline{\mathbf{A}(\alpha)}^{i,j} \tilde{\mathcal{M}}(\theta, \underline{\mathbf{l}_{0_{i,j}}})\right). \tag{8}$$

The same process is to be applied again after translating the pixel $p_{i,j}$ using $\mathbf{t}_1$ and $\mathbf{t}_2$, which corresponds to the translation to switch from one pixel to the other. The projection $p_{i,j}$ of a pixel is moved to a next pixel:

$$p_{i+1,j+1} = \mathcal{T}^2(\mathbf{t}_2)\mathcal{T}^1(\mathbf{t}_1)p_{i,j}\tilde{\mathcal{T}}^1(\mathbf{t}_1)\tilde{\mathcal{T}}^2(\mathbf{t}_2),$$

where $\mathcal{T}(\mathbf{t})$ corresponds to a translation operator in CGA.



(a) A bundle of pixel-cones of a central linear sensor. (b) A bundle of pixel-cones of a central variant linear sensor

Fig. 4 Different configurations of pixel-cones in the case of linear and variant scale sensors. In *red* the principal cone according which every other is located

3.2 The General Model of a Central Cone-Pixel Camera

The general form of a central sensor, whether it has linear resolution (cones vary slightly) or variant resolution, is the expression of a bundle of cones. All cones $\mathcal{C}^{i,j}$ will be located using spherical coordinates and located according to an origin set as the cone $\mathcal{C}^{0,0}(\alpha)$ that has $\mathbf{e}_3$ as a principal axis. The general form of a central linear scale camera (Fig. 4(a)) is then simply given by

$$\mathcal{C}^{i,j}_{\phi,\psi}(\alpha) = \mathcal{M}(\psi, \mathbf{e}_{23})\mathcal{M}(\phi, \mathbf{e}_{13})\mathcal{C}^{0,0}_{0,0}(\alpha)\tilde{\mathcal{M}}(\phi, \mathbf{e}_{13})\tilde{\mathcal{M}}(\psi, \mathbf{e}_{23}), \tag{9}$$

where ψ, ϕ denote the spherical coordinates of the cone, and α is the constant aperture for an uniform resolution.

The general form of a central variant scale sensor is slightly different. Each cone having a different aperture α, cones need to be defined according to their position. The general form becomes

$$\mathcal{C}^{i,j}_{\phi,\psi}(\alpha_{i,j}) = \mathcal{M}(\psi, \mathbf{e}_{23})\mathcal{M}(\phi, \mathbf{e}_{13})\mathcal{C}^{0,0}_{0,0}(\alpha_{i,j})\tilde{\mathcal{M}}(\phi, \mathbf{e}_{13})\tilde{\mathcal{M}}(\psi, \mathbf{e}_{23}). \tag{10}$$

3.3 Intersection of Cones

The previous formulation of cones established in (7) is a parameterized line bundle and cannot be considered as a GA entity and cannot be used as easier with all GA tools. Then to express the intersection of two cones $\mathcal{C}^{i_m,j_m}_{\phi_m,\psi_m}$ and $\mathcal{C}^{i_n,j_n}_{\phi_n,\psi_n}$, we used the usual $\cap$ operator instead of using a GA meet operator defined in (3) which should have been the correct expression of the intersection of classic objects in GA.

The result is a set of points of intersection $P_{m,n}$ between the generatrix lines of the cones,

$$P_{m,n} = \left\{ \mathcal{C}^{i_m, j_m}_{\phi_m, \psi_m} \cap \mathcal{C}^{i_n, j_n}_{\phi_n, \psi_n} \right\}. \tag{11}$$

If the intersection exists, $P_{m,n}$ is not empty, and the set of points then forms a convex hull the volume of which can be computed using [1].

4 General Cone-Pixel Camera Calibration

4.1 Experimental Protocol

Cones being at the heart of the model, we will now give an experimental setup of the calibration procedure to provide an estimation of the cone of view of each pixel of a camera. The method is not restricted to a specific camera geometry; it relies on the use of multiple planes calibration [4, 20]. As shown in Fig. 5, the camera to be calibrated is observing a calibration plane (in our case, a computer screen), the aim is to estimate the cone of view of a pixel $p_{i,j}$ by computing for each position of the screen its projection surface $SP^k(i,j)$, k being the index of the calibration plane. The metric is provided using a reference high-resolution calibrated camera (RC)[1] observing the calibration planes, whose positions and metrics can then be known in the RC coordinates. The impact surfaces $SP^k(i,j)$, once determined on each screen, normally lead (as shown in Fig. 5) to the determination of all pixel-cones parameters.

Figure 6 shows the experimental setup carried out for the experiments. The key-point of the calibration protocol therefore relies on the determination of pixels' impact $SP^k(i,j)$.

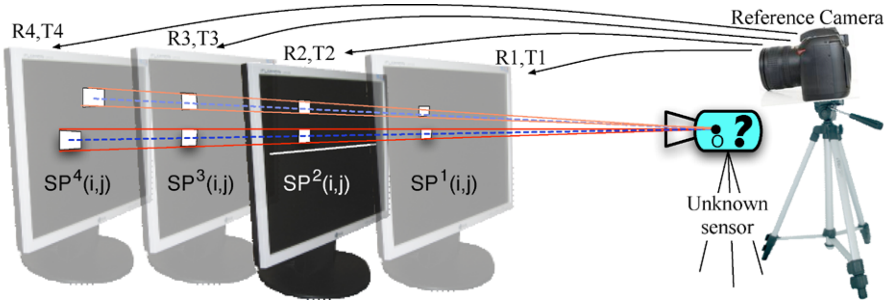


Fig. 5 Experimental protocol: Cone construction and determination of the sensor projection center The R_i, T_i represent the rigid motion between the reference camera and the calibration planes coordinate systems

[1] 6 Megapixel digital single-lens Nikon D70 reflex camera fitted with 18–70-mm Nikkor micro lens. The micro lens and the focus are fixed during the whole experiment.

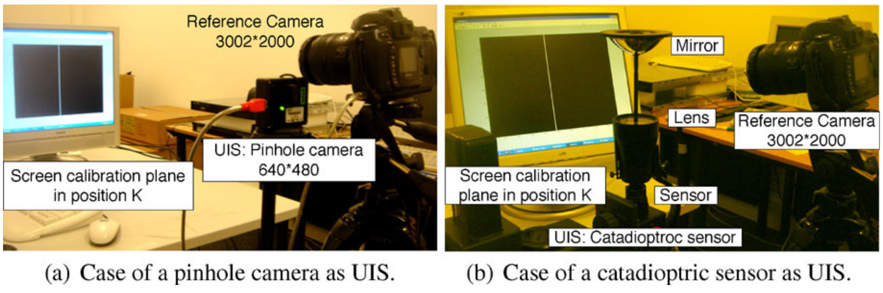


Fig. 6 Experimental protocol for two kinds of unknown image sensor

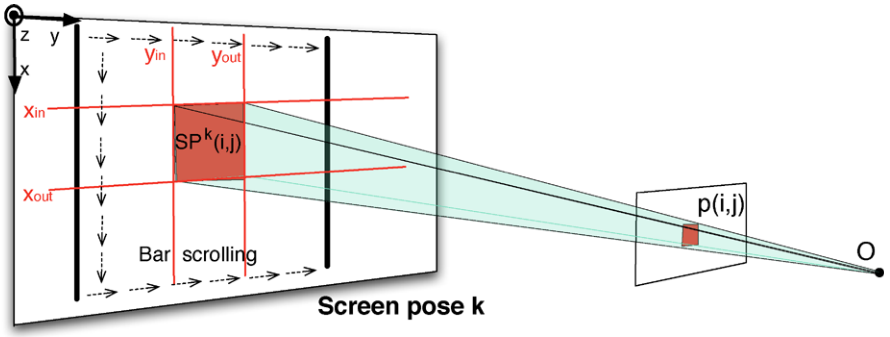
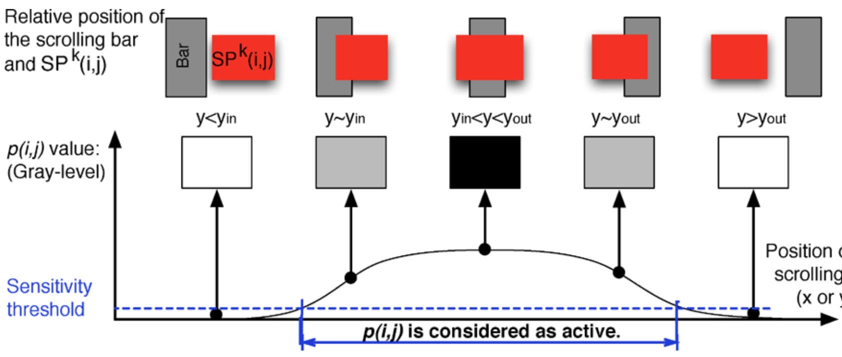


Fig. 7 Intersection surface $SP^k(i, j)$ between the calibration plane k and the cone $\mathcal{C}(i, j)$

The activity of each pixel $p(i, j)$ is then tracked while RC observes the screen calibration planes (see Fig. 7). At each position the screen displays a white bar scrolling on a uniform black background (see Fig. 6). The bar will cause a change in the grey level values of pixels when it is in their cone of vision. The pixels' gray level increases from a minimum value (when the bar is outside $SP^k(i, j)$) to a maximum value (when the bar is completely inside $SP^k(i, j)$) and decreases down to zero when the bar is again outside $SP^k(i, j)$. Figure 8 gives a visual explanation of the process.

A sensitivity threshold can be chosen to determine pixels' activation. Using the reference camera calibration results, it is then possible, once $SP^k(i, j)$ is determined, to compute its edges as the positions of y_{in} and y_{out} in the RC coordinate system. The bar is scrolled in two orthogonal directions producing two other edges x_{in} and x_{out} (Fig. 7). At this stage, the edges of $SP^k(i, j)$ are then completely known. The location and size of pixel-cones can then in a second stage be estimated once all $SP^k(i, j)$ are known. Cones are computed using the center of $SP^k(i, j)$ providing the rotation axis, the cone envelope is given by computing rays that pass through all the intersection points of the vertex of each $SP^k(i, j)$ corresponding to each pixel (Fig. 5).



Relative position of the scrolling bar and SP^k(i,j)
Bar
SP^k(i,j)
y<yin
y~yin
yin<y<yout
y~yout
y>yout
p(i,j) value: (Gray-level)
Sensitivity threshold
Position of the scrolling bar (x or y)
p(i,j) is considered as active.

Fig. 8 Pixel response according to the scroll bar position

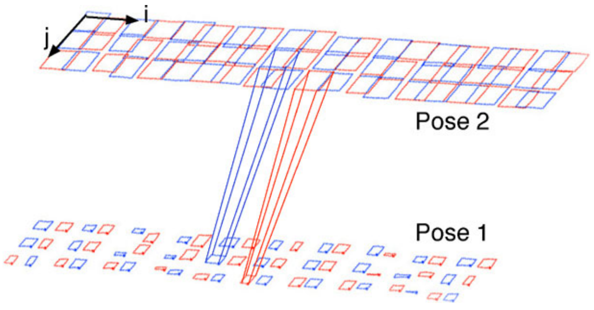


Fig. 9 Experimental results: Cones of view in the case of a pinhole camera

4.2 *Calibration Experimental Results*

The following experiments were carried out using PointGrey DragonFly®2, with a 640×480 resolution, and a parabolic catadioptric sensor with a telecentric lens; both are central sensors (Fig. 6(a)). Figure 9 shows cones reconstruction on $SP^1(i, j)$ and $SP^2(i, j)$ in the case of a pinhole camera. For clarity, only two cones were drawn. As expected, the results show repetitive pattern as corresponding pixel's impact. We can see few bad measurements, especially in Pose 1. Therefore, we use four planes to avoid this problem, using the redundancy of the information. This figure shows that the spatial sensitivity of pixels overlaps and does not correspond to the dimension of the pixels as the informed reader could imagine.

With a catadioptric camera, it is a geometric truth that the aperture angle of each cone increases as pixels are set off the optical axis of the camera. This phenomenon is experimentally shown in Fig. 10, which represents the evolution of the solid angle of pixel-cones. In principle, the solid angle should not vary according to the position of the calibration plane that was used to compute it. The curves are logically very close even if a small bias appears for very large cone-pixels at the periphery of the mirror (where the uncertainties of the measure on the surface due to the nonlinearity of the mirror are the highest).

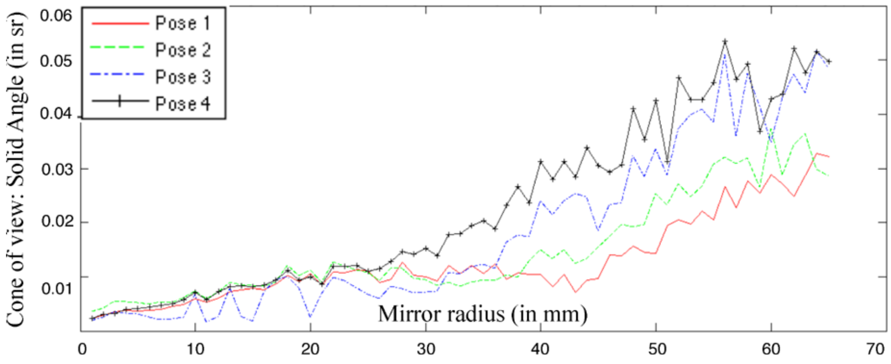


Fig. 10 Solid angle of view according to the mirror radius

Table 1 Central projection point estimation coordinates: case of a pinhole camera

	Ground truth	Axis estimation	Error	Apex estimation	Error
x	−78.33	−78.97	0.65	−78.97	0.65
y	45.36	44.08	1.28	44.07	1.29
z	45.74	57.89	12.15	57.90	12.16

The method allows us the estimation of the central point position of the calibrated sensor. In the case of a pinhole camera the calibration screens were located between 800–1050 mm from the reference camera, whilst the camera to be calibrated was set a few centimeters away (see Fig. 6). In order to obtain a ground truth data, the pinhole camera was calibrated using the classic ray method [19]. Three positions of the optic center are then computed for comparison. The first is given by the classic calibration, the second by the intersection of the rotation axis of estimated cones, and the last by the intersection of all rays (traced as shown in Fig. 9) representing estimated cones. The results are shown in Table 1.

A different single viewpoint is found in each case. There are slight variations in the position of the center in the third coordinate. This can be explained by the fact that the calibrated portion of the sensor used to estimate cones is limited (55×60 pixels located around the center of the image). In this configuration, the depth estimation is obviously less accurate. Concerning the catadioptric sensor, the results show that the cones intersect at a single point. The combination of a parabola and a telecentric lens can only produce a central sensor, which proves the method to be efficient. The estimation of the position of the viewpoint using the principal axis of the estimated cones and all the rays that form the estimated cones produce similar results (in mm: $x = -23.38,\ y = 147.55,\ z = 384.79$ and $x = -23.32,\ y = 147.33,\ z = 385.57$). The mean distance between the rotation axis and their estimated single point is 3.71 mm. The mean distance between the apex and their estimated single point is 2.97 mm.

5 Motion Estimation

5.1 Problem Formulation

Estimating relative camera motion from two calibrated views is a classic problem in computer vision. Till now, this problem has been expressed as minimizing the geometric distances between the measured image features and the reprojected ones with the new motion parameters [10, 11, 18]. In this paper we would like to propose a new geometrical criteria defined with cone intersections which better fits the physical reality while providing more accurate results.

Consider two views I_1 and I_2 of a scene acquired by the same full calibrated projective camera moved from a first to a second location. We assume that this calibration step has provided the intrinsic parameters in the two following ways: Cone-Pixel model for the test presented here and pinhole camera model to ensure a reliable comparison with existing techniques like Bundle Adjustment. We assume that the camera is successively located at $[\mathbf{I}|\mathbf{0}]$ and $[\mathbf{R}|\mathbf{t}]$, where $\mathbf{R}$ is the rotation, and $\mathbf{t}$ is the translation of the motion. Let now X be a set of N 3D feature points and $x_1 \leftrightarrow x_2$ the sets of image points observed respectively in the first and the second image. The problem statement is therefore the following: knowing the measurement x_1 and x_2, how to retrieve the motion $[\mathbf{R}|\mathbf{t}]$.

From a physical point of view and as described above, both rays never strongly intersect (see Fig. 11(a)). The cone defined by the surface of the pixel encompasses all the rays of view of the pixel (Fig. 11(b)), and each ray of view corresponds to the directrix of each cone. A reliable pixel correspondence involves that these two pixel cones of view have a nonnull intersection Fig. 11(c). Noise measurement is not required to encounter this problem, since due to the pixel sampling, it arises in every case of numerical cameras. However, noise will be taken into account in the following section.

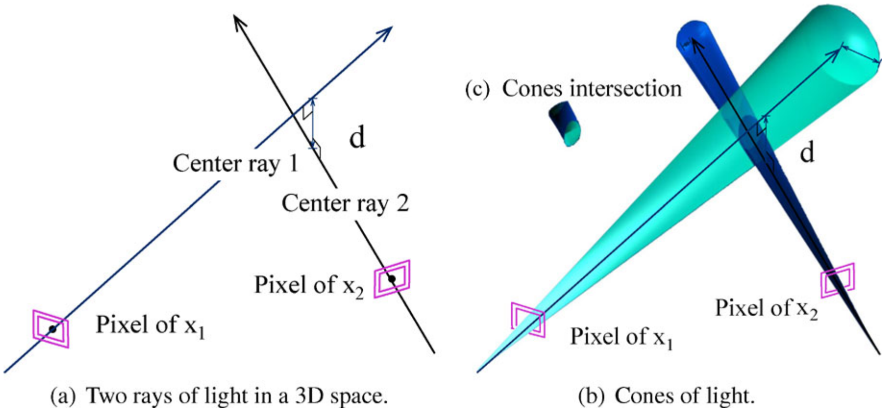


Fig. 11 Difference between rays and cones of light intersections

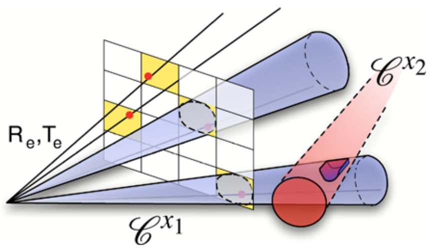


Fig. 12 Motion estimated with cones intersections criterion

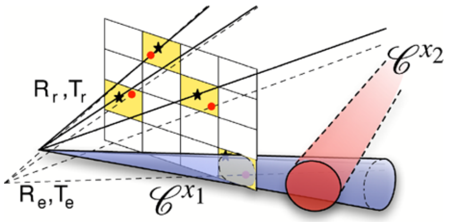


Fig. 13 Motion estimated with reprojection error criterion

It comes then that finding a discrete motion minimizing these quantities for all points could lead to a solution $[\mathbf{R}_r|\mathbf{t}_r]$ which does not provide nonnull cone intersections for all reliable correspondences. An illustration of this major drawback of reprojection error minimization is illustrated in Figs. 12 and 13.

Figure 12 shows an example of four 3D point projections assuming the exact motion $[\mathbf{R}_e, \mathbf{t}_e]$ (for clarity purposes, only a few cones of view are drawn). In this example, most of projected points are located on the pixels' periphery. It follows that using a minimization of the reprojection error as a cost function to estimate a new motion $[\mathbf{R}_r, \mathbf{t}_r]$ could involve the set out of Fig. 13, where the reprojection error is lower (see the black stars representing the reprojected points) even if it eliminates a correspondence of cones which no longer intersect.

5.2 Cone Intersection Score Functions

We introduce here an initial score function S_1 computed according to the following steps: we first compute, for each correspondence, the minimal cone aperture α such that the intersection

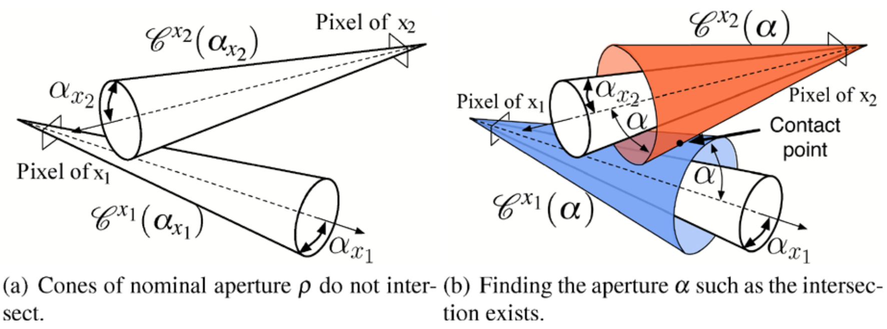
$$P_{x_1,x_2} = \left\{\mathfrak{C}^{x_1}(\alpha) \vee \mathfrak{C}^{x_2}(\alpha)\right\} \tag{12}$$

exists (see the (7) in Sect. 3.3 and Fig. 14).

A binary score s_1 (Fig. 15(a)) is provided for each match according to α by the expression

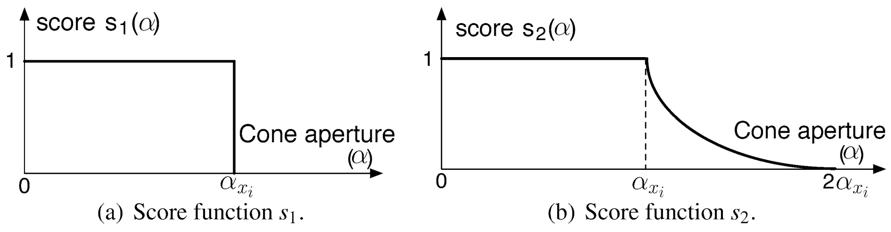
$$s_1(\alpha) = u(\alpha) - u(\alpha - \rho), \tag{13}$$

where $u(\alpha)$ is the classic Heaviside step function ($u(\alpha) = 0$ if $\alpha < 0$ and $u(\alpha) = 1$ otherwise), and ρ is the nominal pixel aperture, which is known from the full cone-



(a) Cones of nominal aperture ρ do not intersect. (b) Finding the aperture α such as the intersection exists.

Fig. 14 Case of nonintersecting nominal cones



(a) Score function s_1. (b) Score function s_2.

Fig. 15 Score functions expressions

pixel calibration step. The total score corresponding to the motion estimation is given by

$$S_1(\mathbf{R},\mathbf{t}) = \sum_{n=1}^{N} s_1(\alpha_k). \tag{14}$$

This score function statement is *the number of correspondences with strongly intersecting cones, given a motion estimation*. It comes from this expression that

$$0 \le S_1(\mathbf{R},\mathbf{t}) \le N \quad \text{and} \quad S_1(\mathbf{R},\mathbf{t}) \in \mathbb{N}. \tag{15}$$

Assuming that there is no error in the pixel matching, a solution can be found such that $S_1(\mathbf{R},\mathbf{t}) = N$.

This constraint fits physical reality better than classic reprojection error but is difficult to use because of its discrete formulation. To ease its use in seeking strategies, a second continuous criterion inspired by S_1 is introduced. A score for each correspondence is computed (see Fig. 15(b)) according to

$$s_2(\alpha) = s_1(\alpha) + \left(2 - \frac{\alpha}{\rho}\right)^2 \big(u(\alpha-\rho) - u(\alpha - 2\rho)\big); \tag{16}$$

the global score function S_2 is then defined as

$$S_2(\mathbf{R},\mathbf{t}) = \sum_{n=1}^{N} s_2(\alpha_k). \tag{17}$$

In addition to being very close to the physical reality (it modelizes the small blurring between neighboring pixels pointed out in the calibration experiments, see Fig. 8), this score function provides good results, as will be shown in the following section.

Because of the convex hull computation, this cost function is not analytic. It will not be possible to use classic minimization (gradient descent, nonlinear programming, SOCP) as seeking strategies. The optimal motion solution according to the cone intersection cost function will be found using stochastic optimization. The method chosen here is Simulated Annealing (SA) [12] because of its simplicity and its convergence properties.

5.3 Simulation Experiments for Motion Estimation Using Cone Intersection Criterion

The validation of this method has been carried out in simulation to control the whole parameter set. It did not depend on possible error measurement or any noise. Let X be a set of $N = 592$ 3D points generated randomly in the field of view of two identical full calibrated (intrinsic and extrinsic parameters) 640×480 views I_1 and I_2. The optimal solution is then exactly known and will be noted as $[\mathbf{R}_e, |\mathbf{t}_e]$ (index e is chosen for "exact"). Many motion estimation algorithms for this problem have been developed. In this section we will review two of these according to cone intersection score functions. A first motion estimation is provided from the fundamental matrix [5] and is then used as the initial step for a BA [18]. Results provided by BA are finally modified with Simulated Annealing. This sequential protocol enables a step-by-step study of each method with the different cost and score functions discussed here. Results are shown in Table 2. It can be noticed that the score functions S_1 and S_2 can be computed to evaluate and compare the different guesses $(\mathbf{R},\mathbf{T})$. Therefore, the corresponding columns are not empty even for the Bundle Adjustment and for the Fundamental Matrix. Two particular facts can be drawn from these results:

(i) The reprojection error is greater for the perfect motion than for the BA solution, entailing that in this case BA is not able to reach the correct motion, since it does not correspond to the minimum of the cost function.
(ii) We considered two images of the same scene and 592 pixel correspondences between both images. There is a set of motion (R,T) between both frame capture such that every couple of cones intersect. Then we can be sure that the correct motion is in this set, and consequently, every motion which does not verify this property cannot be a correct motion. Both columns S_1 and S_2 show consequently that BA provides a nonrealistic solution.

Table 2 Motion estimation results: case of a simple pixel sampling

Method	Rep. error	S_1	S_2	Translation error (%)	Rotation error (%)	Iterations
Fundamental matrix	1.4×10^4	538	577.95	0.073	0.13	–
Bundle adjustment	13.80	587	588.57	0.17	0.054	304
Cone approach with SA	28.59	592	592	0.04	0.068	421
Exact motion $[\mathbf{R}_e, \mathbf{t}_e]$	13.84	592	592	0	0	–

Table 3 Motion estimation results: case of a Gaussian noise ($\sigma = 0.5$)

Method	Rep. error	S_1	S_2	Translation error (%)	Rotation error (%)	Iterations
Fundamental matrix	1.32×10^4	309	388.9	0.426	0.58	–
Bundle adjustment	27.97	471	516.52	0.428	0.084	288
Cone approach with SA	41.27	491	584.3	0.64	0.086	506
Exact motion $[\mathbf{R}_e, \mathbf{t}_e]$	28.05	490	534.92	0	0	–

Algorithm 1 Motion Estimation using cone intersection criterion

```
Require: Initial estimation $[\mathbf{R}_i, \mathbf{t}_i]$, nominal aperture $\rho_n$, $S$
 1: while $S \leq S_{ok}$ do
 2:     Generate a Guess $[\mathbf{R}, \mathbf{t}]$: $\mathbf{R} = \mathbf{R}_i + \lambda_R$, $\mathbf{t} = \mathbf{t}_i + \lambda_t$
 3:     for $n = 1, n \leq N$ do
 4:         find $\alpha$ such as $P_{x_1,x_2}$ exists (12)
 5:         $s_n = s_2(\alpha)$    (16)
 6:     end for
 7:     $S2 = \sum(s_n)$    (17)
 8:     if $S2 > S$ then
 9:         $S = S2$
10:         $\mathbf{R}_i = \mathbf{R}$ and $\mathbf{t}_i = \mathbf{t}$
11:         $\lambda_R = \frac{\lambda_R}{2}$ and $\lambda_t = \frac{\lambda_t}{2}$
12:     end if
13: end while
```

Rotations provided by both BA and Simulated Annealing are very close (difference less than 1×10^{-4}). However, a significant accuracy have been gained in translation estimation.

A second test is carried out by applying an additive Gaussian noise (standard deviation $\sigma = 0.5$) to the same data x_1 and x_2. Similarly, results provided by the different methods are shown in Table 3. It can be noticed that the results provided by BA are a little closer to the exact motion, compared to those provided by the cone intersection criterion and simulated annealing, even if they remain very similar. The notion of "better results" could not be used strictly speaking, because precisely of the noisy aspect. The reader should keep in mind that the Gaussian noise model used there can obviously be easier handled by a quadratic error minimization instead of a cone intersection criterion. Moreover, the score obtained by this presented method exceed those obtained for the exact motion. We can consider that this solution is satisfying and physically more reliable.

6 Conclusion and Future Works

This paper presented a general method to modelize cameras introducing the use of cones to give a better approximation of the pixels' field of view (rather than the usual use of lines). We also introduced an experimental protocol to estimate cones that is not restricted to any geometry of cameras. The model used Conformal Geometric Algebra that allowed us to handle cones in a simple manner using twists. This formulation enabled the introduction of a new pixel matching characterization as a nonnull intersection of cones of view. On this basis, it was possible to successfully address the motion estimation problem using this characterization as a new score function, with better results than classic ray approach. Simulated Annealing was chosen as a seeking strategy. A large panel of others methods could be used instead. The aim of this paper was not to discuss these strategies but to prove that cone intersection score criterion is closer to the physics and better to address computer vision problems such as motion estimation. Current work is focusing on these seeking strategies and the computation of a direct Cone Adjustment algorithm.

References

1. Barber, C.B., Dobkin, D.P., Huhdanpaa, H.: The quickhull algorithm for convex hulls. ACM Trans. Math. Softw. **22**(4), 469–483 (1996)
2. Benosman, R., Kang, S.: Panoramic Vision: Sensors, Theory, Applications. Springer, Berlin (2001)
3. Debaecker, T., Benosman, R.: Bio-inspired model of visual information codification for localization: from retina to the lateral geniculate nucleus. J. Integr. Neurosci. **6**(3), 1–33 (2007)
4. Grossberg, M.D., Nayar, S.K.: A general imaging model and a method for finding its parameters. In: ICCV, pp. 108–115 (2001)
5. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, Cambridge (2003)
6. Hestenes, D.: The design of linear algebra and geometry. Acta Appl. Math.: Int. Surv. J. Appl. Math. Math. Appl. **23**, 65–93 (1991)
7. Hestenes, D., Sobczyk, G.: Clifford Algebra to Geometric Calculus. Reidel, Dordrecht (1984)
8. Kahl, F., Hartley, R.: Multiple view geometry under the l-infinity norm. In: PAMI (2008)
9. Ke, Q., Kanade, T.: Quasiconvex optimization for robust geometric reconstruction. In: ICCV (2005)
10. Kim, J.-H., Hartley, R.I., Frahm, J.-M., Pollefeys, M.: Visual odometry for non-overlapping views using second-order cone programming. In: ACCV (2), pp. 353–362 (2007)
11. Kim, J.-H., Li, H., Hartley, R.: Motion estimation for multi-camera systems using global optimization (2008)

12. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**, 671–680 (1983)
13. Perwass, C.: Applications of geometric algebra in computer vision. Ph.D. thesis, University of Cambridge (2000)
14. Perwass, C., Gebken, C., Sommer, G.: Geometry and kinematics with uncertain data. In: ECCV (1), pp. 225–237 (2006)
15. Rosenhahn, B.: Pose estimation revisited. Ph.D. thesis, Christian-Albrechts-Universitat zu Kiel, Institut für Informatik und Praktische Mathematik (2003)
16. Rosenhahn, B., Perwass, C., Sommer, G.: Free-form pose estimation by using twist representations. Algorithmica **38**(1), 91–113 (2003)
17. Sommer, G., Rosenhahn, B., Perwass, C.: Twists—an operational representation of shape. In: IWMM GIAE, pp. 278–297 (2004)
18. Triggs, B., McLauchlan, P., Hartley, R., Fitzgibbon, A.: Bundle adjustment—a modern synthesis, pp. 298–375 (2000)
19. Zhang, Z.: Flexible camera calibration by viewing a plane from unknown orientations, vol. 1, pp. 666–673 (1999)
20. Zhang, Z.: A flexible new technique for camera calibration. IEEE Trans. Pattern Anal. Mach. Intell. **22**(11), 1330–1334 (2000)