HAL
open science

Continuous Multi-image Preprocessing for Euclidean Reconstruction

Philippe Renault, Olivier Faugeras, Thierry Viéville

To cite this version:

Philippe Renault, Olivier Faugeras, Thierry Viéville. Continuous Multi-image Preprocessing for Euclidean Reconstruction. [Research Report] RR-3482, INRIA. 1998. inria-00073206

HAL Id: inria-00073206

https://inria.hal.science/inria-00073206

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Continuous multi-image preprocessing for Euclidean reconstruction

P. Renault and O.D. Faugeras and T. Viéville

N° 3482

Septembre 1998

THÈME 3

Rapport de recherche

ISSN 0249-6399

Continuous multi-image preprocessing for Euclidean reconstruction

P. Renault and O.D. Faugeras and T. Viéville

Thème 3 — Interaction homme-machine,
images, données, connaissances
Projet RobotVis

Rapport de recherche n° 3482 — Septembre 1998 — 44 pages

Abstract: We address the problem of performing 3D motion and structure analysis along long video sequences (mpeg animation, surveillance recording television programs, etc..) in which we want to :

- be able to segment the sequence in smaller pieces of image sets corresponding to homogeneous parts of the sequence,
- stabilize the sequence in order to reduce the disparity between two consecutive frame,
- detect unexpected events such as moving objects or close obstacles,
- reconstruct the 3D shape of some predefined objects, which implies the self calibration of the image sequence.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, B.P. 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : 04 92 38 77 77 - International : +33 4 92 38 77 77 — Fax : 04 92 38 77 65 - International : +33 4 92 38 77 65

Such data have two characteristics : (a) they have been recorded using cameras with unknown and possibly varying optical parameters* and (b) cameras are moving while some objects are also moving in the scene.

It is, in general, not possible to entirely analyze such sequences automatically because of the complexity of the contained information. On the other hand, a complete manual analysis is also not possible because of the huge amount of data. Here, the idea is to pre-process the image sequence, in order to ease its interactive analysis and provide to the user an enhanced image sequence in which:

- the image sequence have been stabilized to ease its observation, so that.
  1. first order approximations of usual motion equations could be numerically valid,
  2. disparity not related to object motion or object depth parallax displacement could be eliminated,
- regions of disparities corresponding to close obstacles or objects in motion have been segmented,i.e. their contour have been extracted and the relative location/size of these objects have been computed.

Key-words: Visual un-calibrated motion, Reconstruction, Stabilization

* We are in the un-calibrated case quoted before.

Prétraitement de séquences continues d'images en vue de la reconstruction euclidienne.

Résumé : On adresse le problème de l'analyse du mouvment 3D et de la structure le long de longues séquences vidéo que l'on souhaite segmenter en volumes spatio-temporels homogènes, stabiliser de façon à réduire la disparité ne contenant pas d'information de structure dans chaque volume identifié et y détecter les évennements tels qu'objets mobiles ou obstacles proches qui nécessiteront un traitement ultérieur spécifique pour finalement reconstruire la forme 3D de certains objets d'intérêts.

Tenant compte du fait que ces données ont été acquises dans le cas non-calibré avec des variations des paramètres intrinsèques de la caméra et des objets mobiles dans la scène, on se propose d'attaquer le problème en mettant au point un module de prétraitement qui rende valide à la fois l'utilisation des équations au premier ordre du mouvement que l'on revisite ici pour en montrer tout l'intérêt et qui prépare automatiquement le flot de données à des traitement ultérieurs, manuels ou interactifs, sur un ensemble réduit et prétraité de données.

Mots-clés : Mouvement visuel non-calibré, Reconstruction, Stabilisation

1 Introduction

Video sequences preprocessing. We address the problem of performing 3D motion and structure analysis along long video sequences (mpgeg animation, surveillance recording television programs, etc...) in which we want to (i) detect unexpected events such as moving objects or close obstacles (ii) reconstruct the 3D shape of some predefined objects, etc...

Such data has two characteristics:

1. they have been recorded using cameras with unknown and possibly varying optical parameters,
2. cameras are moving and some objects are also moving in the scene[1],
3. they are naturally structured in terms of "small" shots of about 10 to 30 seconds (clip, view of a scene, short dialog in an interview, etc...) i.e. 300 to 1000 frames,

our goal being to analyze such a spatio-temporal volume of data.

It is in general not possible to entirely analyze such sequences automatically because of the complexity of the information they contain. On the other hand, a complete manual analysis is also not possible because of the huge amount of data. Here the idea is to pre-process the image sequence, in order to ease its manual analysis and provide to the user an enhanced image sequence in which :

- the image sequence has been stabilized to ease its observation,
- regions of disparities corresponding to close obstacles or objects in motion have been segmented,
- the relative location/size of these objects have been computed.

In order to be usable such a preprocessing module requires to be *fast* and *robust*.

[1]Usual inter-frame average disparity is about 10 pixels.

Using first-order motion equations. In order to attain these objectives, following [23], we revisit the problem of retinal motion parameterization in an image sequence without calibration in order to attain the previous objective.

At a theoretic level and contrary to these studies, in the present paper, we show that considering derivatives instead of first-order expansions leads to much simpler equations and allows a better understanding of the relation between the retinal motion field and the 3D parameters. The reason of this nice situation is that we obtain equations very similar to those used in the calibrated case [4, 27].

Furthermore, other contributions in the field [3, 13] consider a model based on the retinal projection matrix and leads to very heavy expressions, yet not fully implemented. Here bundle adjustment [4] and frame-to-frame parameterization [24] is preferred. This allows a much simpler implementation of the related motion module.

Notations: We write vectors and matrices in bold letters, matrices being written with capital letters. The duals of vectors are represented as the transpose of a vector and scalars in italic. The notation $\mathbf{x} \wedge \mathbf{y} = [\mathbf{x}]_\wedge \mathbf{y}$ corresponds to the cross-product, the dot-product being written as $\mathbf{x}^T\mathbf{y}$. The matrix $[\mathbf{x}]_\wedge$ is a 3×3 skew-symmetric matrix. The identity matrix is written $\mathbf{I}$. Geometric objects such as points, lines, planes are written with capital letters in 3D, and small letters in 2D. We represent the components of a matrix or a vector using superscripts from 0 to 2, e.g.: $\mathbf{x} = (x^0, x^1, x^2)^T$. The notation $\mathbf{x} \equiv \mathbf{y}$ means $\exists \lambda \neq 0, \lambda \mathbf{x} = \mathbf{y}$. We use $\mathbf{z} = (0, 0, 1)^T$

1.1 Before starting: a short note on basic concepts

Let us first explain some of the concepts and definitions used in the developments. This might help you getting familiar with formal definitions.

The plane at infinity

This plane at infinity is a virtual plane define by all the points which have an infinite depth, in the context of the chosen model. These points also correspond to vanishing points (or "intersection of parallel lines ") or directions of lines. Therefore these points are not affected by translations but only by rotations.

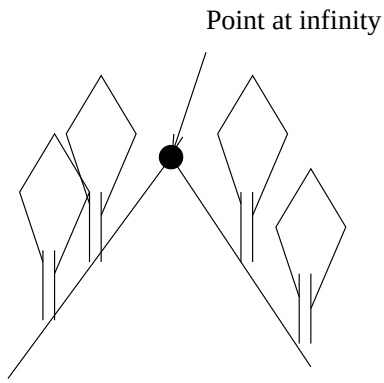

Figure 1: Point of the plane at infinity

In some applications with outdoors scenes this plane corresponds to the plane at horizon. Moreover since we try to detect planar structure, the plane at infinity will be detected as a particular plane. However, without any "semantic" knowledge on the scene, there is no way to detect which plane is the plane at infinity. For example, during the stabilization process, we consider the whole image as an average planar structure. If the general displacement is a pure rotation this approximation is exact (i.e. all points are at infinity), else this "average" plane might approximate the plane at infinity, or provide an initial estimate of it.

Sparse or dense

There are two ways to analyze an image, either by detecting specific points and perform the analysis on these points, or by using the whole image intensity map. We should call these two approaches the sparse and dense one. This is a spatial cue which should not be confused with the problem of the continuous and discrete temporal models.

Continuous and Discrete

In our case, we will NOT use the discrete time model based on correspondences between image tokens in two views, but always use the continuous one[2], us-

[2]Anyway there is no real "continuity" in video image sequences because the disparity between two frames at 25Hz might be of the order of magnitude of 10 pixels. Therefore our stabilization process is mandatory to reduce this disparity to 1-2 pixels so that the derivative operators could be well-defined.

ing velocities related to image intensity variation as measurements. These will be computed from token correspondences by discrete approximations of derivatives operators.

Parameters tuning
A classical issue with such modules is the problem of parameters adjustment. We often see softwares with extraordinary results, but working on only one video sequence! They also must be adapted to another image sequence by changing parameters of the program. Now we want to analyze if there is a way to deduce from the sequence the parameters to be used or if the help of an expert system is required. During all our developments we will take care of this problem and will clarify all the parameters either fixed or set by the programmer or left to the user. A discussion on how to adjust them will be issued.

2 The continuous approach revisited.

2.1 From first order expansion to derivatives.

Camera model and rigid motion .

Since, in this study, *we do not assume the system is calibrated*, we use the following, now well-known (see for instance [8] for a review), projection model :

$$Z\,\mathbf{m} = \mathbf{A}\,\mathbf{M}\ \textit{with} \left\{ \begin{array}{rcl} \mathbf{m} & = & (u, v, 1)^T \\ \mathbf{A} & = & \begin{pmatrix} \alpha_u & -\alpha_v\,\cot(\theta) & u_0 \\ 0 & \alpha_v/\sin(\theta) & v_0 \\ 0 & 0 & 1 \end{pmatrix} \\ \mathbf{M} & = & (X, Y, Z)^T \end{array} \right. \tag{1}$$

where

- $\mathbf{m}$ is the vector of the coordinates of the projected points on the image plane

- **M** is the vector of the coordinates of the real point in a frame of reference attached to the retina

- (u_0, v_0) is the projection of the optical center, i.e. the principal point

- (α_u, α_v) are the horizontal and vertical scale factors, i.e. the size of one pixel on the camera screen.

- θ is the complementary of the angle of distortion of the pixel (or of the optical axes...).

The ration (α_u/α_v) is often known and constant [22], while θ is often equal to $\pi/2$[7] and corresponds to a skew-factor. Therefore the matrix **A** of the **intrinsic parameters** of the camera is defined in practice by three parameters[25], although at a theoretical level it can have up to 5 independent parameters[18].
The quantity Z is the depth of the point along the optical axis. We can also consider A as a 3*4 matrix and add a coordinate to M, $M = (X, Y, Z, U)$ where $U = 1$ for affine points and $U = 0$ for points at infinity. All this can be sum up on a small graph:

More precisely, here, (u, v) are the pixel coordinates of the 2D projection of the 3D point **M** which *homogeneous* coordinates are: $(X, Y, Z, \mathcal{U})^T$. We choose either (a) $\mathcal{U} = 1$ for affine points or (b) $\mathcal{U} = 0$ for projective points at infinity, i.e. 3D line directions or "vanishing points"; our equations will be valid in both cases. It is important to note that M is taken in a *frame of reference attached to the retina* with the Z-axis aligned with the optical axis. As one consequence, Z is the depth of the point along the optical axis.

Since we consider a *rigid motion*, we can write :

$$\dot{\mathbf{M}} = \mathbf{V} + \mathbf{\Omega} \wedge \mathbf{M} \tag{2}$$

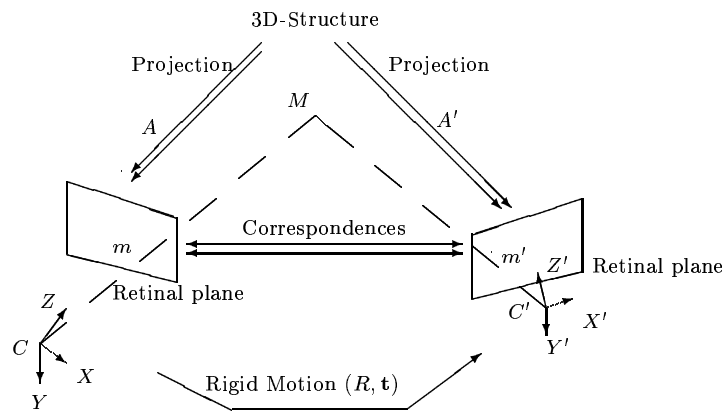

Figure 2: Elements used in the definition of the problem.

where $\mathbf{V}$ is the translational velocity at the optical center and $\mathbf{\Omega}$ the rotational velocity of the rigid object, $\mathbf{M}$ belongs to. For points at infinity, we simply have $\dot{\mathbf{M}} = \mathbf{\Omega} \wedge \mathbf{M}$, i.e. we can consider $\mathbf{V} = 0$.

Derivation of the "3D retinal-motion". Now combining the previous equations we obtain [3]:

$$
\begin{array}{rcll}
Z\,\mathbf{m} & = & \mathbf{A}\,\mathbf{M} & \textit{from } (1) \\
\dot{Z}\,\mathbf{m} + Z\,\dot{\mathbf{m}} & = & \dot{\mathbf{A}}\,\mathbf{M} + \mathbf{A}\,\dot{\mathbf{M}} & \textit{by differenciation} \\
\dot{Z}\,\mathbf{m} + Z\,\dot{\mathbf{m}} & = & \dot{\mathbf{A}}\,\mathbf{M} + \mathbf{A}\,[\mathbf{V} + \mathbf{\Omega} \wedge \mathbf{M}] & \textit{using } (2) \\
\dot{Z}\,\mathbf{m} + Z\,\dot{\mathbf{m}} & = & \dot{\mathbf{A}}\,Z\,\mathbf{A}^{-1}\,\mathbf{m} + \mathbf{A}\,[\mathbf{V} + \mathbf{\Omega} \wedge Z\,(\mathbf{A}^{-1}\,\mathbf{m})] & \textit{using } (1) \\
\dot{Z}\,\mathbf{m} + Z\,\dot{\mathbf{m}} & = & Z\,\underbrace{\dot{\mathbf{A}}\,\mathbf{A}^{-1}}_{\mathbf{B}}\,\mathbf{m} + \underbrace{\mathbf{A}\,\mathbf{V}}_{\mathbf{v}} + Z\,\underbrace{\mathbf{A}\,\mathbf{A}^{T}}_{\mathbf{K}}\,(\underbrace{\mathbf{A}\mathbf{\Omega}/det(\mathbf{A})}_{\omega}) \wedge \mathbf{m} & \textit{since } \begin{array}{l}[M\mathbf{x}] \wedge [\mathbf{M}\mathbf{y}] = \\ det(\mathbf{M})\,\mathbf{M}^{-1\,T}\,[\mathbf{x} \wedge \mathbf{y}]\end{array} \\
\dot{Z}\,\mathbf{m} + Z\,\dot{\mathbf{m}} & = & Z\,\underbrace{(\mathbf{B} + \mathbf{K}\,[\omega]_{\wedge})}_{\mathbf{C}}\,\mathbf{m} + \mathbf{v} & \textit{after factorization} \\
\textit{finally}: & & \boxed{\dot{\mu} = \dot{\mathbf{m}} + \frac{\dot{Z}}{Z}\,\mathbf{m} = \mathbf{C}\,\mathbf{m} + \frac{1}{Z}\,\mathbf{v}} &
\end{array}
\tag{3}
$$

[3] This first straightforward derivation is given in details. Other derivations of this kind will be left to the reader in the sequel.

so that if we define:

$$\left\{\begin{array}{lcll} \mathbf{v} & = & \mathbf{A}\,\mathbf{V} & \textit{the retinal projection of } \mathbf{V} \textit{ or "focus of expansion"}, \\ \omega & = & \mathbf{A}\mathbf{\Omega}/det(\mathbf{A}) & \textit{the normalized retinal projection of } \mathbf{\Omega}, \\ \mathbf{K} & = & \mathbf{A}\,\mathbf{A}^T & \textit{which is in one-to-one correspondance with } \mathbf{A}\ [18,\ 24], \\ \mathbf{B} & = & \dot{\mathbf{A}}\,\mathbf{A}^{-1} & \textit{for which we easily get}: \dot{\mathbf{K}} = \mathbf{B}\,\mathbf{K} + \mathbf{K}\,\mathbf{B}^T \textit{ and} \\ \mathbf{C} & = & \mathbf{B} + \mathbf{K}\ [\omega]_\wedge & \textit{called the "stabilization matrix"} \end{array}\right. \tag{4}$$

we obtain the previous equation which characterizes the "2D retinal-motion" $\dot{\mathbf{m}} = (\dot{u}, \dot{v}, 0)$. We also define the **"3D retinal-motion"** $\dot{\mu}$.

From 3D to 2D retinal-motion. A step further, since by definition, with $\mathbf{z} = (0,0,1)^T$, $\mathbf{m}^T\,\mathbf{z} = 1$ and $\dot{\mathbf{m}}^T\,\mathbf{z} = 0$, from (3), when multiplying by $\mathbf{z}^T$:

$$\dot{Z} = Z\,(\mathbf{z}^T\,\mathbf{C}\,\mathbf{m}) + (\mathbf{z}^T\,\mathbf{v}) \Leftrightarrow \frac{\dot{Z}}{Z} = \mathbf{z}^T\,\dot{\mu} \tag{5}$$

called the "structure evolution" equation.

This last derivation allows to obtain, by elimination of $\dot{Z}$ and dividing by Z in (3), the following fundamental equation:

$$\dot{\mathbf{m}} = \underbrace{\left[\mathbf{I} - \mathbf{m}\,\mathbf{z}^T\right]}_{\mathbf{M}}\,\dot{\mu} \tag{6}$$

The 2D retinal-motion $\dot{\mathbf{m}}$ is thus the projection[4] of the 3D retinal-motion $\dot{\mu}$ onto the retinal plane in the direction of the optical ray.

We also can write the previous equations in a scalar and somehow more readable form, expanding $\dot{\mu}$:

$$\left\{\begin{array}{lcl} \dot{u} & = & (\mathbf{C}^{00} - \mathbf{C}^{22})\,u + \mathbf{C}^{01}\,v + \mathbf{C}^{02} - \mathbf{C}^{20}\,u^2 - \mathbf{C}^{21}\,u\,v + \frac{1}{Z}\,[\mathbf{v}^0 - \mathbf{v}^2\,u] \\ \dot{v} & = & \mathbf{C}^{10}\,u + (\mathbf{C}^{11} - \mathbf{C}^{22})\,v + \mathbf{C}^{12} - \mathbf{C}^{20}\,u\,v - \mathbf{C}^{21}\,v^2 + \frac{1}{Z}\,[\mathbf{v}^1 - \mathbf{v}^2\,v] \end{array}\right.$$
$$\textit{with}$$
$$\dot{Z} = Z\,[\mathbf{C}^{20}\,u + \mathbf{C}^{21}\,v + \mathbf{C}^{22}] + \mathbf{v}^2 \tag{7}$$

[4]The matrix $\mathbf{M}$ in equation (6), has the following properties:

$$\mathbf{M}\,\mathbf{M} = \mathbf{M} \quad \mathbf{M}\,\mathbf{m} = 0 \quad \mathbf{v} \perp \mathbf{z} \Rightarrow \mathbf{M}\,\mathbf{v} = \mathbf{v} \quad \mathbf{M}\,\dot{\mathbf{m}} = \dot{\mathbf{m}}$$

This means that $\mathbf{M}$ is a projection matrix, the projection onto the retinal plane of equation $z = 0$ along the direction of the optical ray $\mathbf{m}$.

Equation invariance. These equations are invariant for the following two transformations:

$$\begin{array}{llll} \textit{Scale} & \textit{factor} & \mathbf{v} \to \mu\,\mathbf{v} & Z \to \mu\, Z \quad \dot{Z} \to \mu\,\dot{Z} \\ \textit{Expansion} & \textit{factor} & \mathbf{C} \to \mathbf{C} + \lambda\,\mathbf{I} & \dot{Z} \to \dot{Z} + \lambda\, Z \end{array} \tag{8}$$

The *scale factor* problem is well-known for monocular image sequences (again [8] gives a review).

During numerical estimations, we will assume, in the first frame, either $v^0 = 1$ or $v^1 = 1$ or $v^2 = 1$ or $v^0 = v^1 = v^2 = 0$ to avoid this indetermination.

The *expansion factor* is also known as an indetermination in monocular image perception, in the absence of calibration. It corresponds to the fact that a "zoom" can not be distinguished from a world expansion, i.e. that we must take similarities into account, not only rigid displacements [17]. This fact has been analyzed in biological vision [5, 6] and parameterized, in the discrete case, for computer vision [24].

During numerical estimations, this indetermination will be avoided, assuming $\mathbf{C}^{22} = 0$, since it does neither change the form of the equations nor its numerical conditioning.

A reasoning identical to [24] allows us to state that these are the only two such transformations which leave (7) invariant.

2.2 Comparison with other parameterizations of the retinal motion.

The calibrated case. In the calibrated case, the motion equation has indeed the same form (see for instance [8]) with $\mathbf{C} = [\mathbf{\Omega}]_\wedge$ and $\mathbf{v} = \mathbf{V}$. The suitable notations, used here, allows to deal with similar formulas in both cases. The notion of 3D retinal-motion is also similar to the calibrated case [6].

Parameterization in the discrete case. The present approach corresponds to the so called "Q-s" decomposition[5] of un-calibrated motion [24] with

[5]Given two frames $\mathcal{R}$ and $\mathcal{R}'$ the Qs-equation is: $\frac{Z'}{Z}\,\mathbf{m}' = \mathbf{Q}\,\mathbf{m} + \frac{1}{Z}\,\mathbf{s}$ with $\mathbf{Q} = \mathbf{A}'\,\mathbf{R}\,\mathbf{A}^{-1}$ and $\mathbf{s} = A'\,\mathbf{t}$.

the following correspondences, shown in [23]:

$$\begin{array}{rcrcl} \mathbf{F} & \equiv & [\mathbf{s}]_\wedge \, \mathbf{Q} & = & d\tau \, [\mathbf{v}]_\wedge + d\tau^2 \, [\mathbf{v}]_\wedge \, \mathbf{C} + o(d\tau^3) \\ \mathbf{e} & \equiv & \mathbf{s} & = & d\tau \, \mathbf{v} + o(d\tau^2) \\ \mathbf{H}_\infty & \equiv & \mathbf{Q} & = & \mathbf{I} + d\tau \, \mathbf{C} + o(d\tau^2) \end{array} \tag{9}$$

where $\mathbf{F}$ is the Fundamental matrix, $\mathbf{e}$ the epipole and $\mathbf{H}_\infty$ the collineation of the plane at infinity.

From a numerical point of view, the gain of the continuous approach is the linearity and a better conditioning with respect to the motion parameters (here $\mathbf{C}$ and $\mathbf{v}$). The quantities are expressed directly in pixel, whereas this was not true in the discrete case[6] [24].

Parameterization from the projection matrices. Furthermore, in the discrete case and using more than two views [10, 16] considering the projection matrix:

$$\bar{\mathbf{P}} = \left[\underbrace{\mathbf{A}\,\mathbf{R}}_{\mathbf{P}} \,|\, \underbrace{\mathbf{A}\,\mathbf{t}}_{\mathbf{p}} \right]$$

instead of the Qs-representation leads to a simpler representation.

This is not true in the continuous case, since the corresponding approach requires the construction of rather heavy algebraic objects [13] (see appendix B for details).

In order to get a step further, let us now analyze in details the planar case.

2.3 Analyzing the planar case.

We assume in this sub-section that all points $\mathbf{M}$ belongs to a plane P of equation:

$$d(M, P) = d - \mathbf{N}^T \, \mathbf{M} = 0 \quad \textit{with} \quad d > 0 \quad \textit{and} \quad ||\mathbf{N}|| = 1 \tag{10}$$

where, as detailed for instance in [26], $\mathbf{N}$ is the plane normal and $d = d(P, O) \geq 0$ its distance to the origin and $d(M, P)$ the Euclidean distance of M to the plane p.

[6] In the discrete case, one could either write a sub-optimal linear criterion or an optimal non-linear criterion.

Introducing (1) in (10) we can write $\mathbf{n}^T\,\mathbf{m} = \frac{1}{Z}$ where we define $\mathbf{n} = \frac{\mathbf{A}^{-1\,T}\,\mathbf{N}}{d}$ which is *the projection of the plane vector* $\mathbf{N}/d$. Planes which contain the optical center are excluded.

This allows to eliminate $\frac{1}{Z}$ in (6) and if we define:

$$\mathbf{C}_P = \mathbf{C} + \mathbf{v}\,\mathbf{n}^T + \xi\,\mathbf{I} \tag{11}$$

where ξ is an undefined scale factor, we easily simplify (6) as $\dot{\mu} = \mathbf{C}_P\,\mathbf{m}$ or :

$$\left\{\begin{array}{rcl} \dot{u} & = & (\mathbf{C}_P^{00} - \mathbf{C}_P^{22})\,u + \mathbf{C}_P^{01}\,v + \mathbf{C}_P^{02} - \mathbf{C}_P^{20}\,u^2 - \mathbf{C}_P^{21}\,u\,v \\ \dot{v} & = & \mathbf{C}_P^{10}\,u + (\mathbf{C}_P^{11} - \mathbf{C}_P^{22})\,v + \mathbf{C}_P^{12} - \mathbf{C}_P^{20}\,u\,v - \mathbf{C}_P^{21}\,v^2 \end{array}\right. \tag{12}$$

Analyzing the equations. This is a quadratic field, as in the calibrated case [27]. This equation is also the same as (6) but with $\mathbf{v} \to 0$ and $\mathbf{C} \to \mathbf{C}_P$. As previously, numerical estimations are done with $\mathbf{C}_P^{22} = 0$ to avoid undetermination.

The $\mathbf{C}_P$ matrices correspond to collineations in the discrete case, as for $\mathbf{H}_\infty$ in (9) :

$$\mathbf{H}_P \equiv \mathbf{I} + d\tau\,\mathbf{C}_P + o(d\tau^2)$$

In coherence with the fact that $\mathbf{H}_P$ is defined up to a scale factor $\mathbf{C}_P$ is defined up to a factor of the identity, as formalized in (11).

This formalization includes the case of the "horizon" or plane at infinity P_∞, equivalent to a pure rotation. From an algebraic point of view, if either $\mathbf{V} = \mathbf{v} = 0$, or considering P_∞ (i.e. $\mathbf{n} = 0$), we have $\mathbf{C}_{P_\infty} = \mathbf{C}$.

Evolution from one frame to another. A step further we can easily calculate the evolution of the projection of the planar vector, and a calculus very similar to the previous ones (and left to the reader) gives :

$$\dot{\mathbf{n}} = -\mathbf{C}_P^T\,\mathbf{n} \tag{13}$$

while we had $\dot{\mathbf{N}} = \mathbf{N} \wedge \mathbf{\Omega}$ and $\dot{d} = \mathbf{N}^T\,\mathbf{V}$ for the Euclidean plane geometric elements.

Computing the focus of expansion. Considering a point m, not necessarily projection of a 3D point in the plane P we can easily derived with $\dot{\mathbf{m}} = \mathbf{M}\,\mu$:

$$\dot{\mu} = \mathbf{C}_P\,\mathbf{m} + \underbrace{\left[\frac{1}{Z} - (\mathbf{n}^T\,\mathbf{m})\right]}_{\frac{1}{Z_P}}\,\mathbf{v} \tag{14}$$

This means that if the retinal motion of a point not in the plane is approximated by the planar motion the related error is always aligned with the focus of expansion $\mathbf{v}$, more precisely:

$$\mathbf{v}^T\,[\mathbf{m} \wedge (\dot{\mathbf{m}} - \mathbf{M}\,\mathbf{C}_P\,\mathbf{m})] = 0 \tag{15}$$

and the focus of expansion can be estimated as soon as two generic points not in the plane are considered.

Relative location with respect to a plane. Furthermore, from [26] we know that:

$$\frac{1}{Z_P} = \frac{d(M,P)}{d(M,O)\,d(P,O)}$$

which is a measure of "proximity" from a point to a plane, its *sign* determines whether M is behind or in front of the plane, as introduced in [19].

Points "in front of" the plane (i.e. between the plane and the retina) have the same sign as $\left[1 - (\mathbf{n}^T\,\mathbf{m})\right]$ which is the value of $\frac{1}{Z_P}$ for the intersection of the optical ray with the retina.

Detecting moving planar patches. From (11), two matrices $\mathbf{C}_{P_1}$ and $\mathbf{C}_{P_2}$ correspond to the same rigid displacement, with matrix $\mathbf{C}$ and focus of expansion $\mathbf{v}$, if and only if $\mathbf{Z} = [\mathbf{C}_{P_1} - \mathbf{C}_{P_2}]$ *has two eigen-values which are equal*[7] or equivalently:

$$\mathbf{v} \wedge [(\mathbf{C}_{P_1} - \mathbf{C}_{P_2})\,\mathbf{v}] = 0 \tag{16}$$

[7]From (11), we have $\mathbf{Z} = \mathbf{v}\,\left[\underbrace{\mathbf{n}_1 - \mathbf{n}_2}_{\mathbf{n}}\right]^T + \underbrace{\xi_1 - \xi_2}_{\xi}\,\mathbf{I}$ for some $\mathbf{n}$ and ξ if and only if they correspond to the same matrix $\mathbf{C}$ and vector $\mathbf{v}$.
Such a matrix $\mathbf{Z}$ is of the form $\mathbf{Z} = \mathbf{v}\,\mathbf{n}^T + \xi\,\mathbf{I}$ for some $\mathbf{n}$ and ξ if and only if two eigen-values

for a given $\mathbf{v}$.

This provides an algebraic condition to verify if (i) two rigid planar displacements to correspond to the same rigid motion or (ii) two rigid planar displacements to be compatible with a given focus of expansion.

2.4 Considering line tokens.

We can also introduce *line tokens* in the estimation of planar or rigid structures as in the discrete case [24].

Here, the notion of line-token d of equation :

$$\mathbf{m} \in d \Leftrightarrow \mathbf{m}^T\,\mathbf{d} = 0$$

is integrated in our formalism, by introducing the idea of sliding-points as in [24] or [26]:

$$i \in \{1,2\}, \mathbf{m}_i = \mathbf{m}_\bullet + \lambda_i\,\underbrace{\mathbf{z} \wedge \mathbf{d}}_{\delta} \in d \quad for\ some\ \lambda_i \tag{17}$$

where λ_i is an undefined, fixed quantity and $\mathbf{m}_\bullet$ is an arbitrary point on the line.

Two sliding-points define the 2D-line since $\mathbf{m}_1 \wedge \mathbf{m}_2 = (\lambda_1 - \lambda_2)\,\mathbf{d}$, while each $\mathbf{m}_i$ is defined in the $\mathbf{d}$ direction only. Reciprocally, $\dot{\mathbf{m}}_i$ is defined in the direction of $\dot{\mathbf{d}}$ only, as easily established by derivation of (17).

Furthermore, for a line-segment, sliding points positions in the direction of the line token are bounded by the line segment length, thus providing another approximate measure.

The statistical framework defined in the next section will allow to implement this idea.

---

are equal because on one hand, it is easy to see that:

$$\begin{array}{rcrcl} \mathbf{x} \perp \mathbf{v} & \Rightarrow & \mathbf{Z}\,\mathbf{v} & = & \xi\,\mathbf{v} \\ \mathbf{x} || \mathbf{n} & \Rightarrow & \mathbf{Z}\,\mathbf{n} & = & (\xi + \mathbf{v}^T\,\mathbf{n})\,\mathbf{v} \end{array}$$

so that $\mathbf{Z}$ has indeed two eigen-values which are equal, while on the other hand, if a matrix $\mathbf{Z}$ has two eigen-values ξ which are equal $\mathbf{E} = \mathbf{Z} - \xi\,\mathbf{I}$ is of rank 1 since for any vector $\mathbf{x}$ in the corresponding two dimensional eigen-space $\mathbf{E}\,\mathbf{x} = 0$ and is thus of the form $\mathbf{E} = \mathbf{a}\,\mathbf{b}^T$.

Finally to eliminate $\mathbf{n}$ we must compute $[\mathbf{v}]_x\,Z = \xi\,[\mathbf{v}]_x$ and to eliminate ξ: $[\mathbf{v}]_x\,Z\,\mathbf{v} = 0$.

3 Motion equation applied to image points.

Using corner detection and local correlation. Following most implementations [28, 24, 23] we can detect points at which the motion-field is defined in two directions, i.e. "corners" (here using [20]), whereas along edges the retinal motion-field is in general defined in the edge normal direction only (known as the aperture problem) while on points not on edges the retinal motion-field is ill-defined (see for instance [21] for a discussion). Since we are in an image sequence, we can easily match each corner from one image to another using a simple correlation operator [24, 23] and consider only the corners tracked during the whole sequence. This will provides us with a local estimation of the retinal motion and the "flatness" of the correlation function around the match allows to have an estimation of its uncertainty, as for stereo-scopic perception [9].

Implementing the motion computation. Let us:

- Detect corners in the image and track them from one frame to another. The corner detection module has been upgraded and now uses both spatial, intensity, and intensity gradient proximity to match corners. To enhance future computation we therefore consider only the corners successfully matched during the whole sequence.

- Compute parameterized motion models as follows :

  1. *Image stabilization.* We can consider all the points and look for the $\mathbf{C}_P$ matrix which minimizes their average disparity. By doing so we do not estimate a collineation corresponding to a real planar structure but an image transformation which minimizes the residual disparity between two consecutive images.

  2. *Planar structure detection.* We randomly select subsets of points and choose the $\mathbf{C}_P$ matrix which canceled at most, the residual disparity for those points. From this estimation we segment all points in two classes whether their retinal motion is compatible or not with the $\mathbf{C}_P$ matrix (outliers rejection). Among all trials we

select the best one, i.e. the trial which statistical likehood is the highest.

Repeating this several times allows us to identify the different planar structures of the scene.

3. *Rotational motion detection.* If some points are at the horizon, one of these "planar" displacements correspond to the rotational part of the motion i.e. the planar displacement of the horizon of plane at infinity (i.e. $\mathbf{C}=\mathbf{C}_P$). From (14) we can decide if others points are behind or in front of a given plane. Indeed, no points are "behind" the horizon. Therefore, candidates for horizon planes are those for which no points are behind them, i.e. κ in (14) has always the same sign, which can be statistically tested.

   Additionally, for natural scenes we can assume that points at the horizon are in the upper part of the image etc...

4. *Rigid modelization.* This initial estimate of $\mathbf{C}_P$ used as an initial value in the first order motion equation allows to estimate with an iterative algorithm $\mathbf{v}$, then C, then $\mathbf{v}$ ... We hope this algorithm to converge and give us the real $\mathbf{C}$ matrix, or more exactly $\mathbf{C}' = \mathbf{C} + \mathbf{v}\mathbf{h}^{\mathbf{T}}$ with a given $\mathbf{h}$. To compute $\mathbf{v}$ we use the first order motion equation (see Annex A):

$$(\mathbf{v} \wedge \mathbf{m})^T \, \dot{\mathbf{m}} - (\mathbf{v} \wedge \mathbf{m})^T \, \mathbf{C} \, \mathbf{m} = 0$$

   So if we call $\mu_i = (\dot{m}_i - Cm_i) \wedge m_i$ we have to minimize $\mathcal{L} = \sum_i \|v^T \mu_i\|^2 = v^T \sum_i \mu_i \mu_i^T v$. We therefore have a new way to compute $\mathbf{v}$, as the unitary eigen vector associated with the smallest eigen value of this quadratic form.

5. *Depth computation.* Once the $\mathbf{C}'$ matrix and $\mathbf{v}$ vector has been computed, we can from (14) compute the depth 1/Z for each corner $\frac{1}{Z_i} = \frac{\|\dot{m}_i - \mathbf{M_i C m_i}\|}{\|\mathbf{M_i v}\|} . Signe\,((\dot{m}_i - \mathbf{M_i C m_i}).(\mathbf{M_i v}))$. As we had $\mathbf{C}'$ instead of C, we obtain $\frac{1}{Z_i'} = \frac{1}{Z_i} + \mathbf{h^T m_i}$ instead of $\frac{1}{Z_i}$. The $\mathbf{h}$ parameter can be fixed by the user to obtain the more realistic depth map.

6. *Auto-calibration.*

If you look back at equation (4) you will see that we have

$$C = B + K[w]_\wedge + vh^T + \lambda I$$

with

$$K = \begin{pmatrix} f^2 + u_0^2 & u_0 v_0 & u_0 \\ u_0 v_0 & f^2 + v_0^2 & v_0 \\ u_0 & v_0 & 1 \end{pmatrix} = m_0 m_0^T + f^2[xx^t + yy^T]$$

$$A = f[xx^T + yy^T + m_0 z^T$$

$$B = \begin{pmatrix} \frac{\dot{f}}{f} & 0 & \dot{u}_0 - \frac{\dot{f}}{f} u_0 \\ 0 & \frac{dotf}{f} & \dot{v}_0 - \frac{\dot{f}}{f} v_0 \\ 0 & 0 & 0 \end{pmatrix} = [\dot{m}_0 z^T] + \frac{\dot{f}}{f}[I - m_0 z^T]$$

This is the general case that can't be solved, but we can study some cases where it is possible to extract or to track the camera intrinsic parameters. We will make the following hypothesis : $\frac{\alpha_u}{\alpha_v} = f \ et \ \theta = \frac{\pi}{2}$. We now have only three parameters to compute, in the following cases :

(a) Fixed and known principal point: u_0, v_0
(b) Pure rotation, and u_0, v_0, f fixed: $\mathbf{C} = \mathbf{K}[\mathbf{w}]_\wedge + \lambda \mathbf{I}$
(c) Pure Translation: $\mathbf{C} = \mathbf{B} + \mathbf{vh^T} + \lambda \mathbf{I}$
(d) Pure Rotation and u_0, v_0, f variable. $\mathbf{C} = \mathbf{B} + \mathbf{K}[\mathbf{w}]_\wedge + \lambda \mathbf{I}$
(e) General movement, fixed parameters: $\mathbf{C} = \mathbf{K}[\mathbf{w}]_\wedge + \mathbf{vh^T} + \lambda \mathbf{I}$

The case 3 is a well-known case where we can't compute the parameters but we can can compute their variation. So if we have an initial estimation of these parameters we can track them during the whole sequence. The case 2 is a system where the number of equations is the same as the number of unknown variable so it can be solved , considering some slight variable substitution, by a maple program.

4 Motion equation at the intensity level.

It is well-known that we have two ways of considering the problem:

1. either use a sparse method in which:
   - we detect "corners", i.e. points with high curvature, at which the motion field is defined in two directions,
   - we track these corners. As two consecutive images are similar, we have good chances to match each corner from one image to another using for instance a correlation operators,
   - we implement the different equations reviewed before only on these corners. This means that the stabilization and planar structure detection will be based only on data obtained on these points. This is valid only if they correctly characterize the image,
2. or add a bit more equations and use the information given on the whole image. Obviously what we only have at first is the intensity map. We thus must derive an equation relating the variation of the intensity at each point to the point motion in the retinal plane.

Using a multi-scale dense motion map. Here we consider each image point $\mathbf{m}_i$ and try to compute its corresponding retinal motion $\dot{\mathbf{m}}_i$ with a given information matrix Λ_i^{-1} (the inverse of a covariance) eventually zero if the retinal motion is not defined at this point.

We consider an image sequence as "spatio-temporal" volume without any "past" and "future" but want to make the best out the whole data volume.

Finally we implement a multi-scale approach i.e. consider computations in low resolution images first and then use it as initial values for a finer computation at a higher resolution. Lower resolutions are used because the residual disparity between two frames is inversely proportional to the resolution.

All these can be summarized by considering the vectorial indices i as a 4-dimensional vector :

$$i = (\underbrace{(u,v)}_{\text{Image Location}}, \underbrace{t}_{\text{Image Number}}, \underbrace{s}_{\text{Image Resolution}})$$

4.1 Using the Verri-Pogio operator.

A biased operator for motion computation. A usual assumption for retinal-motion estimation is that from one frame to another the intensity is constant. This is wrong except for highly contrasted points or for Lambertian surfaces in pure rotation [21]. In other situations a bias is introduced, but the magnitude of this bias can be estimated.

More precisely, we can compute the retinal motion field from the intensity derivatives using the rather efficient, Verri-Torre formula [12]:

$$\underbrace{\begin{pmatrix} \frac{\partial^2 \mathcal{I}}{\partial x^2} & \frac{\partial^2 \mathcal{I}}{\partial x\, \partial y} \\ \frac{\partial^2 \mathcal{I}}{\partial x\, \partial y} & \frac{\partial^2 \mathcal{I}}{\partial y^2} \end{pmatrix}}_{\mathbf{H}_i} \underbrace{\begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix}}_{\dot{\mathbf{m}}_i} + \underbrace{\begin{pmatrix} \frac{\partial^2 \mathcal{I}}{\partial x\, \partial t} \\ \frac{\partial^2 \mathcal{I}}{\partial y\, \partial t} \end{pmatrix}}_{\dot{\mathbf{g}}_i} = \epsilon_i \simeq 0 \tag{18}$$

where ϵ_i is bias estimated in [21].

This bias ϵ_i is known to be proportional to the magnitude of the image contrast and isotropic [21]. This can be formalized, using covariances, as follows:

$$\Lambda_{\epsilon_i}^{-1} \equiv ||\mathbf{g}_i||^2\, \mathbf{I} \tag{19}$$

Computation of the operator covariance. Considering a quantification noise of σ_I (typically 2 to 4 for 0-255 height bits monochrome images), we can easily neglect points with : $||\mathbf{g}_i||^2 < \sigma_I^2$ since they do not correspond to points with sufficient contrast[8].

The information matrix can then be easily computed, up to the first order for points with a non-negligible information, from (18) and (19):

$$\Lambda_i^{-1} = \begin{cases} k\, ||\mathbf{g}_i||^2\, \mathbf{H}_i^T\, \mathbf{H}_i & if \quad ||\mathbf{g}_i||^2 > \sigma_I^2 \\ 0 & if \quad ||\mathbf{g}_i||^2 \leq \sigma_I^2 \end{cases} \tag{20}$$

for some undefined constant k.

A step further, the covariance matrix is well-defined at points where: $det(\mathbf{H}_i) = \frac{\partial^2 \mathcal{I}}{\partial x^2}\, \frac{\partial^2 \mathcal{I}}{\partial y^2} - \left[\frac{\partial^2 \mathcal{I}}{\partial x\, \partial y}\right]^2 \neq 0$. Such points correspond to "corners" as explained in [11]

[8] These points corresponds to homogeneous regions of the image, whereas points a with non-negligible gradient magnitude correspond to edges, corners or junctions of the image

($det(\mathbf{H}_i)$ is directly related to corners operators)[9]. Other points were $\mathbf{H}_i$ is of rank 1 correspond to regular edges and still allow to compute the retinal-motion but only in the gradient direction.

Retinal disparity with the Verri-Poggio operator. Let us now consider a retinal-motion parameterized by a "stabilization matrix" $\mathbf{C}$ and a "focus of expansion" $\mathbf{v}$ as defined in the previous section.

Given such a retinal-motion, we define a *stabilized retinal motion*:

$$\dot{\mathbf{s}}_i = \dot{\mathbf{m}}_i - \underbrace{\left[\mathbf{I} - \mathbf{m}_i\,\mathbf{z}^T\right]}_{\mathbf{M}_i} \left[\mathbf{C}\,\mathbf{m}_i + \frac{1}{Z_i}\,\mathbf{v}\right] \tag{21}$$

Ideally if the model is perfect, we could expect $\dot{\mathbf{s}}_i$. In practice this not the case and we must define a statistical function for this quantity.

Knowing the covariances of each retinal motion, we can defined the "likehood" of the stabilized retinal motion, i.e. the Mahalanobis distance [1]. In our case, introducing robust statistics [14], we consider a statistical function $\delta(\Xi)$ which is a monotonic function of Ξ if the data point is plausible and 0 for outliers:

$$\rho(\dot{\mathbf{s}}_i) = \delta(\dot{\mathbf{s}}_i^T\,\Lambda_i^{-1}\,\dot{\mathbf{s}}_i) \tag{22}$$

Considering (18) and (20) we easily derive:

$$\rho(\dot{\mathbf{s}}_i) = \delta\left(k\,||\mathbf{g}_i||^2\,||\dot{\mathbf{g}}_i - \mathbf{H}_i\,\mathbf{M}_i\,\left[\mathbf{C}\,\mathbf{m}_i + \frac{1}{Z_i}\,\mathbf{v}\right]||^2\right) \tag{23}$$

Depth from motion. Given an a priori estimation of $\mathbf{C}$ and $\mathbf{v}$, the optimal estimation of the depth at each point is easily obtained minimizing the following criterion:

$$\rho(\dot{\mathbf{s}}_i) + \mathcal{R} \tag{24}$$

where $\mathcal{R}$ is regularization term and which leads to:

$$k\,||\mathbf{g}_i||^2\rho'(\dot{\mathbf{s}}_i)\,\left[||\mathbf{H}_i\,\mathbf{M}_i\,\mathbf{v}||^2\,\frac{1}{Z_i} - [\mathbf{H}_i\,\mathbf{M}_i\mathbf{v}]^T\,[\mathbf{H}_i\,\mathbf{M}_i\dot{\mathbf{s}}_i]\right] + \frac{\partial\mathcal{R}}{\partial\frac{1}{Z_i}} = 0 \tag{25}$$

[9] Corners are precisely the points were the retinal-motion can be computed in two directions.

well-defined except either (i) if the gradient magnitude vanishes, or (ii) if the retinal location corresponds to focus of expansion or (ii) if the projection of the translation in the direction of the optical ray is aligned with edge orientation.

4.2 Motion estimation as a correlation operator.

Motion estimation. Given the previous estimation of $\frac{1}{Z_i}$, in $\delta(\Xi)$, Ξ is a quadratic function of the motion parameters $\mathbf{v}$ and $\mathbf{C}$ and is thus easily minimized using a global criterion:

$$\mathcal{L} = \biguplus_i \rho\left(\dot{\mathbf{s}}_i\right) + \mathcal{R} \tag{26}$$

where $\mathcal{R}$ is a term of regularization and $\biguplus$ depends on the chosen statistical method: for least-squares this is just a sum, for least-median squares it is the median [14], while in both cases the monotonic function related to $\delta(\Xi)$ is the identity. On the contrary, for M-estimators, $\biguplus$ is still a sum, but the monotonic function related to $\delta(\Xi)$ is chosen differently [14]. Other methods, such as R.A.N.S.A.C. [2], can also be represented that way. See [15] for a review of these methods in our domain.

5 Experimental Results

5.1 Implementation of the algorithm.

1. **Step 1**

   - We consider an off-line application where the image sequences have been saved with a normalized file name : toto.x.pgm or toto.x.gif with x a two digits integer. The algorithm we have implemented manages the format of names and perform the format conversion to pgm if needed.

   - Then we create an array of corners matched from one frame to another. This enables us to evaluate the velocities of a given corner.

- From this data, we run a minimization algorithm realized to determine the C_σ that minimizes the disparity between each pair of consecutive frames. From this stabilization parameters, we can re-interpolate each image, to cancel the disparity modelized by the C_σ matrix. This rectification process outputs a sequence of stabilized images saved to avoid repeating this computation for each action.

The minimization problem :

We consider a set of couples (u_i, v_i) et $(\dot{u}_i, \dot{v}_i)$ corresponding to the corners tracked during the whole sequence. As we are trying to approximate the average global motion of the image by a planar motion, we fit the equation of a planar displacement,i.e.:

$$\begin{cases} \dot{u} = (C_P^{00} - C_P^{22})u + C_P^{01}V + C_P^{02} - C_P^{20}u^2 - C_P^{21}uv \\ \dot{v} = C_P^{10}u + (C_P^{11} - C_P^{22})v + C_P^{12} - C_P^{20}uv - C_P^{21}v^2 \end{cases} \tag{27}$$

Our aim is to try to find a matrix C, that minimizes the least-square average retinal velocity magnitude error for this model:

$$min \sum_i \|\dot{u}_i - ((C_P^{00} - C_P^{22})u_i + C_P^{01}v_i + C_P^{02} - C_P^{20}u_i^2 - C_P^{21}u_iv_i)\|^2$$

$$+\|\dot{v}_i - (C_P^{10}u_i + (C_P^{11} - C_P^{22})v_i + C_P^{12} - C_P^{20}u_iv_i - C_P^{21}v_i^2)\|^2$$

with respect to the eight independent components of the $3*3$ matrix C (remember $C_P^{22} = 0$), rewritten as an eight dimension vector x, and we can write the minimization problem as :

$$min \left(\sum_i \|A_i x - b_i\|^2 \right)$$

with

$$\mathbf{A} = \begin{pmatrix} u & v & 1 & 0 & 0 & 0 & -u^2 & -uv \\ 0 & 0 & 0 & u & v & 1 & -uv & -v^2 \end{pmatrix}$$

$$x = (C_{00}, C_{01}, C_{02}, C_{10}, C_{11}, C_{12}, C_{20}, C_{21}, C_{22})^T$$

$$\mathbf{b} = (\dot{u}_i, \dot{v}_i)^T$$

Since the criterion is quadratic, its solution is given by the normal equation[10].

$$\mathbf{A}^T\mathbf{A}x = \mathbf{A}^T b$$

and the Cholesky algorithm solves the equation in our case since $A^T A$ is definite and positive. Here we write $C = T^T T$ with T triangular inferior. And we then only have to solve the two triangular systems $\begin{cases} T^T y = b \\ Tx = y \end{cases}$.

2. **Step 2**
Then the stabilization process launches a second routine which computes a certain number of data :

   - we introduce the value of C_σ in the fundamental motion equation.

$$(v \wedge m_i)^T \dot{m}_i = (v \wedge m_i)^T C_\sigma m_i \tag{28}$$

This equation is obtained by multiplying (3) by $(v \wedge m)^T$ to get rid of Z and is linear in C and bilinear in v. What we hope is that this initial value given by an average plane motion will lead us to the real C matrix. For that we compute with a least square algorithm the value of v, and then a new value of C,etc...To compute **v** we use the first order motion equation below so if we call $\mu_i = (\dot{m}_i - Cm_i) \wedge m_i$

[10] **Quadratic minimization**

$$\begin{aligned} \mathcal{L} = \|\mathbf{A}x - b\|^2 &= (\mathbf{A}x - b)^T(\mathbf{A}x - b) \\ &= x^T\mathbf{A}^T\mathbf{A}x - 2b^T\mathbf{A}x + b^T b \end{aligned}$$

The derivative is therefore :

$$\mathcal{L}' = 2x^T\mathbf{A}^T\mathbf{A} - 2b^T\mathbf{A}$$

and the result of the minimization problem is the points where $\mathcal{L}' = 0$ so the vector x solution of :

$$\mathbf{A}^T\mathbf{A}x = \mathbf{A}^T b$$

This last problem is solved with a Cholesky algorithm for instance.

we have to minimize $\mathcal{L} = \sum_{\rangle} \|\sqsubseteq^{\mathcal{T}} \mu_{\rangle}\|^{\in} = \sqsubseteq^{\mathcal{T}} \sum_{\rangle} \mu_{\rangle} \mu_{\rangle}^{\mathcal{T}} \sqsubseteq$. We therefore have a new way to compute $\mathbf{v}$, as the unitary eigen vector associated with the smallest eigen value of this quadratic form. In the calculation of C we used a new adjustment method, i.e. instead of computing a least square method on the eight parameters of the C matrix, we have managed to express this matrix with five parameters in the actual equation, and this has provided a better

convergence of the algorithm[11]. To study the validity of the result and the convergence of the algorithm, we can compute the least squared average error. We will see the results in the "Results" part.

[11]We are in the case of the fundamental motion equation :

$$(\dot{m} - Cm).(v \wedge m)^T = 0$$

This can be rewritten :

$$|\dot{m}, v, m| = m^T[[v]_\wedge C]_e m$$

where A_e is the even part of matrix A. This relation can be reversed and we obtain :

$$C = -[v]_\wedge S[I + vv^T] + v\alpha^T + \beta I$$

For our iterative algorithm, we see that we can express C in a base with v as a base vector without loosing any information, if we forgot for a while the βI term. Then we have

$$C = v_1 x^T + v_2 y^T + v z^T$$

$$v_1 \perp v, \|v_1\| = 1$$

$$v_2 \perp v, \|v_2\| = 1$$

To obtain the base where C will have the bigger components to avoid round about errors, we choose more precisely $i. = argmin(|v^0|, |v^1|, |v^2|)$ with respect to e^0, e^1, e^2:

$$v_1 = \frac{v \wedge e_{i.}}{\|v \wedge e_{i.}\|}$$

$$v_2 = v \wedge v_1$$

If we express the first relation in this new base :

$$x = \alpha_0 v + \alpha_1 v_1 + \alpha_2 v_2$$

$$y = \beta_0 v + \beta_1 v_1 - \alpha_1 v_2$$

then

$$C_{v,v_1,v_2} = \begin{pmatrix} 0 & \alpha_0 & \beta_0 \\ 0 & \alpha_1 & \beta_1 \\ 0 & \alpha_2 & -\alpha_1 \end{pmatrix} + (Z, 0, 0)$$

Then we have the minimization on the five parameters $\alpha_0, \alpha_1, \alpha_2, \beta_0, \beta_1$.

- Finally this routine compute the depth map. For that we can use (4), that can be written $\dot{m}_i = M_i C m_i + \frac{1}{Z_i} M_i v$. In fact we don't have C but $C + vh^T$. So if we compute $\frac{1}{Z_i}$ with a mean square on $\frac{\|(\dot{m}_i - M_i C m_i)\|}{\|(M_i v)\|}$ we obtain a relative depth map $\frac{1}{Z_i} + (h^T m_i)$. We have chosen to offer to the user a control panel enabling him to make $h = (h_0, h_1, h_2)$ vary.

3. **Step 3**
   The user can now launch a planar structure detection. We know how to compute a planar motion. So what we do is to take subsets of points and check how many points seem to be in the same planar structure. If there are too few of them, this was not a real planar structure and give it up. Algorithmically, the process is nearly the same, with a Cholesky algorithm to compute the C_P matrix of the subset of points, and then an error calculation to detect outliers. The only problem, is a statistical one. How many points should be chosen in the subset, how many iteration should there be to be sure not to miss the main planar structures, what threshold should be chosen for outliers ?

   To solve this problem, we have used standard statistical methods[12].

[12]In image analysis, we often consider a lot of points and we need to make some tests on subsamples to be able to detect some substructures. In all these tests we pick some outliers that don't fit thee constraint we are trying to apply on them, for example wrong matches when matching, or bad locations, or points belonging to moving objects and not to the background. So we have to use statistical estimations. But to be able to determine the different parameters of the tests, the number of points in the subsample, the number of try, the minimum probability to be accepted, we need to use easy statistical ideas.
The case we consider is the case of the detection of planar patches by making some try on subsamples and finding the most likely one, that means the one for which there is the higher number of points belonging to the same plan. To eliminate the outliers, some authors have realized least-median square algorithm. This method estimates the parameters by finding the smallest value for the median of the squared residuals computed for the entire data set. It turns out that this method is very robust to false matches as well as outliers due to bad locations.
A Monte Carlo type technique is used to draw r subsamples of dim(x) point correspondences. For each subsample indexed by J we determine the parameter **x**. For each parameter, we can determine the trimmed-median of the squared residuals $f_m(x)$, denoted by M_J, with respect

4. **Step 4**
   The last way we have explored using sparse model, is the auto-calibration. We have seen that all this work is situated in the un-calibrated case. To enable 3D reconstruction, or any application needing precise spatial information of the real points, we have to know the camera intrinsic parameters.

5.2 Presentation of sample results.

We have implemented these equations in the sparse case and we will now present the results. When the user launches the global stabilization process, he first sees the corners being tracked. here is an example of two sequences of 8 images being tracked and matched.

---

to the whole set of correspondences. We retain the estimate for which M_J is minimal among all r M_J's. The question now is how to determine r ? Because that 's the parameter missing. How many try should we make to be sure with a high probability that we have the best solution. A subsample is good if it consists of $dim(x)$ good correspondences. Assuming that the whole set of correspondences may contain up to a fraction ϵ of outliers, the probability that at least one of the r subsamples is good is given by :

$$P = 1 - [1 - (1 - \epsilon)^{dim(\mathbf{X}}]^r$$

By requiring that P must be near 1, one can determine r for given values of $dim(\mathbf{x})$ and ϵ. We won't go further in the explanation but we can just suggest that, the least-median-square efficiency being poor in the presence of Gaussian Noise, some work can be done to refine the method after this first estimation by carrying out a weighted least-squares procedure. We have

$$\sum_{i=1}^{k} \left(1 - [1 - (1 - \epsilon)^{dim(\mathbf{X}}]^r\right) = 0,9$$

with ϵ proportion of outliers, r number of iteration, n the number of points in the subset, k the number of awaited planes, 0,9 the expected value of success. In our study of planar structure we have $\epsilon = \frac{N-K}{N}$ with N number of corners in the image, K average number of points in a planar structure. For our study, we take k=10, N=150, K=15, n=5 and we find:

$$r = \frac{0,9N^n}{kK^n}$$

and the result is r =900. This can give an idea of the calculation delay.

Figure 3: Run being tracked

Then after a rectification time and various calculations of errors, the user has access to the stabilized run. Here is an example of a run and a stabilized run :

Figure 4: Sample run

Figure 5: Stabilized run

As you can see on the stabilized run the global translation of the camera has been canceled, and the local movement of translation of the white block

is still there. Of course there is a loss of information on the latest images, but the aim of this stabilization is mainly to obtain a C matrix which will be an initial value for further calculations.
Then from this stabilized sequence we can launch the iterative algorithm who computes the real C matrix and the real translation vector. The analysis of the errors gives the following graphs :

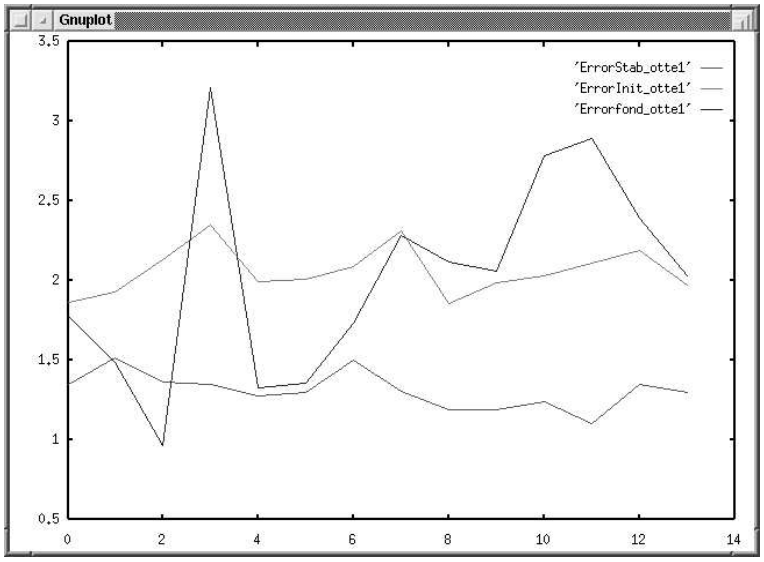

Figure 6: Errors graphs on a 15 images run

The ErrorInit graph is the quadratic disparity of the initial run, $\|\dot{m}\|$. The ErrorStab is the quadratic disparity after stabilization, $\|\dot{m} - \mathbf{MCm}\|$. This error is the result of a minimization algorithm on the initial error, so must be inferior to the previous graph. The ErrorFond is slightly different. it is the result of the convergence of an algorithm using the first order motion equation, so must be close to $\|(\dot{m} - Cm).\frac{(v \wedge m)}{\|v \wedge m\|_{Oxy}}\|$ and should also be inferior to the first graph. This sometimes fail on the example, because the convergence is empirical.
When all this has been realized we then have a depth map computed for each image, giving the theoretical depth of each corners. We have seen in step 2

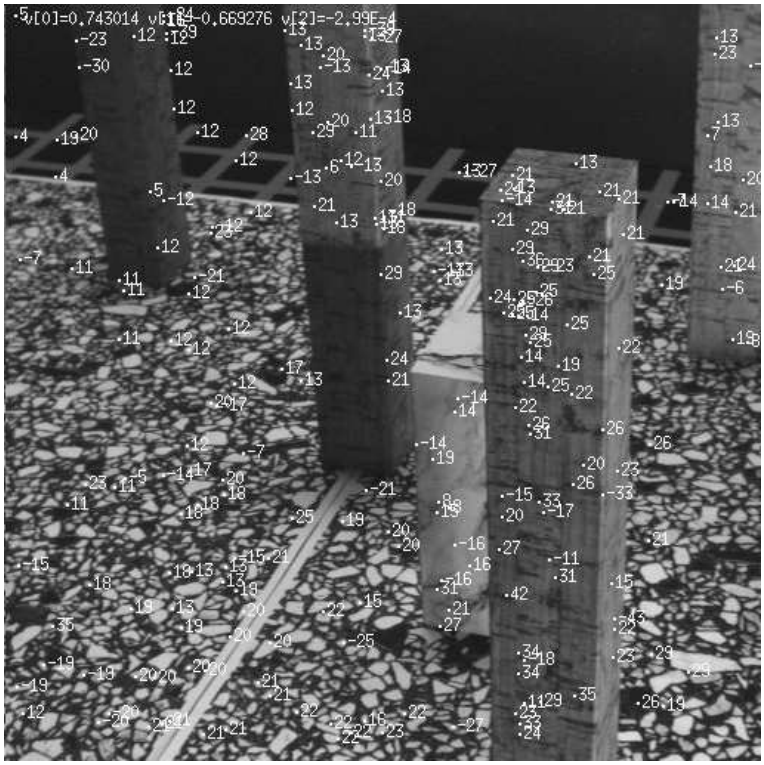

Figure 7: Depth map

of chapter 7.3.1, that this map was parameterized by a vector h, such that $\frac{1}{Z'} = \frac{1}{Z} + (h^T m_i)$. If we make a parameter change $h = d.(cos(\alpha)cos(\beta), sin(\alpha)cos(\beta), sin(\beta))$, the user will act on d, α, β and therefore act on the distance and Eulerian angle of the Stabilization plane to the optical center. This is a more physical way of acting on the parameters.

References

[1] Y. Bar-Shalom and T. Fortmann. *Tracking and Data Association*. Academic, New York, 1988.

[2] R. C. Bolles and M. A. Fischler. A RANSAC-based approach to model fitting and its application to finding cylinders in range data. In *International Joint Conference on Artificial Intelligence*, pages 637–643, Vancouver, Canada, Aug. 1981.

[3] M. Brooks, L. Baumela, and W. Chojnacki. An analytical approach to determining the egomotion of a camera havingfree intrinsic parameters. Technical Report TR 96-04, Dept Comp Sci, Univ. Adelaide, Australia, 1996.

[4] D. Brown. The bundle adjustment - progress and prospect. In *XIII Congress of the ISPRS*, Helsinki, 1976.

[5] V. Cornilleau-Pérès and J. Droulez. The visual perception of 3d shape from self-motion and object-motion. *Vision Research*, 34:2331–2336, 1994.

[6] V. Cornilleau-Pérès, E. Marin, and J. Droulez. The dominance of static depth cues over motion parallax in the perception of surface orientation. *Perception*, 25(40), 1996.

[7] F. Du and M. Brady. Self-calibration of the intrinsic parameters of cameras for active vision systems. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pages 477–482, New-York, NY, June 1993. IEEE Computer Society, IEEE.

[8] O. Faugeras. *Three-Dimensional Computer Vision: a Geometric Viewpoint.* MIT Press, 1993.

[9] O. Faugeras, B. Hotz, H. Mathieu, T. Viéville, Z. Zhang, P. Fua, E. Théron, L. Moll, G. Berry, J. Vuillemin, P. Bertin, and C. Proy. Real time correlation based stereo: algorithm implementations and applications. *The International Journal of Computer Vision*, 1993. To appear.

[10] O. Faugeras and S. Laveau. Representing three-dimensional data as a collection of images and fundamental matrices for image synthesis. In *Proceedings of the International Conference on Pattern Recognition*, pages 689–691, Jerusalem, Israel, Oct. 1994. Computer Society Press.

[11] G. Giraudon and R. Deriche. On Corner and Vertex Detection. In *Computer Vision and Pattern Recognition*, pages 650–655, Lahaina, Maui, Hawai, June 3-6 1991.

[12] F. Girosi, A. Verri, and V. Torre. Constraints for the Computation of Optical Flow. In *Proceedings Workshop on Visual Motion*, pages 116–124. IEEE Computer Society, 1989.

[13] A. Heyden. *Geometry and Algebra of Multiple Projective Transformations.* PhD thesis, Lund University, 1995.

[14] P. Huber. *Robust Statistics.* John Wiley & Sons, New York, 1981.

[15] K. Kanatani. *Geometric computation for machine vision.* Oxford university press, 1992.

[16] S. Laveau. *Géométrie d'un système de N caméras. Théorie, estimation et applications.* PhD thesis, École Polytechnique, May 96.

[17] Q.-T. Luong and T. Viéville. Canonic representations for the geometries of multiple projective views. In J.-O. Eklundh, editor, *Proceedings of the 3rd European Conference on Computer Vision*, volume 1 of *Lecture Notes in Computer Science*, pages 589–599, Stockholm, Sweden, May 1994. Springer-Verlag.

[18] S. J. Maybank and O. D. Faugeras. A theory of self-calibration of a moving camera. *The International Journal of Computer Vision*, 8(2):123–152, Aug. 1992.

[19] L. Robert and O. Faugeras. Relative 3-D positioning and 3-D convex hull computation from a weakly calibrated stereo pair. *Image and Vision Computing*, 13(3):189–197, 1995. also INRIA Technical Report 2349.

[20] S. Smith and J. Brady. Susan - a new approach to low level image processing. *IJCV*, 23(1):45–78, 1997.

[21] A. Verri and T. Poggio. Against quantitative optical flow. In *Proceedings First International Conference on Computer Vision*, pages 171–180. IEEE Computer Society, 1987.

[22] T. Viéville, E. Clergue, R. Enciso, and H. Mathieu. Experimentating with 3-D vision on a robotic head. *Robotics and Autonomous Systems*, 1995. 14(1).

[23] T. Viéville and O. Faugeras. The first order expansion of motion equations in the uncalibrated case. *CVGIP: Image Understanding*, 64(1):128–146, July 1996.

[24] T. Viéville, O. D. Faugeras, and Q.-T. Luong. Motion of points and lines in the uncalibrated case. *The International Journal of Computer Vision*, 17(1):7–42, Jan. 1996.

[25] T. Viéville and D. Lingrand. Using singular displacements for uncalibrated monocular visual systems. In *4th ECCV*, volume 2, pages 207–216, Apr. 1996.

[26] T. Viéville, C. Zeller, and L. Robert. Use vieville-zeller-etal:96. *None*, 1995.

[27] A. M. Waxman, B. Kamgar-Parsi, and M. Subbarao. Closed-Form Solutions to Image Flow Equations for 3D Structure and Motion. *The International Journal of Computer Vision*, 1:239–258, 1987.

[28] Z. Zhang, R. Deriche, O. Faugeras, and Q.-T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence Journal*, 78(1-2):87–119, 1994. Appeared in October 1995, also INRIA Research Report No.2273, May 1994.

A The first order motion equation.

The fundamental motion equation. Following [23], if order to estimate the motion parameters $\mathbf{C}$ and $\mathbf{v}$ from 2D data ($\mathbf{m}$ and $\dot{\mathbf{m}}$), we have to eliminate Z in (3), easily done by multiplying by $(\mathbf{m}\wedge\mathbf{v})^T$. We obtain the "retinal motion" equation:

$$\begin{array}{rcl} (\mathbf{v}\wedge\mathbf{m})^T\,\dot{\mathbf{m}} & = & (\mathbf{v}\wedge\mathbf{m})^T\,\mathbf{C}\,\mathbf{m} \\ & \Leftrightarrow & \\ |\dot{\mathbf{m}},\mathbf{v},\mathbf{m}| & = & \mathbf{m}^T\,\underbrace{[[\mathbf{v}]_\wedge\,\mathbf{C}]_e}_{\mathbf{S}}\,\mathbf{m} \end{array} \tag{29}$$

where $\mathbf{S}$ is the symmetric or even part of $[\mathbf{v}]_\wedge\,\mathbf{C}$ (i.e. $[\mathbf{M}]_e = (\mathbf{M}+\mathbf{M}^T)/2$), verifying the constraint $\mathbf{v}^T\,\mathbf{S}\,\mathbf{v} = 0$, with the following reverse relation (see [23] for the derivation):

$$\mathbf{C} = -\,[\mathbf{v}]_\wedge\,\mathbf{S}\,[\mathbf{I}+\mathbf{v}\,\mathbf{v}^T] + \mathbf{v}\,\alpha^T + \beta\,\mathbf{I}$$

for some α and β. We have taken $||\mathbf{v}|| = 1$, in the previous relation, for simplicity.

Therefore, $\mathbf{C}$ is defined from (29) up to 4 degrees of freedom, and subject to 5 independent constraints, though the relation: $[\mathbf{v}]_\wedge \, \mathbf{C} - \mathbf{C}^T \, [\mathbf{v}]_\wedge = 2\,\mathbf{S}$. In other words, the motion parameters are not observable from this equation, while they are from the second order motion equation as reviewed in the next section.

This *retinal motion equation* corresponds to the fundamental matrix equation in the discrete case. Its geometric interpretation is that *the component of the reduced retinal velocity* $\dot{s} = \dot{\mathbf{m}} - \mathbf{C}\,\mathbf{m}$ *normal to the epipole line (which vector is* $\mathbf{d} \equiv \mathbf{v} \wedge \mathbf{m}$*) vanishes.*

Structure from motion. The *structure from motion* equation is defined at any point of the retina, except at the *focus of expansion* (i.e. $\mathbf{m}_o \equiv \mathbf{v}$). We obtain, from (3):

$$|| \, [\dot{\mathbf{m}} - \mathbf{C}\,\mathbf{m}] \wedge \mathbf{m}|| = \frac{1}{Z} \, ||\mathbf{m} \wedge \mathbf{v}|| \tag{30}$$

This equation, is "optimal" in the sense that it is the only one which can be derived from the retinal motion information to compute the "affine depth" Z if defined. This has been discussed previously [23].

Finally, the fundamental motion equation (3) is equivalent to three equations: (i) the motion equation (29), (ii) the structure from motion equation (30) and (iii) the structure evolution equation (5). These three equations correspond to the usual decomposition of a motion equation.

B The n-th order motion equation.

Now, if we want to provide the equations at the n-th order, the best formulation has been provided by [13]. Let us briefly review their main result.

We can easily translate definition 5.5.1 of [13] in our notations, switching from Taylor series to derivatives. The motion constraints found by the author can thus be written, after a few algebra :

The n:th order continuous constraint for an un-calibrated setting is :

$$rank \begin{bmatrix} \dot{\mathbf{m}} - \dot{\mathbf{P}}\,\mathbf{m} & \dot{\mathbf{p}} & \mathbf{m} & 0 & \cdots & 0 \\ \frac{\ddot{\mathbf{m}}}{2} - \frac{\ddot{\mathbf{P}}}{2}\,\mathbf{m} & \frac{\ddot{\mathbf{p}}}{2} & \dot{\mathbf{m}} & 2\,\mathbf{m} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\mathbf{m}^{(n)}}{n!} - \frac{\mathbf{P}^{(n)}}{n!}\,\mathbf{m} & \frac{\mathbf{p}^{(n)}}{n!} & \frac{\mathbf{m}^{(n-1)}}{(n-1)!} & \frac{\mathbf{m}^{(n-2)}}{(n-2)!} & \cdots & \mathbf{m} \end{bmatrix} \leq n+1$$

and in the case where $\mathbf{s} \neq 0$ *it is established that the full camera motion is observable from the second order continuous constraints (whereas not from the first one).*

The motion equations proposed by these authors are very nice since they allow to derive motion equations of any order, but in an **implicit form** only. Furthermore, it is rather easy to see that this leads to high degree polynomial equations, not implementable.

On the contrary, our approach where we estimate structure and motion at the same time allows to avoid these problems.

C Motion equations for line tokens.

Although line tokens have been introduced in our system as "sliding-points" we show here that we could have directly manage line equations with our formalism. The point is to show that (i) at a theoretic level the proposed derivations are not only applicable to points even if (ii) at the implementation level these equations are more heavy to use, making easier to use "sliding-point".

Representation of a 3D line. A 3D-line L is defined by its *Plücker* coordinates $(\mathbf{L}, \mathbf{D})$ (see [24] or [8] for a review), i.e. :

$$M \in L \Leftrightarrow M = \lambda\,\mathbf{L} + \underbrace{\mathbf{L} \wedge \mathbf{D}}_{\mathbf{M}_\bullet} \quad with \quad \left\{ \begin{array}{rcl} ||\mathbf{L}|| & = & 1 \\ \mathbf{L}^T\,\mathbf{D} & = & 0 \end{array} \right. \tag{31}$$

where $\mathbf{L}$ is a unit vector in the direction of the 3D-line and $\mathbf{D}$ a vector normal to the plane defined by the 3D-line and the origin, which magnitude $||\mathbf{D}||$ is

equal to the distance from the line to the origin. From (31) and (2) we easily derive the formulas:

$$\dot{\mathbf{L}} = \mathbf{\Omega} \wedge \mathbf{L} \quad \textit{and} \quad \dot{\mathbf{D}} = \mathbf{V} \wedge \mathbf{L} + \mathbf{\Omega} \wedge \mathbf{D} \tag{32}$$

Representation of a 2D line. A 2D line l is defined by a 3D vector $\mathbf{d} \equiv (a, b, c)^T$:

$$\mathbf{m} \in l \Leftrightarrow a\,u + b\,v + c \equiv \mathbf{d}^T\,\mathbf{m} = 0 \tag{33}$$

and if $\mathbf{m}_1 \in l$ and $\mathbf{m}_2 \in l$ then $\mathbf{d} \equiv \mathbf{m}_1 \wedge \mathbf{m}_2$.

Equivalently, given a two points $\mathbf{m}_1 \in l$ and $\mathbf{m}_2 \in l$ in homogeneous coordinates we have :

$$\mathbf{m} \in l \Leftrightarrow \exists (Z_1, Z_2), \mathbf{m} = Z_1\,\mathbf{m}_1 + Z_2\,\mathbf{m}_2 \Leftrightarrow |\mathbf{m}, \mathbf{m}_1, \mathbf{m}_2| = 0 \tag{34}$$

Projection of a 3D line. Combining these two definitions with (1) and using derivations similar to what have been done before, we obtain the relations between the 2D and 3D representations:

$$Z\,\mathbf{m} = \lambda \underbrace{\mathbf{l}}_{Z_\infty\,\mathbf{m}_\infty} + \underbrace{\left[K^{-1}\,\mathbf{l}\right] \wedge \mathbf{d}/det(\mathbf{A})}_{Z_\bullet\,\mathbf{m}_\bullet} \quad \left\{ \begin{array}{ll} \textit{while} & Z\,\mathbf{m} \wedge \mathbf{l} = \mathbf{d}/det(\mathbf{A}) \\ \textit{with} & \left\{ \begin{array}{lcl} \mathbf{l} & = & \mathbf{A}\,\mathbf{L} \\ \mathbf{d} & = & \mathbf{A}^{T\,-1}\,\mathbf{D} \end{array} \right. \end{array} \right. \tag{35}$$

where $(\mathbf{l}, \mathbf{d})$, with $\mathbf{l}^T\,\mathbf{d} = 0$.

This defines the "correct" projection of the Euclidean Plücker representation of the line. It allows to compute the depth of any point on the line, knowing the line parameters projection.

Here $\mathbf{m}_\infty$ is the vanishing-point of the line i.e. the projection of the point a t infinity in the direction of the line direction, whereas $\mathbf{m}_\bullet$ is the projection of the point on the 3D line which distance to the origin is minimal.

Evolution equation of the line. If the line belongs to a plane P (with vector $\mathbf{n}$ and collineation $\mathbf{C}_P$), from (33) and (12) we easily derive, since $\mathbf{n}^T\,\mathbf{l} = 0$:

$$l \in P \Leftrightarrow \left\{ \begin{array}{lcl} \dot{\mathbf{d}} & = & -\mathbf{C}_P^T\,\mathbf{d} \\ \dot{\mathbf{l}} & = & \mathbf{C}_P\,\mathbf{l} \end{array} \right. \tag{36}$$

while, in the case of a non-planar rigid motion we have from (33), (1) and (32):

$$l \in P \Leftrightarrow \left\{ \begin{array}{rcl} \dot{\mathbf{d}} & = & -\mathbf{C}^T\,\mathbf{d} + \mathbf{v} \wedge \mathbf{l}/det(\mathbf{A}) \\ \dot{\mathbf{l}} & = & \mathbf{C}\,\mathbf{l} \end{array} \right. \tag{37}$$

D Technical structure of the program

1. **Choosing Java/C languages** We have chosen to realize this program in Java and C. What are the reasons of this choice?

   First of all Java is actually certainly the best object oriented language on the market. A lot of annoying concepts present in C++ have been cleaned, and many tools are developed with Java to enhance its use, so this might be a choice for the future.

   In particular when making nice Graphic User Interface, Java is a good choice. And we must develop nice G.U.I for the parameter adjustment problem mentioned in our introduction. Actually, the main drawback with Java is that it is 5 to 20 times slower than C. Therefore critical calculations must be implemented in C. At INRIA, tools allow to automatically generates Java/C native interfaces to call C routine as Java methods, while all basic computation libraries are in this Java compatible C restrained language. This ensure a good encapsulation of the structure of the program modules, and the computation overhead for the runtime Java/C interface is negligible with respect to Java execution times.

2. **The libraries** We will now shortly list the already existing libraries to make image processing, acquisition and analysis.

   - Basic mathematical lib :
     - **kvectEst** This class encapsulates Kalman estimations methods
     - **kvectMat** This class encapsulates elementary real-time operations on matrices.
     - **kvectQuad** This class encapsulates elementary real-time operations on quadratic forms.
     - **kvectVec** This class encapsulates elementary real-time operations on vectors.
   - Image Acquisition :
     - **rtvcAcqui** Interface with the SUNWrtvc acquisition system
   - Image Analysis :

- **imaCorners** Image Corner detections using SUSAN module.
- **imaRectif** Performs the rectification of an image using a collineation.
- **imaSmooth** Implementation of the Deriche First-Order Smooth Filter
- **rtvcPoints** Encapsulates functions to detect, track and triangulate points of interest.

3. **Structure of the program**

   The program tree automatically drawn by Javadoc is shown in figure...

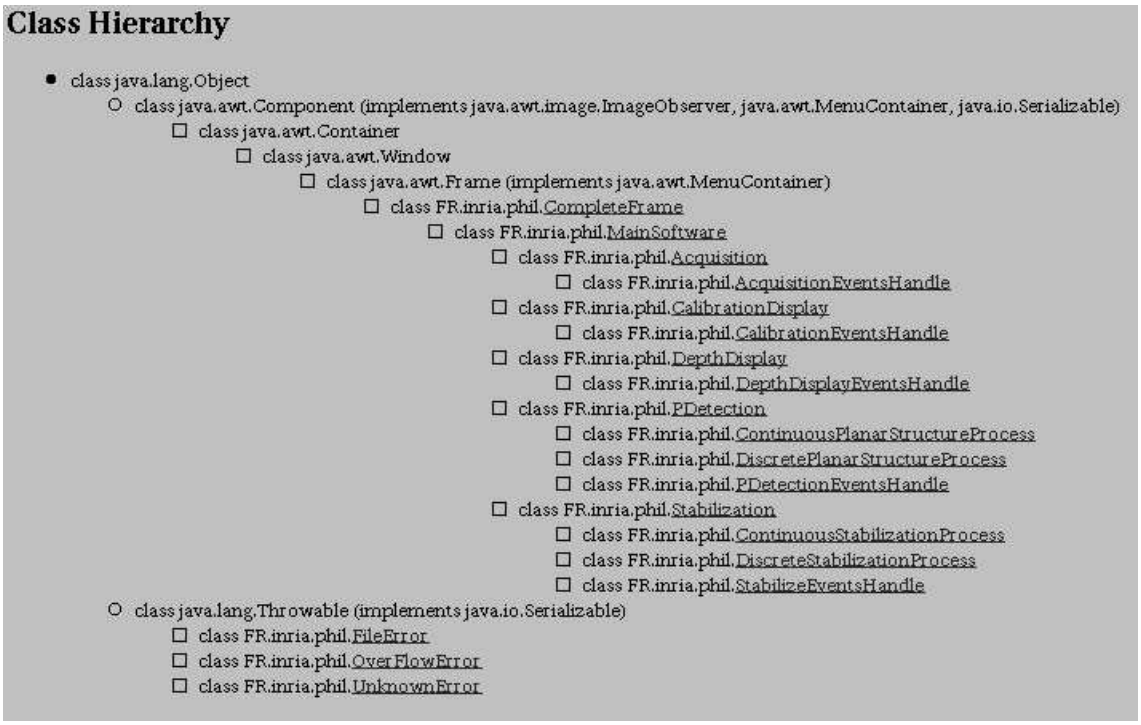

Figure 8: Classes tree

4. **User interface**

First here is an example of the interface offered to the user.

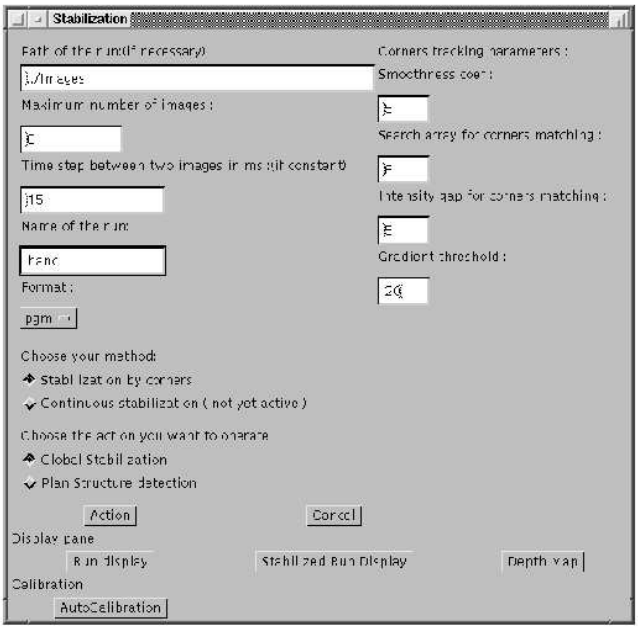

Figure 9: Interface of the project

The path and run-name are here to access the file. The different parameters on the right are for the Corners Tracking and Matching, whose gradient algorithms use various parameters. Finally the user is offered

two methods (but the continuous one has not be implemented), and two different actions global stabilization, and planar structure detection.

When the user launches the global stabilization process, he first sees the corners being tracked. Here is an example of two sequences of 8 images being tracked and matched.

Then after a rectification time and various calculations of errors, the user has access to the stabilized run and related computed data.

E The maple autocalibration program

Here is the small maple routine that enables us to solve the auto-calibration problem in some given cases. When it can't give the parameters, it gives the constraint equations of the model.

```
with(linalg1):

#
# Definition des matrices du probleme
#

m0 := array([u0, v0, 1]): d_m0 := array([d_u0, d_v0, 0]):

x := array([1,0,0]): y := array([0,1,0]): z := array([0,0,1]):

v := array([v1, v2, v3]): w := array([w1, w2, w3]): h := array([h1, h2, h3]):

K := evalm(m0 &* &t m0 + f^2 * (x &* &t x + y &* &t y));

B := evalm(d_m0 &* &t z + d_f / f * (array(1..3,1..3,identity) - m0 &* &t z));
C := evalm(B + K &* tilde(w) + v &* &t h + l * array(1..3,1..3,identity)):

vC := map(normal, evalm(tilde(v) &* C)):
ChgtDeVariables := { f^2 = f2, d_f = f1 * f}:
```

```
#
# Definition de quelques cas particuliers
#

FixedParameters := {
 d_u0 = 0, d_v0 = 0, d_f = 0 }:

PureRotation := {
 v1 = 0, v2 = 0, v3 = 0} union FixedParameters:

NoRotation := {
 w1 = 0, w2 = 0, w3 = 0}:

KnownPrincipalPoint := {
 u0 = 0, v0 = 0, d_u0 = 0, d_v0 = 0}:

#
# Generation des equations de Kruppa
#

MapEq := (eq, m) ->
 map(factor, subs(ChgtDeVariables, subs(eq, evalm(m)))):

GenEq := (eq, m) ->
 MapEq(eq, convert(evalm(m - matrix(3, 3, (i,j)->cat(M,i,j))), set)):

GenKruppa := (eq, m) ->
 eliminate(GenEq(eq, m), {w1, w2, w3, l}):

#
# Resolution de quelques cas
#

# Rotation pure : 3 equations lineaires en (u0, v0) et 2 lineaires en f2
```

```
eq := GenKruppa(PureRotation, C);

sl1 := leastsqrs(map(e -> if not has(e, f2) then e fi, op(2, eq)), {u0, v0});

sl2 := leastsqrs(map(e -> if has(e, f2) then e fi, op(2, eq)), {f2});

convert([
 Return(M31^2+M32^2=0, -1),
 op(sl1),
 op(sl2),
 Return(f2 <=0, -1),
 f = sqrt(f2)
], c);

# Translation pure : pas d'equations

eq2:=GenKruppa(NoRotation union FixedParameters, C);

sl1 := leastsqrs(map(e -> if not has(e, f2) then e fi, op(2, eq2)), {u0, v0});

sl2 := leastsqrs(map(e -> if has(e, f2) then e fi, op(2, eq2)), {f2});
```

Contents

1 Introduction **4**
1.1 Before starting: a short note on basic concepts 5

2 The continuous approach revisited. **7**
2.1 From first order expansion to derivatives. 7
2.2 Comparison with other parameterizations of the retinal motion. 11
2.3 Analyzing the planar case. 12
2.4 Considering line tokens. 15

3 Motion equation applied to image points. **16**

4 Motion equation at the intensity level. 19
4.1 Using the Verri-Pogio operator. 20
4.2 Motion estimation as a correlation operator. 22

5 Experimental Results 22
5.1 Implementation of the algorithm. 22
5.2 Presentation of sample results. 28

A The first order motion equation. 33

B The n-th order motion equation. 34

C Motion equations for line tokens. 35

D Technical structure of the program 38

E The maple autocalibration program 41

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Lorraine : Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 Villers lès Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot St Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, B.P. 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399