Three-dimensional structure from a monocular sequence of images

Jean-Luc Jezouin, Nicholas Ayache

INRIA

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

N° 1282

Programme 6
Robotique, Image et Vision

THREE-DIMENSIONAL STRUCTURE FROM A MONOCULAR SEQUENCE OF IMAGES

Jean-Luc JEZOUIN
Nicholas AYACHE

Août 1990

RR_1282

Three-Dimensional Structure from a Monocular Sequence of Images

Jean-Luc JEZOUIN[1]
MATRA
38, boulevard Paul Cézanne
B.P. 235
78052 Saint-Quentin-en-Yvelines Cédex, France

Nicholas AYACHE
INRIA
Domaine de Voluceau
Rocquencourt, B.P. 150
78153 Le Chesnay Cédex, France

Abstract

In this paper, we address the following problem: given a camera moving in an unknown environment, we want to obtain a 3D description of the environment. We present a unifying approach to the problem of the moving camera by deriving a unique formalism to process uniformly different but complementary features, namely points and linear segments, in order to solve important dynamic vision problems such as tracking and structure from motion. We first present different concept for tracking features: (1) *2D tracker*: 2D features (points and segments) are tracked using an order 1 dynamic model for their evolution; (2) *2D + estimation tracker*: 3D fusion of 2D features is performed recursively, then the value predicted at time t for these 3D features is projected at time $t+1$ onto the camera focal plane and replaces the dynamic model used in the 2D tracker. This allows for introducing 3D informations into the 2D features tracker without prior knowledge of the environment. (3) *3D tracker*: the 2D tracker disappears totally, and all computations are 3D. This tracker combines the simplicity of the 2D tracker and the efficiency of the 2D+estimation tracker. We then describe the mechanisms of fusion that integrate 2D measurements into an estimate of the feature 3D parameters. Uncertainties are taken into account through extended Kalman filtering. Feature parameterization are chosen such as to reduce the algorithmic complexity, simplify the linearization process and ensure numerical stability.

Key-words: Token tracker, Structure from motion, Kalman Filter, Uncertainties, Time fusion.

Reconstruction tridimensionnelle à partir d'une séquence monoculaire d'images

Résumé

Dans cet article, nous étudions le problème suivant : étant donné une caméra en mouvement dans un environnement inconnu, comment obtenir une description de la géométrie tridimensionnelle (3D) de celui-ci ? Nous proposons un formalisme unifié pour résoudre simultanément les problèmes de poursuite d'indices et de reconstruction 3D : ce formalisme permet de traiter des indices visuels différents et complémentaires, les points et les segments de droites.

Nous présentons d'abord plusieurs approches pour résoudre la poursuite d'indices visuels : (1) poursuite 2D : les indices 2D (points et segments) sont poursuivis en utilisant un modèle dynamique d'ordre 1, (2) poursuite 2D avec estimation : la fusion 3D des indices 2D est effectuée récursivement, et la valeur prédite à l'instant t des indices 3D est projetée dans le plan image. Ceci remplace le modèle dynamique utilisé dans le suive 2D et permet d'introduire une information 3D dans la poursuite 2D sans connaissance a priori sur l'environnement. (3) Poursuite 3D : tous les calculs se font en 3D. Cette poursuite combine la simplicité de la poursuite 2D et l'efficacité de l'estimation 3D.

Nous décrivons ensuite les mécanismes de fusion qui permettent d'intégrer des mesures 2D pour estimer les paramètres des primitives 3D. Les incertitudes sont prises en compte grâce à un filtre de Kalman étendu. La paramétrisation des primitives géométriques 2D et 3D est choisie pour assurer la stabilité numérique des calculs, réduire la complexité algorithmique, et simplifier les équations de mesure.

Mots-clefs : Poursuite d'indices, Stéréovision à partir du Mouvement, Filtre de Kalman, Incertitude, Fusion temporelle.

---

[1]supported in part by Direction de la Recherche et des Etudes Techniques grant # 87331

TIME- AND FEATURE- FUSION IN A MONOCULAR SEQUENCE OF IMAGES

Abstract

In this paper, we address the following problem: given a camera moving in an unknown environment, we want to obtain a 3D description of the environment. We present a unifying approach to the problem of the moving camera by deriving a unique formalism to process uniformely different but complementary features, namely points and linear segments, in order to solve important dynamic vision problems such as tracking and structure from motion. We first present different concept for tracking features: (1) *2D tracker*: 2D features (points and segments) are tracked using an order 1 dynamic model for their evolution; (2) *2D+estimation tracker*: 3D fusion of 2D features is performed recursively, then the value predicted at time t for these 3D features is projected at time $t + 1$ onto the camera focal plane and replaces the dynamic model used in the 2D tracker. This allows for introducing 3D information into the 2D feature tracker without prior knowledge of the environment. (3) *3D tracker*: the 2D tracker disappears totally, and all computations are 3D. This tracker combines the simplicity of the 2D tracker and the efficiency of the 2D+estimation tracker. We then describe the mechanisms of fusion that integrate 2D measurements into an estimate of the feature 3D parameters. Uncertainties are taken into account through extended Kalman filtering. Feature parameterization are chosen such as to simplify the linearization process and to ensure numerical stability.

1 Introduction

In many applications, it is desirable to update accurately the navigation of a mobile system using passive means. We restrict the domain to that of an airborne vehicle equipped with only one forward looking camera. Matching images to model yield measurements that can be integrated by the navigation update sub-system. Many attempts can be found in the litterature that try to solve the image-to-model matching problem. The main two approaches are parameter space exploration and feature correspondences. ([2] [3] [4] [5] [11] [14] [16] [17] [22] [24])

The parameter space methods are very systematic and make use of every visual cue in the images. However they present three serious flaws: (i) no partial interpretation of images is available until the final result has been reached, (ii) computational complexity cannot be cut down easily if the multi-dimensional cost surface does not have a simple (convex) shape, and (iii) time fusion is difficult to handle in a natural fashion.

The feature correspondence methods work on a very sparse representation of images and, therefore can recover position and orientation of objects or sensor at unmatched speed rate if the images and model are in the same dimensionality. However they rely heavily on heuristics in an attempt to keep the complexity manageable as soon as local constraint propagation becomes ambiguous, which is the case for 2D to 3D matching. This is true when the geometrical transformation from model coordinate system to image coordinate system is a several-to-one mapping (e.g. projection).

When motion between consecutive images is known or can be computed with small uncertainty [9], [12], it becomes possible to fuse 2D information across images to extract 3D data from the sequence. The matching problem then becomes 3D to 3D and feature correspondance solutions can be used to solve for position and orientation.

This paper focuses on the tracking and fusion processes. Tracking is the establishment of correspondences across consecutive images; fusion is the estimation of 3D parameters through time-integration of 2D parameters. [15] showed how to use time integration to introduce 3D information into a 2D segment tracker without prior knowledge of the environment. This paper explores the construction of a pure 3D tracker that no longer needs a 2D bootstrap. This paper also introduces undifferentiated processing of points *and* linear segments in the tasks of 2D and 3D tracking, motion estimation, and structure from motion.

Section 2 explains feature parameterization (points and linear segments) used throughout the paper; section 3 describes several trackers: purely 2D, 3D bootstrapped by a 2D phase or purely 3D, and indicates when to use them; section 4 describes the fusion mechanisms; some results are given in section 5; concluding remarks are in section 6.

2 Feature geometry

This section explains feature parameterizations that we used to represent points and segments. We believe that both types are usable *simultaneously* because they are complementary: an edgel is detected where gradient is strong in *one* direction, a point is detected where gradient is strong in 2 directions ("L"), 3 directions ("T"), 4 directions ("X") or all directions (isolated point).

2.1 Points

Points are extracted using Moravec's interest operator ([20]) or Plessey's corner detector ([13]).

2.1.1 2D Geometry

A 2D point is represented by its image coordinates:

$$\mathbf{p} = \begin{pmatrix} u \\ v \end{pmatrix}$$

and a covariance matrix of its position error:

$$R = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$$

Usually u and v are considered independent gaussian random variables: $a = c = \sigma^2$ and $b = 0$.

2.1.2 3D Geometry

There exist two parametrizations of 3D points:

normal space: it corresponds to the intuitive three-dimensional space. A 3D point is represented by the vector:

$$\mathbf{p} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

and the covariance matrix of its position error is a 3 by 3 matrix.

projective space: we represent a 3D point by the vector:

$$p = \begin{pmatrix} u \\ v \\ d \end{pmatrix}$$

with $u = fx/z, v = fy/z, d = f/z$, and an associated covariance matrix. This parameterization presents the advantage of a linear relation between 2D measurements and 3D estimate (cf section 4.1.1)

2.2 Linear segments

Linear segments are extracted by performing the following operations: Deriche gradient detector [8], non-maxima suppression, hysteresis thresholding [6], edge linking [10] and line fitting.

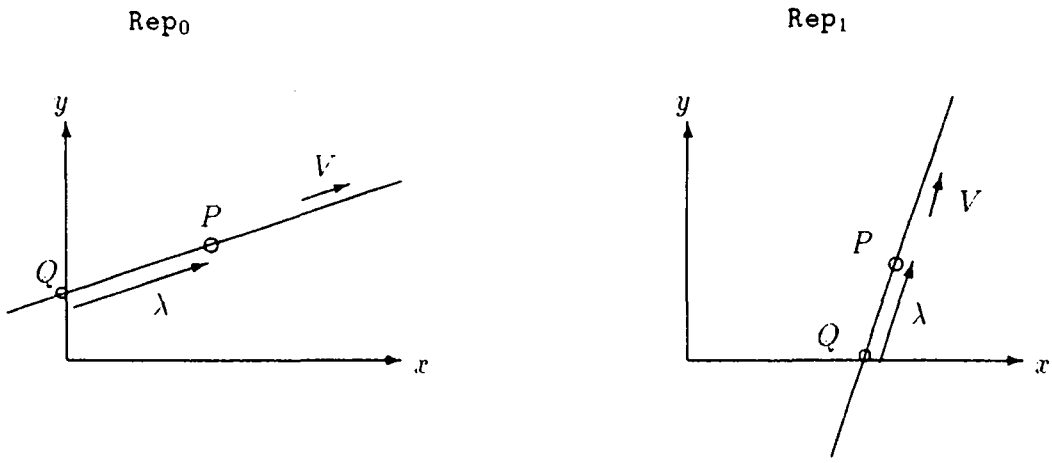


Figure 1: 2D Segment Geometry

2.2.1 2D Geometry

A 2D line is represented by a dot product between the line "state vector" and the 2D point homogeneous coordinates. With the conventions of Figure 1, a 2D line equation may be written as any of two representations, preventing the slope to become infinite [1]:

$$\begin{aligned} \text{Rep}_0 &: \alpha u + v + \mu w = 0 \\ \text{Rep}_1 &: u + \alpha v + \mu w = 0 \end{aligned}$$

In Figure 1, **N** is the unit vector normal to the 2D line; **V** is parallel to it; **Q** is the intersection of the line with the y-axis (Rep0) or the x-axis (Rep1); **P** is the center of the linear segment.

There exist two $3 \cdot 3$ matrices M_{2Di}, $i = 0, 1$ s.t., with $\mathbf{y}' = (\alpha, \mu, 1)^t$:

$$\mathbf{y}'^t M_{2D} \begin{pmatrix} u \\ v \\ w \end{pmatrix} = 0$$

The matrices are given in appendix A.

A segment is fully determined with 2 more parameters: the position of its center along the line λ and its length l, and a $4 \cdot 4$ matrix accounting for the uncertainty to which the segment is measured, R. This representation separates between stable parameters (μ and α) and very uncertain ones (λ, l) which are affected by occlusion, bad detection or segment splits and merges.

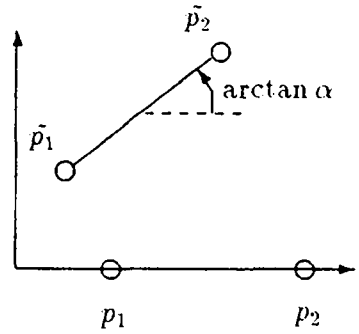


Figure 2: Measurement Covariance

When performing edge detection and linking for an horizontal segment, we can easily attribute a covariance matrix to its endpoints (Figure 2):

$$R_0(p_1, p_2) = \begin{pmatrix} \epsilon^2 & 0 & 0 & 0 \\ 0 & \frac{l^2}{4} & 0 & 0 \\ 0 & 0 & \epsilon^2 & 0 \\ 0 & 0 & 0 & \frac{l^2}{4} \end{pmatrix}$$

where l is the length of the detected segment and ϵ the tolerance in straight line fitting.

By rotating the segment by an angle θ, the covariance matrix becomes:

$$\tilde{R} = \mathrm{Rot}_\theta R_0 \mathrm{Rot}_{-\theta}, \quad \theta = \arctan \alpha$$

Finally, we have to change the parameterization from $\mathbf{x_1} = (\tilde{x_1}, \tilde{y_1}, \tilde{x_2}, \tilde{y_2},)^t$ to $\mathbf{x_2} = (\mu, \alpha, \lambda, l)^t$; there exist an obvious function f (see appendix B) such that $\mathbf{x_2} = f(\mathbf{x_1})$, and we can compute the covariance matrix attached to x:

$$R = \frac{\partial f}{\partial \mathbf{x_1}} \tilde{R} \frac{\partial f}{\partial \mathbf{x_1}}^t$$

whose non-diagonal terms can be shown to be negligible. The diagonal part of R amounts to:

$$R(\mu, \alpha, \lambda, l) = \begin{pmatrix} \epsilon^2 & 0 & 0 & 0 \\ 0 & \frac{2\epsilon^2(1+\alpha^2)^2}{l^2} & 0 & 0 \\ 0 & 0 & \frac{l^2}{8} & 0 \\ 0 & 0 & 0 & \frac{l^2}{4} \end{pmatrix}$$

The function f depends on the representation chosen, but it is easy to alternate from one representation to the other (section B).

2.2.2 3D Geometry

An infinite 3D line maybe in one of the three representations described in the following table:

Rep.	ρ	*Equation*	$\mathbf{V}$	$\mathbf{P}$
0	x	$\begin{cases} y = ax + p \\ z = bx + q \end{cases}$	$(1, a, b)^t$	$(0, p, q)^t$
1	y	$\begin{cases} z = ay + p \\ x = by + q \end{cases}$	$(b, 1, a)^t$	$(q, 0, p)^t$
2	z	$\begin{cases} x = az + p \\ y = bz + q \end{cases}$	$(a, b, 1)^t$	$(p, q, 0)^t$

A point on the 3D line verifies:

$$(x, y, z, 1)^t = \begin{pmatrix} \mathbf{V} & \mathbf{P} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \rho \\ 1 \end{pmatrix}$$

$\mathbf{V}$ is the unit vector parallel the the 3D line; $\mathbf{P}$ is the point at the intersection of the line and the yz (resp. xz,xy) plane in representation Rep0 (resp. Rep1, Rep2).

There exist three 4 by 5 matrices M_{3DV_i} and three 4 by 5 matrices M_{3DP_i} $(i = 0 \ldots 2)$ s.t. with $\mathbf{x}' = (a, b, p, q, 1)^t$:

$$(x, y, z, 1)^t = \begin{pmatrix} M_{3DV}\mathbf{x}' & M_{3DP}\mathbf{x}' \end{pmatrix} \begin{pmatrix} \rho \\ 1 \end{pmatrix} = (\rho M_{3DV} + M_{3DP})\mathbf{x}'$$

The matrices are given in appendix A.

3 Tracking

We mentioned in the introduction the intimate relationship between tracking and 3D estimation. This section will first present a realization of 2D tracking, where image features (points and segments) are tracked using an order 1 dynamic model for their time evolution in the image plane. The formalism used encompasses all kinds of features, and will be illustrated for points and segments. Next we will discuss the necessity to "go 3D" to sometimes improve the tracking. We will then present a pure 3D tracker. Choosing between a 2D or a 3D tracker is essentially a question of camera and object motion.

3.1 Image feature tracker

The parameterizations we use have quasi-independent variables since feature measurement covariance matrices are diagonal (for points) or nearly diagonal (for segments). For each parameters in $x_i \in \{u, v\}$ (points), or $x_i \in \{\mu, \alpha, \lambda, l\}$ (segments), a scalar Kalman filter updates its value.

State equation:

$$\begin{pmatrix} x_i \\ \dot{x}_i \end{pmatrix}_{k+1} = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ \dot{x}_i \end{pmatrix}_k + \begin{pmatrix} 0 \\ 1 \end{pmatrix} q_{i_k}$$

Measurement equation:

$$z_{ik} = x_{ik} + v_{ik}.$$

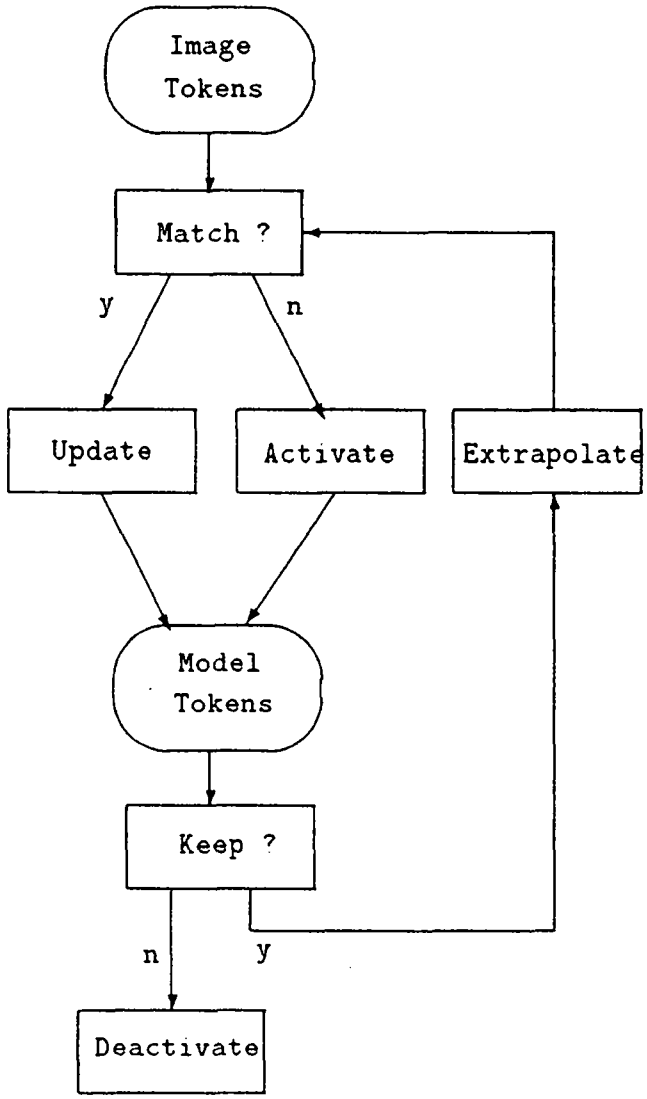


Figure 3: 2D Feature Tracker

The filter equations are extremely simple since the state is measured in the case of 2D tracking. The model noise q_i can be related to an upper-bound to the acceleration $\ddot{x}_i$. This is necessary since the filter will update only the value x_i and its first derivative $\dot{x}_i$.

The correspondence of measured feature to feature predicted from the model is determined by a linear complexity algorithm using the Mahanalobis distance on the feature parameters ($\{\mu, \alpha, \lambda, l\}$ for segments, and $\{u, v\}$ for points). Bucketing can bring an order of magnitude speed-up. Other attributes such as intensity and oriented gradients are used to remove ambiguous candidates. The affectation of a measured feature to a model feature (prediction) is a one-to-one mapping, depending on which pair is the best as seen from the measured feature *and* the model feature points of view. The image flow is depicted in Figure 3. [7] presented a very close implementation of a 2D segment tracker.

The feature tracker provides tracks which are correspondences of features from image

to image along with their (smoothed) estimates and attached covariance matrices. Tracks are terminated upon a confidence factor that represents the number of occlusions tolerable during which estimates and covariances are merely propagated. Tracks are initialized every time a measured feature has not been matched to an already existing model feature.

3.2 Cooperation 2D-tracking / 3D-estimation

The tracking and 3D reconstruction processes are somehow antinomic since the first one will be efficient under small camera motion when the second one draws its accuracy from large camera motion. Small feature displacement between consecutive images will indeed facilitate correspondences and reduce the amount of search, therefore it will yield very few ambiguities. Besides, the simple dynamic model of the Kalman filter in the feature tracker will be sufficient if the speeds of features in the image plane remain quasi-constant in time. On the opposite, 3D reconstruction of small motion features will be acceptable only if enough measurements are made available.

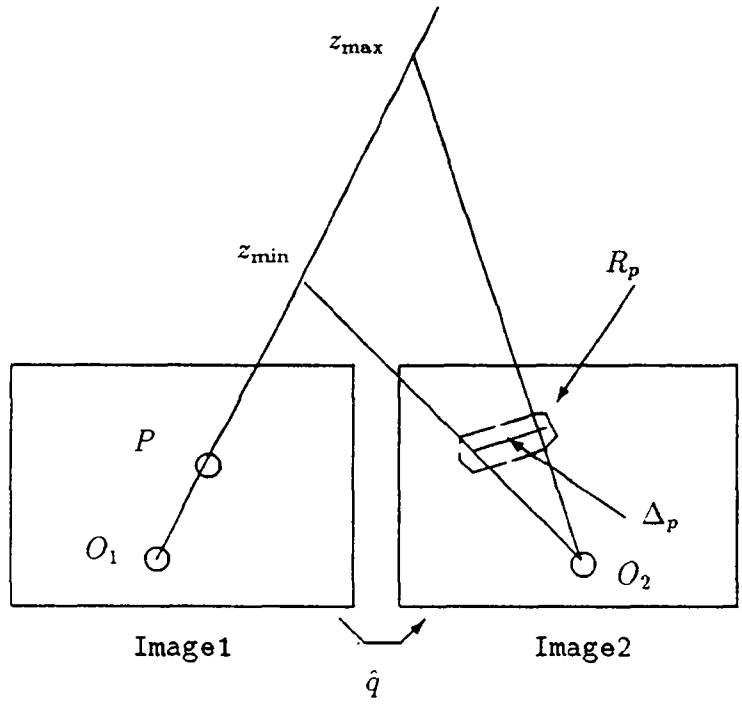
This behavior is affected when features migrate away from the focus of expansion. Prediction errors increase and make correspondences less robust to non-detection and ambiguities. Ironically, the track of a feature may be lost when large displacements would deliver accurate 3D data estimation. Projection of 3D parameter estimates onto the next frame (when camera motion is known) can be used to control this problem. Even though depth computation converges slowly, the fusion process is very accurate at positioning the interpretation line or plane of any feature; hence, the projection of this line or plane onto the next image coordinate frame can be used to predict the image feature location without going through the linearization step involved in the tracker measurement equation.

This composite 2D+estimation tracker is made of a 2D tracker for the *bootstrap mode* that passes 2D data over to the time-integration module. This module produces a 3D estimate of the feature. When the 3D estimate becomes precise enough (small covariance matrix), the system enters a *steady mode* and the predicted projection onto the next image of the 3D estimate is used as a predictor for the 2D tracker, bypassing its own internal model. The 2D tracker plays the role of a track initiator as long as a 3D estimate is not available, then it acts like a mere local matcher when the estimator takes over.

3.3 Three-dimensional tracker

Unlike the 2D traker or the 2D+estimation tracker, the 3D traker should not try to maintain a representation of two-dimensional data. The motivation is that these 2D data are often internal to the tracker and unusable in the rest of the vision system. It would then be helpful (and economical) not to generate them.

Figure 4 shows the similarity between the initiation of a 3D track and the epipolar geometry that a typical matcher could use. When camera motion is available, matching usually proceeds along the epipolar line Δ_p which is the projection of the 3D line $O_1 P$ (O_i is the optical center of the camera in position i), limited to the interval $[z_{\min}, z_{\max}]$, onto the image plane of the camera at position 2, derived from position 1 by a transformation q hypothesized to q_0. The correspondence in image2 of P should lie along the segment Δ_p. However, q_0 is just a prior, so we actually look for correspondences in an area *around* Δ_p, denoted R_p in Figure 4(a).



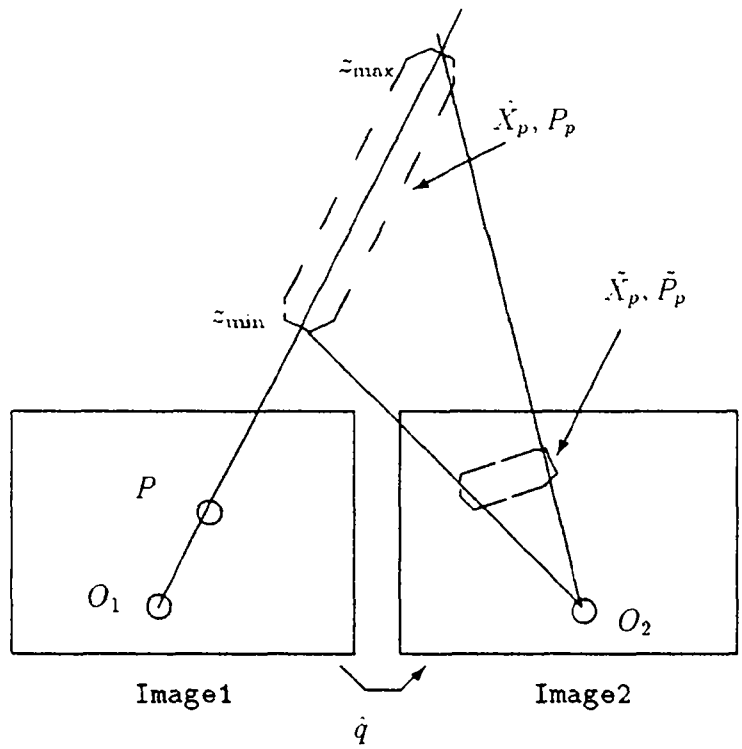


Figure 4: Epipolar Geometry vs 3D Track Initiation

Now, instead of thickening the projection of O_1P, we could consider projecting a thick version of O_1P, that could be an elongated ellipsoid centered around $(z_{\min} + z_{\max})/2$ and of standard deviation proportional to $(z_{\max} - z_{\min})/2$. That could be nothing else but the instantiation of a 3D feature from the single 2D feature P.

Conceptually, the 3D tracker is a 2D+estimation tracker with a bootstrap mode reduced to no-ops. Two conditions must be met for the 3D tracker to exist:

1. instantiation: a 3D feature must be instantiable from a *single* 2D measurement: when a feature is detected for the first time, it must initiate a 3D track;
2. estimation: a recursive estimation of the 3D parameters must be possible.

If the two conditions are met, there are two ways to implement the 3D tracker:

3D state/2D measurements: this is the standard way. The relation between the 3D state $\mathbf{x}$ and the 2D measurement $\mathbf{y}$ is:

$$\mathbf{y} = h(\mathbf{x}) + v$$

Appendix C shows how to obtain a linear equation in the general case:

$$\mathbf{z} = H\mathbf{x} + v'$$

where $\mathbf{z}$ is called a *macro-measurement*. This form directly integrates the 2D measurement into the estimation process. Its performance depends heavily on the linearization step (if required).

3D state/3D pseudo-measurements: due to Condition 1. we can generate a 3D *pseudo*-measurement ξ of same nature than $\mathbf{x}$, uniquely from the 2D measurement $\mathbf{y}$. Hence the measurement equation becomes trivial:

$$\xi = \mathbf{x} + v''$$

(v, v', v'' are white gaussian measurement noises).

In both implementations. the correspondence and affectation of measured feature to feature predicted from the model is determined by the same algorithm than for the 2D tracker. Bucketing of projections can also be used to speed up correspondences. 3D attributes are directly copied from 2D measurement attributes to allow a direct comparison with correspondence candidates. Track management is identical to the 2D tracker, using a confidence factor to terminate broken tracks.

Details about instantiation and recursive estimation will be given in section 4 concerning the actual implementation of a point and segment 3D tracker. We shall see that the segment 3D tracker uses the first implementation, whereas the point 3D tracker uses the second form.

3.4 Comparison of trackers

Figure 5 illustrates the 3 approaches. In each drawing, the update phase (measurement equation) is depicted by a "U". and the extrapolation phase (state equation) by an "E".

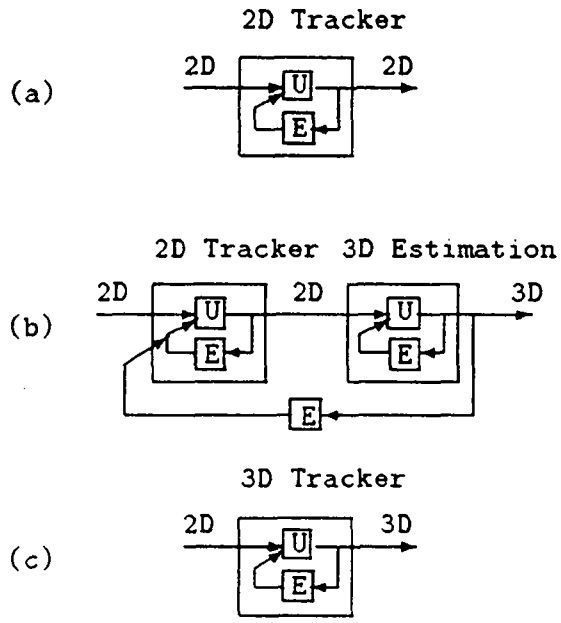


Figure 5: 2D and 3D Trackers

Drawing (a): this drawing illustrates the 2D tracker, where measurements and state are vectors of the same two-dimensional quantities. This tracker is the simplest, both from a numerical complexity and a numerical stability points of view. Numerical complexity is minimum since the measurement equation is always $\mathbf{z} = \mathbf{x} + v$. By the same token, numerical stability is favorized by a linear measurement equation.

This tracker is applicable when the internal model is a good approximation of the real feature evolution, i.e. when the apparent velocity of features is quasi-constant. Also, this tracker will yield good results with moving objects, even with a moving camera, as long as the velocity constraint is enforced.[1]

Drawing (b): is the illustration of the composite 2D+estimation tracker. Moving objects are difficult to handle: the fusion equations depend not only on camera motion and image measurements, but also on unknown object motion. This problem can be categorized as "passive ranging", a research domain by itself which has to make extensive hypothesis on object dynamics (see [21] for instance). When objects come close to the camera, the 3D nature of this tracker gives it a superiority over the 2D tracker, since object motion is no longer uniform. However, difficulties may arise if the measurement equation is extremely non linear, which is fortunately not the case for points and segments (cf section 4).

Drawing (c): here, the 2D tracker and the 3D estimator have been merged to form a single filter: the 3D tracker. This method is computationally cheaper than the previous

[1] If one is able to model more accurately features dynamics, one should obviously use this information through a higher order filter!

one since it runs one filter instead of two. When a track starts near the focus of expansion (FOE), the 2D+estimation tracker and the 3D tracker will perform identically because the feature motion is quasi-uniform around the FOE, giving time to the 3D estimator to overtake the 2D tracker before the track becomes non-uniform. On the opposite, the 3D traker will outperform the other one when a track starts far from the FOE: being non-uniform at the origin, it will prevent the 2D+estimation tracker to even *initiate* a track, since track initiation is solely based on the 2D tracker.

Estimation improvements: numerous techniques have been developped in the litterature [19] to obtain better 3D estimates from 2D measurements. Among them *exponential aging* of measurements allows the filter to "remember" better recent measurements than older ones. The net effect is to give a larger weight to those measurements that are likely to correspond to larger displacements (moving away from the FOE). (We could also think of weighting the features as a function of their displacement instead of the inverse of their age, but no implementation has been tested yet). The *limited memory* filter also tends to overweight the most recent measurements, but in a slightly different manner: it removes the effect of measurements that are older than a preset threshold. In other words, only the N most recent measurements act in the least-squares solution.

4 Estimation of 3D parameters

The fusion operation consists in estimating the parameters of an assumed static 3D feature that projects into a track at each camera position. The equation that relates the parameters of the 3D feature to its 2D projections is typically non linear. Appendix C describes the general linearization process of the measurement equation. The rest of this section explores the equations that relate 2D measurements, 3D estimate and camera motion parameters (the so-called measurement equations) for points, lines and linear segments.

4.1 Point estimation

4.1.1 Instantiation, measurement equation

The advantage of using *projective coordinates* is demonstrated by the simplicity of the measurement equation. A point $\mathbf{x} = (x, y, z)^t$ with projective coordinates $(u, v, d)^t$, with $u = fx/z, v = fy/z, d = f/z$ is related to a 2D point $\mathbf{y} = (i, j)^t$ through the equality:

$$\mathbf{y} = H\mathbf{x} + v$$

with

$$H = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

and v a white gaussian noise.

Each 2D measurement $\mathbf{y}$ corresponds to a 3D cylinder (in projective space) whose axis is $O\mathbf{y}$ (O is the optical center), and whose cross-section is the 2D error ellipse attached to $\mathbf{y}$: this is the 3D point *instantiation* process from a single 2D measurement. The cylinder

can be "simulated" with a 3D ellipsoid, allowing the usual gaussian random variable formalism [18].

Noting the instantiated 3D variable ξ:

$$\xi = H^t \mathbf{y}$$

and the measurement equation becomes:

$$\xi = \mathbf{x} + v'$$

with v' a white gaussian noise.

4.1.2 State equation

The measurement equation does not take camera motion into account. This is left to the state equation:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, q_{k,k+1}) + w_k$$

f applies the camera motion $q_{k,k+1}$ between frame k and frame $k+1$ to the current estimate $\hat{\mathbf{x}}_k^+$ to produce the extrapolated value at time $k+1$, $\hat{\mathbf{x}}_{k+1}^-$. w_k is a gaussian random vector that accounts for the error made in evaluating $q_{k,k+1}$ (inertial device reading or computed motion). Basically, f converts $\hat{\mathbf{x}}_k^+$ into normal space coordinates, translates then rotates this vector by $q_{k,k+1}$, and converts this last quantity back to projective space, yielding $\hat{\mathbf{x}}_{k+1}^-$. Appendix E shows how to perform these operations.

4.2 Line estimation

4.2.1 Measurement equation

A point $(u, v, w)^t$ on line 2D $\mathbf{y} = (\alpha, \mu)^t$ belongs to line 3D $\mathbf{x} = (a, b, p, q)^t$ iff (with $\mathbf{y}' = (\mathbf{y}^t \quad 1)^t$ and $\mathbf{x}' = (\mathbf{x}^t \quad 1)^t$)

$$\mathbf{y}'^t M_{2D} T \begin{pmatrix} M_{3DV}\mathbf{x}' & M_{3DP}\mathbf{x}' \end{pmatrix} \begin{pmatrix} \rho \\ 1 \end{pmatrix} = 0, \forall \rho$$

where T is the projection, rotation and translation matrix.[2] By rearranging terms, and extracting m_V and m_P, sub-matrices of M_{3DV} and M_{3DP} (see appendix A) we get:

$$\begin{cases} (\mathbf{y}^t \quad 1) M_{2D} T m_V \begin{pmatrix} \mathbf{x_V} \\ 1 \end{pmatrix} = 0 \\ (\mathbf{y}^t \quad 1) M_{2D} T m_P \begin{pmatrix} \mathbf{x_P} \\ 1 \end{pmatrix} = 0 \end{cases}$$

with $\mathbf{x_V} = (a, b)^t$ and $\mathbf{x_P} = (p, q)^t$. It is worth noting that we obtain two *independent* equations, relating the observation $\mathbf{y}$ to either the *direction* $\mathbf{x_V}$ of the 3D segment or to its *position* $\mathbf{x_P}$. The estimates $\hat{\mathbf{x}}_\mathbf{V}$ and $\hat{\mathbf{x}}_\mathbf{P}$ will be correlated though since there is a common source of noise through $\mathbf{y}$ in the 2 equations.

[2] 3D segments are evaluated in a global coordinate system; the rotation and translation in question are between this system and the current image.

By noting $M_i = M_{2D}Tm_i$, $i = V, P$ and A_i, B_i, C_i, D_i such that:

$$M_i = \begin{pmatrix} A_{i_{11}} & A_{i_{12}} & C_{i_{11}} \\ A_{i_{21}} & A_{i_{22}} & C_{i_{21}} \\ B_{i_{11}} & B_{i_{12}} & D_{i_{11}} \end{pmatrix}$$

we can rearrange the equations to produce:

$$\begin{cases} (\mathbf{y}^t A_V + B_V)\mathbf{x_V} + \mathbf{y}^t C_V + D_V = 0 \\ (\mathbf{y}^t A_P + B_P)\mathbf{x_P} + \mathbf{y}^t C_P + D_P = 0 \end{cases}$$

which can also be written as:

$$\begin{cases} H_V \mathbf{x_V} - z_V = 0 \\ H_P \mathbf{x_P} - z_P = 0 \end{cases}$$

Since $\hat{\mathbf{x_V}}$ and $\hat{\mathbf{x_P}}$ are correlated we must reintroduce the global state vector $\mathbf{x} = (\mathbf{x_V}^t \quad \mathbf{x_P}^t)^t$. The measurement equation is then:

$$\mathbf{z} = H\mathbf{x}$$

where:

$$\mathbf{z} = \begin{pmatrix} z_V \\ z_P \end{pmatrix} \qquad H = \begin{pmatrix} H_V & O_{12} \\ O_{12} & H_P \end{pmatrix}$$

The measurement noise covariance matrix $R = E(\mathbf{z}\mathbf{z}^t)$ integrates the effect of the noise in $\mathbf{y}$. The equation is (for $i = V, P$):

$$z_i = H_i \mathbf{x_i}$$

By moving noisy terms to the right hand-side:

$$z_i^0 = H_i^0 \mathbf{x_i} + v_i$$

with:

$$v_i = \delta H_i \mathbf{x}_i^0 - \delta z_i$$

substituting the expressions for H_i and z_i:

$$v_i = \delta y^t (A_i \mathbf{x}_i^0 - C_i)$$

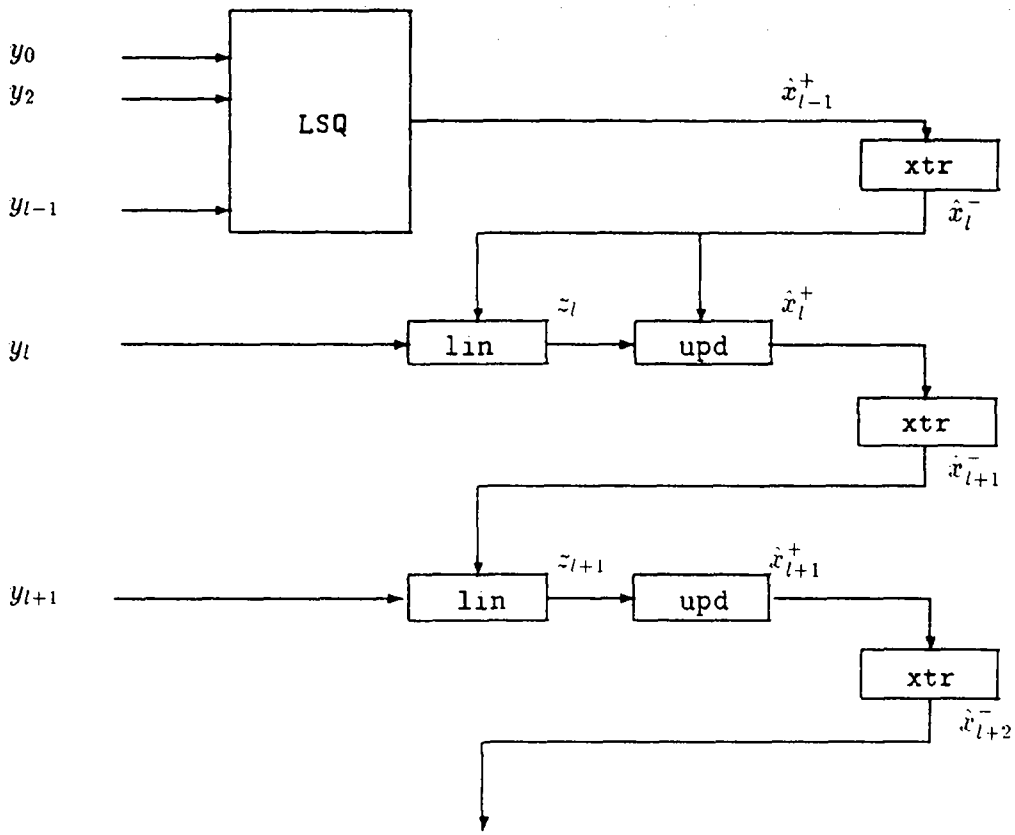
and covariance coefficients are:

$$E(v_i v_j^t) = \text{trace}((A_i \mathbf{x}_i^0 - C_i)(A_j \mathbf{x}_j^0 - C_j)^t R_y)$$

where R_y is the covariance matrix of the observation $\mathbf{y}$. Then:

$$R = \begin{pmatrix} E(v_V v_V^t) & E(v_V v_P^t) \\ E(v_V v_P^t) & E(v_P v_P^t) \end{pmatrix}$$

The linearization of the measurement equation around the estimate of the state vector can induce large errors if done thoughtlessly. A cheap way to avoid this problem is to obtain a first estimate by least-squares batching the first few measurements. After this initialization precaution, recursive estimation of $\mathbf{x}$ can be performed. This scheme is depicted in Figure 6.



LSQ = batch least squares

LIN = linearize

UPD = update

XTR = extrapolate

Figure 6: Stereo from Motion

4.2.2 State equation

Unlike the case of points, it was possible to handle (easily) reference changes in the measurement equation. Therefore the update takes place between 3D parameters expressed in a fixed reference and 2D observations expressed in the current camera coordinate system. Thus, the state equation is straightforward:

$$\begin{pmatrix} a \\ b \\ p \\ q \end{pmatrix}_{k+1} = \begin{pmatrix} a \\ b \\ p \\ q \end{pmatrix}_{k} + I_4 \begin{pmatrix} q_a \\ q_b \\ q_p \\ q_q \end{pmatrix}_{k}$$

The model noise (q_k) takes into account the uncertainty in the transformation matrix from the first image coordinate frame to the k^{th} frame.

For numerical stability, we chose a UD-scalar implementation [18] (see appendix D).

4.3 End point estimation

Estimating 3D endpoints is a requirement for cutting down the complexity of the model matching stage. However, it does not have to be performed recursively, so the best (last) estimate of the 3D line can be used for this purpose. The method is:

- project the 3D line L onto each image plane k yielding 2D lines l'_k;
- project the end points of the 2D lines l_k used for estimating L onto l'_k;
- backproject these points onto L;
- when done for each image, take the median at each end.

With:

$$M_{3D} = \begin{pmatrix} \mathbf{V} & \mathbf{P} \\ 0 & 1 \end{pmatrix}$$

a 3D point parametrized by ρ is:

$$(x, y, z, 1)^t = M_{3D} \begin{pmatrix} \rho \\ 1 \end{pmatrix}$$

and its projection is:

$$w(u/w, v/w, 1)^t = T M_{3D} \begin{pmatrix} \rho \\ 1 \end{pmatrix}$$

The backprojection of a 2D point parametrized by λ is:

$$\begin{aligned} w(u/w, v/w, 1)^t &= w \begin{pmatrix} \lambda \\ -\alpha\lambda - \mu \\ 1 \end{pmatrix}_{\text{Rep}_u} \\ &= w \begin{pmatrix} -\alpha\lambda - \mu \\ \lambda \\ 1 \end{pmatrix}_{\text{Rep}_l} \end{aligned}$$

We can find two 3×3 matrices L_{2D} that allow to write a single equation (cf appendix A):

$$w \begin{pmatrix} \lambda \\ -\alpha\lambda - \mu \\ 1 \end{pmatrix} = L_{2D} T M_{3D} \begin{pmatrix} \rho \\ 1 \end{pmatrix}$$

which, with obvious notations, can be rewritten as:

$$\begin{pmatrix} \lambda \\ -\alpha\lambda - \mu \\ 1 \end{pmatrix} = sM \begin{pmatrix} \rho \\ 1 \end{pmatrix}$$

From there we extract ρ:

$$\rho = \frac{M(0,1) - \lambda M(2,1)}{\lambda M(2,0) - M(0,0)}$$

The back-projected point is given by:

$$(x, y, z, 1)^t = M_{3D} \begin{pmatrix} \rho \\ 1 \end{pmatrix}$$

With such a value of ρ we can compute the error covariance matrix attached to the end points. Let $\mathbf{p} = (x, y, z)^t$. As above:

$$\begin{pmatrix} \mathbf{p} \\ 1 \end{pmatrix} = (\rho M_{3DV} + M_{3DP}) \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}$$

Introducing A, B, C:

$$\begin{pmatrix} \mathbf{p} \\ 1 \end{pmatrix} = (\rho \begin{pmatrix} A_{34} & B_{31} \\ O_{14} & O_{11} \end{pmatrix} + \begin{pmatrix} C_{34} & O_{31} \\ O_{14} & O_{11} \end{pmatrix}) \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}$$

$$\mathbf{p} = (\rho A + C)\mathbf{x} - \rho B$$

$$\delta\mathbf{p} = (\rho A + C)\delta\mathbf{x} + \delta\rho(A\mathbf{x} + B)$$

We notice that $\mathbf{V} = A\mathbf{x} - B$ is the direction of the 3D segment. Then:

$$E(\delta\mathbf{p}\delta\mathbf{p}^t) = (\rho A + C)P_{\mathbf{x}}(\rho A + C)^t + \sigma_\rho^2 \mathbf{V}\mathbf{V}^t$$

where $P_{\mathbf{x}}$ is the error covariance matrix of the 3D line. This expression of the error is very interesting in that it clearly indicates the error *perpendicular* to the 3D segment (the first term) that should decrease steadily as the estimation proceeds, and the error along the 3D segment $\mathbf{V}$ that cannot be evaluated and has to be set: σ_ρ. We could use typically half the measured length.

5 Results

The images of an outdoor scene presented here have been obtained with a realistic image synthesis system that has been developped at out laboratory: it uses a 3D model of the scene (supervised stereo of aerial images) and a few real grey level images taken around the scene. The geometry of any new image is done by z-buffering the 3D model: the radiometry is done by fetching in the grey-level images (which have also been z-buffered after calibration) the exact color of a pixel. Several natural effects (haze, sun position, ...) are under current implementation.

Seq1: 2D tracker on segments For readibility, we limited the number of tracks to 40, all of them lasting more than 25 frames, i.e. 1 second in our experiment. Segments of different images corresponding to the same tracked edge carry the same number. Limitations of the feature tracker (losing or splitting tracks) are essentially due to image segmentation. Off-the-shelf segmentation algorithms are used in order to insure a greater generality for the overall processing chain. The feature tracker has been tested on many more image sequences yielding equivalent (and expected) results. The only parameter that has to be set manually is the minimal length of tracks kept for further processing.

Seq2: 2D+estimation tracker on segments The 2D tracker is the same as before; the 3D estimator implements the equations of section 4.2 and section 4.3. This sequence shows the projection on every image of sequence 1 of the reconstructed 3D endpoints after fusion has been performed; segments in sequence 2 corresponding to tracks in sequence 1 carry the same number also.

Seq3: 2D tracker on points Actually, we use the *same* 2D tracker for both points and segments. An object-oriented implementation [23] allows us to track them concurrently. As previously, points of different images corresponding to the same tracked point carry the same number.

Seq4: 3D tracker on points The 3D tracker is the one described in section 3.3, implementing the equations of section 4.1.1. Besides the tracked points are displayed the projected 3D error ellipsoid. We can notice their correct centering around the real points, but also their large elongation in a direction parallel to the optical axis.

The accuracy of 3D parameter reconstruction depends heavily on camera motion. The larger the baseline, the more accurate the estimation of 3D parameters. Forward-looking imagery has operational advantages (unconstrained trajectory) but is an extremely unfavorable case for stereo accuracy. The relative precision in depth that we have achieved is below 1% for points and correctly detected segments (more than 15 pixels long) up to 10% for unsteady features (e.g. 5-pixel vertical segments).

6 Conclusion

This paper has presented a unifying approach to the problem of the moving camera. It has shown how to derive a unique formalism to process uniformely different but complementary features, namely points and linear segments, in order to solve important dynamic vision problems such as tracking and structure from motion. The dynamic nature of an image sequence was exploited in order to filter the information extracted from individual images so as to reinforce confidence in features. Time-fusion provides redundancy for higher robustness and its implementation (Kalman filtering) keeps track of uncertainties at each step and combines them to achieve a better accuracy. Feature-fusion integrates more low-level information into a richer representation of the environment and makes the system more versatile, able to deal simultaneously with indoor and outdoor scenes. Future research will extend this simultaneous processing of points and segments to the motion

estimation process, and will aim at designing new strategies for matching a reconstructed 3D surface to a 3D model of the scene.

A Matrices for line segments

A.1 2D segments

Matrix	Rep0	Rep1
M2D	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$
L2D	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

A.2 3D segments

Matrix	Rep0	Rep1	Rep2
M3DV	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$
M3DP	$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$
m_V	$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$
m_P	$\begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

B Parameterization change for 2D segments

f is given by (Rep0):

$$\mathbf{x} = \begin{pmatrix} \mu \\ \alpha \\ \lambda \\ l \end{pmatrix} = f(\tilde{x_1}, \tilde{y_1}, \tilde{x_2}, \tilde{y_2}) = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix}$$

with:

$$f_2 = \frac{\tilde{y_2} - \tilde{y_1}}{\tilde{x_2} - \tilde{x_1}}$$

$$f_1 = -\frac{1}{2} f_2(\tilde{x}_2 + \tilde{x}_1)$$

$$f_3 = \frac{1}{2}\sqrt{(\tilde{x}_2 + \tilde{x}_1)^2 + (\tilde{y}_2 + \tilde{y}_1 - 2f_1)^2}$$

$$f_4 = \sqrt{(\tilde{x}_2 - \tilde{x}_1)^2 + (\tilde{y}_2 - \tilde{y}_1)^2}$$

We get f in Rep1 by noting that if $\mathbf{x} = (\mu, \alpha, \lambda, l)^t$ in Rep0 (resp. Rep1), it is transformed into $\mathbf{x}' = (\mu', \alpha', \lambda', l')^t$ in Rep1 (resp. Rep0), and:

$$\begin{pmatrix} \mu' \\ \alpha' \\ \lambda' \\ l' \end{pmatrix} = \begin{pmatrix} \mu/\alpha \\ 1/\alpha \\ -\text{sign}(\alpha)(\lambda + \mu\sqrt{1+\alpha^2}/\alpha) \\ l \end{pmatrix}$$

C Linearization of the measurement equation

In general, the observation equation is $f(\mathbf{x}, \mathbf{y}) = 0$ where $\mathbf{x}$ is the state vector, and $\mathbf{y}$ the measurement vector. Linearization yields:

$$f(\hat{\mathbf{y}}^+, \hat{\mathbf{x}}^-) + \frac{\partial f}{\partial \mathbf{y}}(\mathbf{y} - \hat{\mathbf{y}}^+) + \frac{\partial f}{\partial \mathbf{x}}(\mathbf{x} - \hat{\mathbf{x}}^-) = 0$$

Hence the linearized measurement equation becomes: $\mathbf{z} = H\mathbf{x} + v$ with:

$$\mathbf{z} = -f + \frac{\partial f}{\partial \mathbf{x}}\hat{\mathbf{x}}^-$$

$$H = \frac{\partial f}{\partial \mathbf{x}}$$

$$R = E(vv^t) = \frac{\partial f}{\partial \mathbf{y}} P_{\mathbf{y}}^+ \frac{\partial f^t}{\partial \mathbf{y}}$$

D UD scalar Kalman filter

This decomposition [18] aims at obtaining a higher numerical accuracy accelerating the Kalman filter convergence. Starting from the measurement equation:

$$\mathbf{z} = H\mathbf{x} + v$$

$$R = E(vv^t)$$

we proceed with the following steps:

1. compute: $R^* = \sqrt{R}$ (s.t. $R^* R^{*t} = R$)
2. solve for H^* in $R^* H^* = H$
3. solve for $\mathbf{z}^*$ in $R^* \mathbf{z}^* = \mathbf{z}$
4. set R to identity

The estimation of $\mathbf{x}$ is invariant to "*-ing" the measurement equation, so the Kalman filter equations remain exactly *the same*. Since R becomes equal to the identity matrix, each component of the measurement vector can processed independently of the others in a recursive manner. The covariance matrix decomposes uniquely into two matrices U (upper triangular) and D (diagonal) s.t. $P = UDU^t$. This manipulation doubles the numerical dynamics of the computations and saves numerical operations since the particular structures of U and D can be used in subsequent computations.

E Conversion between normal and projective 3D spaces

Let $\xi = (x, y, z)^t$, the coordinates of a point in 3D space. Let $\mathbf{x} = (u, v, d)^t$ its coordinates in projective space. Let g the function that maps ξ onto $\mathbf{x}$. Then:

$$g(\begin{pmatrix} x \\ y \\ z \end{pmatrix}) = \begin{pmatrix} fx/z \\ fy/z \\ f/z \end{pmatrix}$$

and g is an involution:

$$g = g^{-1}$$

$$\text{if} \quad \xi = g(\mathbf{x}), \text{then} \quad \mathbf{x} = g(\xi)$$

References

[1] N. Ayache. *Vision Stéréocsopique et Perception Multisensorielle: Application à la Robotique Mobile*. Inter-Editions, 1989.

[2] N. Ayache and O. D. Faugeras. Hyper: A New Approach for the Recognition and Positionning of Two-Dimensional Objects. January 1986.

[3] R. C. Bolles and R. A. Cain. Recognizing and Localizing Partially Visible Objects: the Local Feature Focus Method. *The International Journal of Robotics Research*, 1(3):57–82, 1982.

[4] R. C. Bolles and P. Horaud. 3DPO: A Three-Dimensional Part Orientation System. *The International Journal of Robotics Research*, 5(3):3–26, 1986.

[5] R. A. Brooks. Symbolic Reasoning among 3-D Models and 2-D Images. *Artificial Intelligence*, (17):285–348, 1981.

[6] J. F. Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[7] J. L. Crowley, P. Stelmaszyk, and C. Discours. Measuring Image Flow by Tracking Edge Lines. In *Proceedings of International Conference on Computer Vision*, Tampa (FL), December 1988.

[8] R. Deriche. Using Canny's criteria to derive an optimal edge detector recursively implemented. In *The International Journal of Computer Vision*, 1987.

[9] D.B. Gennery. Stereo Camera Calibration. In *Proceedings of the DARPA Image Understanding Workshop*, pages 101–108, 1979.

[10] G. Giraudon. *Chaînage Efficace de Contours*. Technical Report 605, INRIA, 1987.

[11] W. Grimson and T. Lozano-Pérez. Localizing Overlapping Parts by Searching the Interpretation Tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):469–482, 1987.

[12] C. Harris. Determination of Ego-Motion from Matched Points. In *Proceedings of Alvey Vision Conference*, MacGraw-Hill, Cambridge, England, 1987.

[13] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Proceedings of Alvey Vision Conference*, Cambridge, England, 1987.

[14] M. Herman and T. Kanade. *The 3D MOSAIC Scene Understanding System: Incremental Reconstruction of 3D Scenes from Complex Images*. Technical Report.

[15] J. L. Jezouin and N. Ayache. Three-Dimensional Fusion from a Monocular Sequence of Images. In *NATO ARW Series: Sensor Fusion for Computer Fusion*, Grenoble, France. May 1989.

[16] Y. Lamdan and H.J. Wolfson. *Geometric Hashing: a General and Efficient Model-Based Recognition Scheme*. Technical Report.

[17] D.G. Lowe. Three-Dimensional Object Recognition from Single Two-Dimensional Images. *Artificial Intelligence*, 31:355–395, 1987.

[18] P.S. Maybeck. *Stochastic Models. Estimation and Control (vol1)*. Academic Press, 1979.

[19] P.S. Maybeck. *Stochastic Models. Estimation and Control (vol2)*. Academic Press, 1982.

[20] H. P. Moravec. Towards automatic visual obstacle avoidance. In *Proceedings of International Joint Conference on Artificial Intelligence*, August 1977.

[21] J. M. F. Moura, H. L. Van Trees, and A. B. Bagueroer. Space/Time Tracking by a Passive Observer. In *Fourth Symposium on Non-Linear Estimation Theory and its Applications*, San-Diego, CA.

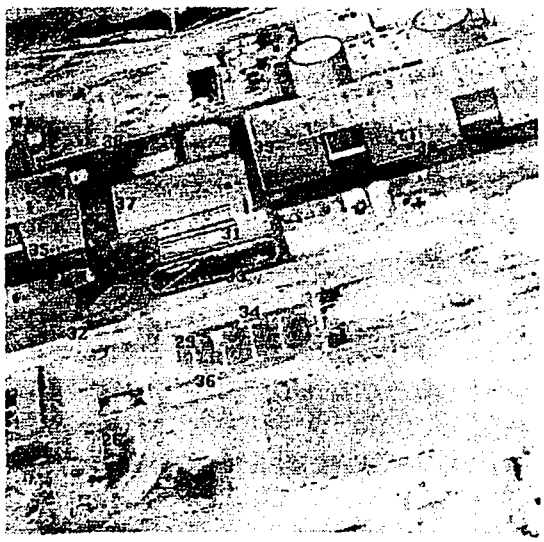
[22] F. Stein and G. Medioni. *Gray-Code Representation and Indexing*. Technical Report, Institute for Robotics and Intelligent Systems, University of Southern California, Los Angeles, August 1989.

[23] B. Stroustrup. *The C++ Language*. Addison Wesley, 1987.

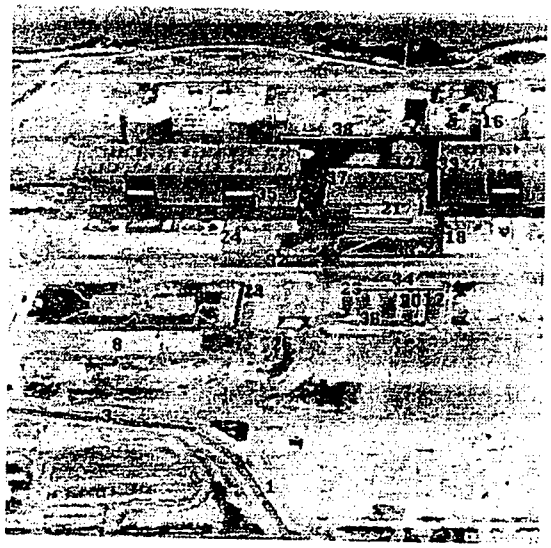
[24] D.W. Thompson and J.L. Mundy. Three-Dimensional Model Matching from an Unconstraind Viewpoint. In *International Conference on Robotics and Automation*, pages 208–220, 1987.

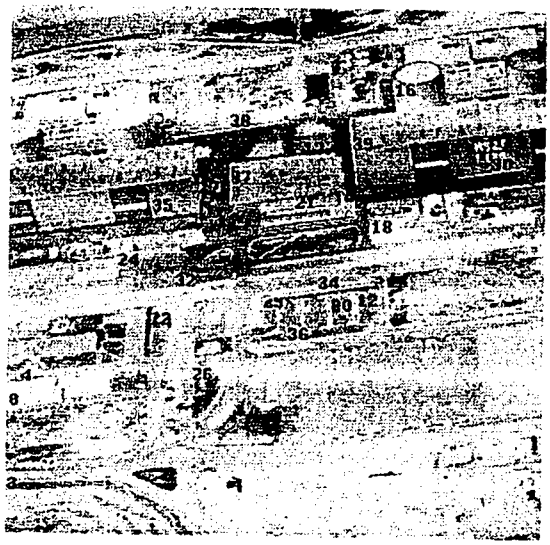
F Tracking and estimating segments

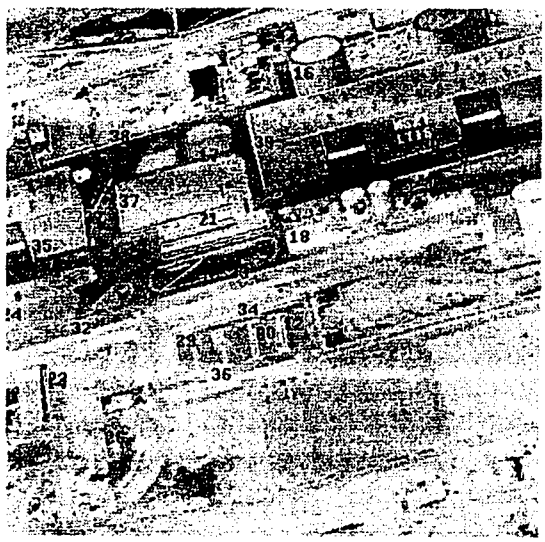
Sequence 1: 2D tracker



Sequence 2: 2D+estimation tracker

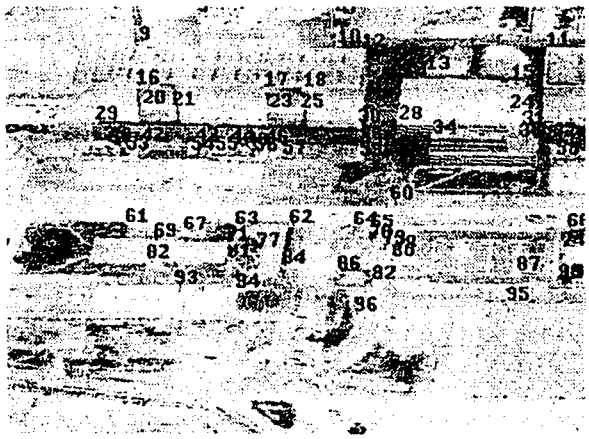


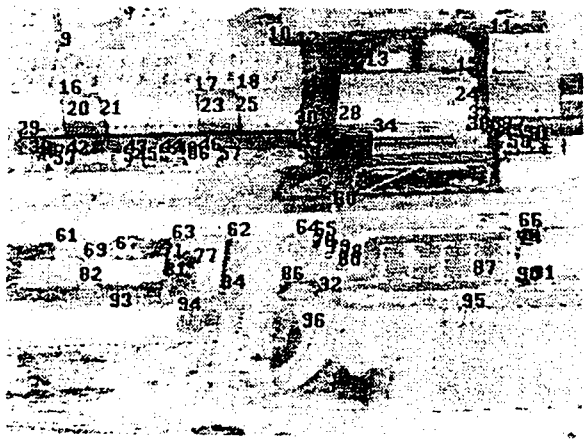


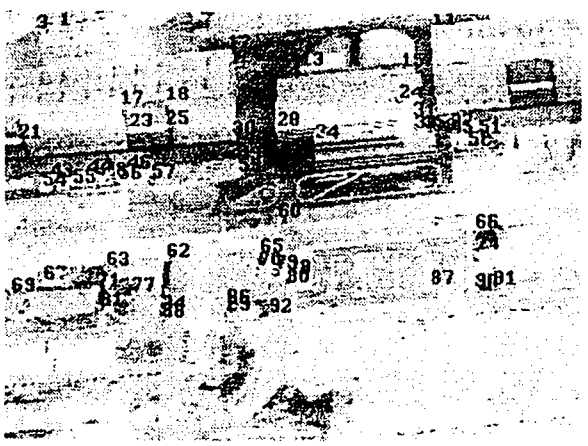


G Tracking and estimating points

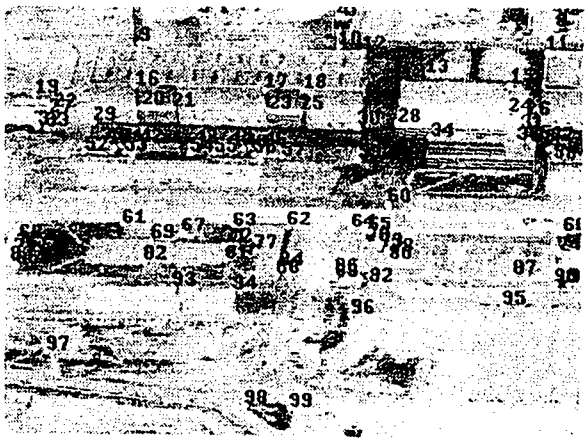
Sequence 3: 2D tracker

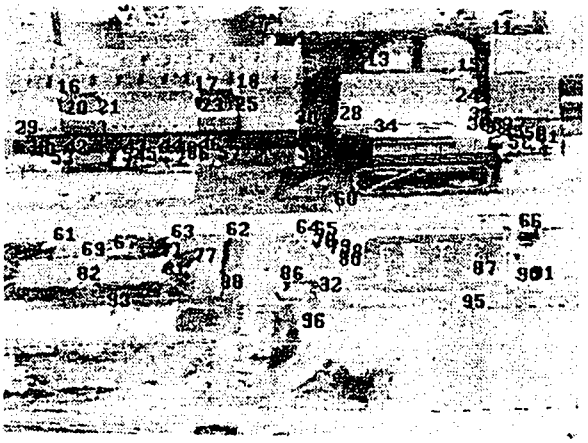






Sequence 4: 3D tracker





3D Tracker on Points

Image 1

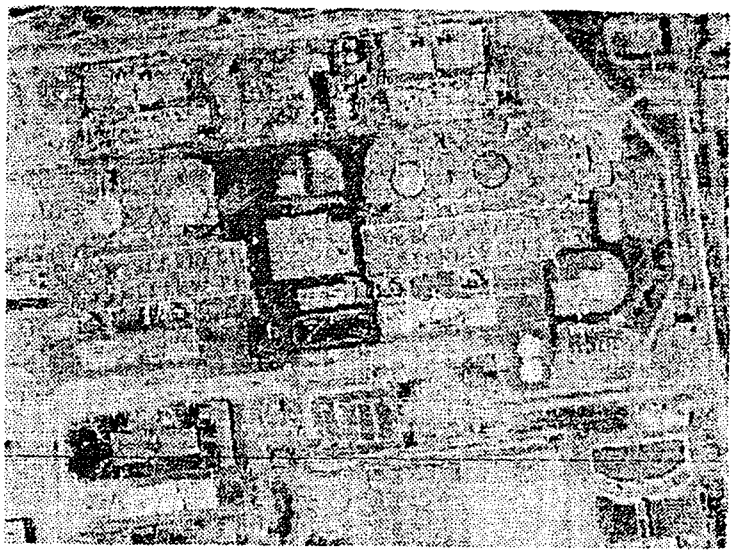
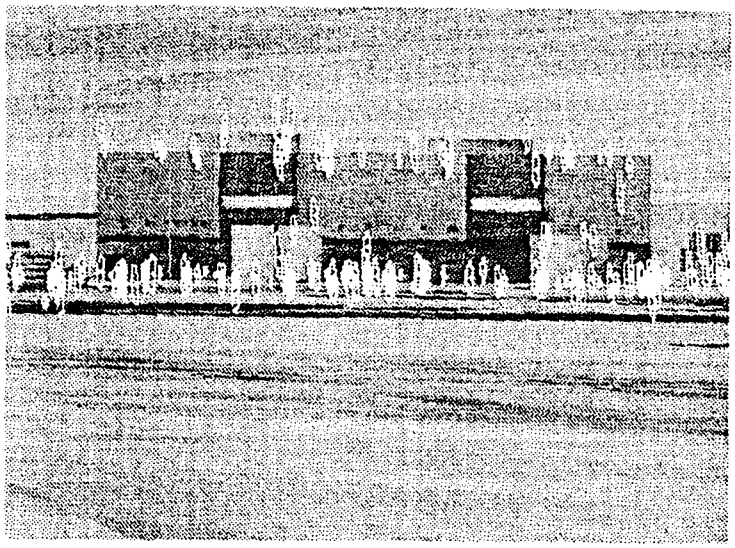


Image 2



The 3D tracker is the one described in section 3.3, implementing the equations of section 4.1.1. The stereo conditions are: the flight is at an altitude of 4500 m above the ground during 2 seconds covering a horizontal distance of 1000 m in 25 images; the buildings are at most 50 m tall. Image 1 (2) shows the projection on a top (side) view of the estimated 3D points and 2-σ error ellipsoids. Are drawn only points such that $\sigma_{max} < 2m$, i.e. with a relative depth error $\Delta z/z < \frac{1}{2250}$.

Imprimé en France
par
. l'Institut National de Recherche en Informatique et en Automatique .

ISSN 0249 - 6399