A FAST AND RELIABLE TOKEN TRACKER

P. STELMASZYK (ITMI)
C. DISCOURS (ITMI and LIFIA)
A. CHEHIKIAN (LTIRF)

ITMI
Chemin des Prés
38240 MEYLAN, France

LIFIA & LTIRF
INPG
46 Av F.VIALET
38031 Grenoble, France

ABSTRACT

The use of depth and motion for 3-D scene analysis requires a system which can accurately and reliably measure image motion. Such measurements may be obtained by tracking the image position and velocities of edge lines observed in a a closely spaced monocular sequence of images. The reliability and precision of such a technique is greatly enhanced by maintaining a model of image flow composed of the position and velocities of tokens constructed from edge lines.

In this paper, we present algorithmic aspects of such a system and show how its complexity can be reduced, for optimizing the hardware implentation. Results on real data are provided and the hardware complexity is evaluated.

1 : INTRODUCTION.

Techniques for inferring depth from motion are notoriously sensitive to noise in flow measurement due to the fact that most mechanical means for displacing a camera generate a considerable amount of vibration. An actively updated "flow model" provides a technique for minimizing the degradation due to mechanical noise and occlusion. An overview of such a process, that matches tokens and maintains the flow model, is represented in Figure 1 :

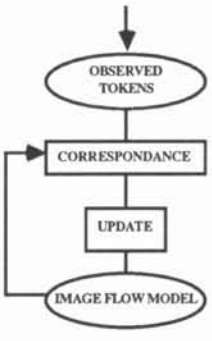

Figure 1

Newly observed tokens arrive continously from a token extraction process and a suitable parametric representation is computed. The tokens extracted from each image, referred to as an "observation", are matched to a flow model. This flow model is composed of "active" tokens which are expressed in an internal representation composed of geometric and dynamic attributes. The geometric attributes are the same as those of the observation tokens. The dynamics attributes include estimates of the speed of the tokens as well as their confidence factor.

The matching process develops a list of observed tokens which correspond to each model token. Matching is based on position and orientation of line segments located in a search area. The matching process is made very fast by avoiding search. There is no attempting to exploit structural information contained in the image.

The dynamic and geometric attributes of each token are then updated using each of the corresponding observed tokens. The update formula for both the geometric and dynamic attributes are based on a Kalman filter. At the same time, the flow model is updated by adding or deleting tokens. A more complete description of this process is described in [CRO 88].

2 : KALMAN FILTER FOR HARDWARE IMPLEMENTATION.

With regard to the prediction and model update phase, the requirement of fast and simple hardware imposes a need to avoid the matrix inversions inherent in the classic Kalman Filter Formulation. It can be done if we assume that an observed token can be expressed as a vector :

$$T^t = (t_1,\ t_2,\ t_3\ ..\ .t_n)$$

with parameters t_i not correlated.Thus, each parameter can be processed independantly, and the flow model of a token can be expressed as a set of flow models, each one representative of one parameter. Then we can save computation time by implementing n Kalman filters working in parallel.

For the same goal we need to reduce the complexity of Kalman filter's equations, this will be possible making some assumptions :

i) We assume that the transition matrix φ is not time dependant, ie, we consider a stationnary model updated at constant time intervals t. If a model token cannot be matched with an image token, we update the model using the predicted value.

ii) We also assume that each parameter t can be expressed as a vector :

$$t_i = (x_i, v_i)$$

In which x_i represents the position and v_i the velocity. We don't estimate the acceleration and we consider that between time t at which the model has been updated and time $t+\Delta t$ for which we want to predict the position of the token, there is no velocity change. Making such an assumption is equivalent to make a model error which commonly leads the Kalman filter to diverge, but it is also well known that such a problem can be avoided by increasing the estimated state covariance matrix Q [LAB].

With these assumptions, the transition matrix can be written :

$$\varphi = \begin{vmatrix} 1 & \Delta t \\ 0 & 1 \end{vmatrix} = \begin{vmatrix} 1 & 1 \\ 0 & 1 \end{vmatrix} \quad \text{if } t = 1$$

Then we can express the Kalman's equations at time t = k as :

predicted state vector :

$$x_{k/k-1} = x_{k-1/k-1} + v_{k-1/k-1} \quad (1)$$

$$v_{k/k-1} = v_{k-1/k-1} \quad (2)$$

predicted state covariance :

$$P_{k/k-1} = \Phi \, . \, P_{k-1/k-1} \, . \, \Phi^T + Q \quad (3)$$

predicted measure convariance :

$$U_{k/k-1} = H \, . \, P_{k/k-1} \, . \, H^T + R \quad (4)$$

Kalman gain :

$$K_k = P_{k/k-1} \, . \, H^T \, [H \, . \, P_{k/k-1} \, . \, H^T + R]^{-1} \quad (5)$$

updated state vector :

$$x_{k/k} = x_{k/k-1} + K_1 \, . \, [Z_k - x_{k/k-1}] \quad (6)$$

$$v_{k/k} = v_{k/k-1} + K_2 \, . \, [Z_k - x_{k/k-1}] \quad (7)$$

updated state covariance :

$$P_{k/k} = P_{k/k-1} - K_k \, . \, H \, . \, P_{k/k-1} \quad (8)$$

In equations (6) and (7), Z_k represents the position of the matched image token, matching being done on the basis of predicted position in accordance with predicted measure covariance. If no matching occurs, then Z_k takes the value of the predicted position and updated state vector will be equal to the predicted state vector. We can notice that equations (3), (4), (5) and (8) are independant from observations, depending only on subscript k for given Q and R, so that filter's parameters defined by these equations can be computed during an initialization phase and read in memory during operating phases.

So, implementing Kalman filter needs only a few arithmetic operations :

* compute predicted position, equation (1) : 1 addition,
* update model token position and velocity, equations (6), (7) : 1 substraction, 2 multiplications, 2 additions.

Due to the simplicity of these operations, the cost of token tracking will be dramatically lower than the cost of matching, and major effort is to be done to reduce this cost.

3 : SEARCH AREA AND MATCHING.

A : PARAMETRIC REPRESENTATION FOR A TOKEN.

By definition, the Token Tracker aims to track any token one can extract from a scene such as corners, junctions, line segments, regions. For historical reasons, we have choosen to deal with line segments extracted from the scene being analysed. Edges are detected by a version of the Canny operator designed and programmed by Deriche [CAN 86][DER 87]. An edge linking step [GIR 86] and a polygonal approximation [BER 85] provide the edge segments on which the tracking is done.

A minimal representation for such an edge segment requires 4 parameters. The classic representation are the cartesian coordinates of the two end points. Unfortunately, this representation is inconvenient for matching, particulary when matching is based on distance normalized by a covariance.

If we consider the uncertainties in extracting edge lines, we can make a number of observations, in particular :

1 - the perpendicular position of a line segment is reliable. Its precision depends of segment extraction process and is usually on the order of some pixels.

2 - the orientation of a line segment has a precision which is proportional to the inverse tangent of the ratio of the precision of the perpendicular position to the length of the segment (i.e. longer segments have a more precise orientation).
The set of these 2 first parameters defines the "carrier line" of the segment with a high accuracy.

3 - the tangential distance from the origin of a line segment is often unreliable, due to random effects wich break edge lines into smaller segments.
This parameter provides the mid-point position along the carrier line with a poor precision.

4 - The lenght of a line segment is not reliable for the same random effects than here above described. This information gives the line extend centered on the mid point.

These considerations suggest the use of parametric representation for a line segment composed of the midpoint, the orientation and the length. It contains four parameters and is mathematically equivalent to a representation in terms of end-points. Furthermore, it allows us to group each parameter as an estimated value and a variance.

The mid-point is normaly expressed in cartesian coordinates. However the parameter uncertainties perpendicular to the line segment are qualitatively different from those along the segment. This suggests a transformation of the center point into parameters (c, d) where :
c is the perpendicular distance to the origine, and
d is the displacement along the line equation from the perpendicular intersect to the mid-point.
The parameters (c, d) are equivalent to rotating the segment by - theta about the origine so that the mid-point lies on the X axis. That is (cf Figure 2) :

$$d = x \cos(\theta) + y \sin(\theta)$$

$$c = -x \sin(\theta) + y \cos(\theta)$$

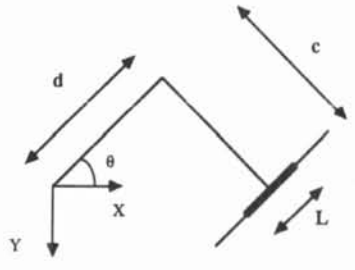

Figure 2

If these four parameters are sufficient for describing a line segment, they have to be completed for the matching process by an euclidan distance between mid points of the predicted and the observed token. In fact, large uncertainty on d introduce a leverage effect that can lead to match 2 segments that are far, in term of euclidean distance, even if c, theta, d and L are close (cf Figure 3).

A simple test rejects for the matching all segments located, in term of euclidean distance, far away the predicted position.

The predicted search area is then determined by a process that tests if the difference on attributes (between observed and predicted value) is less than 3 standard deviations. These tests are followed by another one based on euclidean distance measurement which consists to check overlapp between the two segments. It uses explicitely the lenght of a segment and avoids some empiric threshold. Then the best match is computed between all segments that succed to these 4 tests by using a cost function based on a mahanalobis distance.

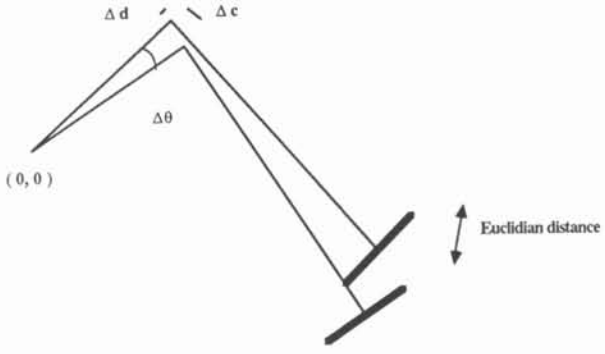

Figure 3

B : HARDWARE IMPLEMENTATION.

From a hardware point of vue, kalman filter computations are based on the use of n kalman filters working in parallel. Each of them is assigned to a parameter of the vector S describing a segment. In a same way, all tests necessary for determining if a token is, or not, inside the search area are computed in the same time, the cost function being evaluated only if all tests succeed.

In our hardware development, we have been using 5 kalman modules assigned to the 5 attributes of the vector S = {c, theta, L, x, y}. The computation of c, theta, L determines of the search area while x and y test the euclidean distance. Such a representation provides several advantages:

- It is less expensive (in term of number of chips and procesing time) to evaluate x and y using a kalman filter, rather than computing them from c and d. Kalman filter requieres only one substraction, 2 multiplications and 2 additions. The computation of x and y (x = d Cos theta - c Sin theta & y = d Sin theta + c Cos theta) needs 4 multiplications, 2 additions and the determination of sinus and cosinus for the predicted and the observed token.
- The 5 components of the vector V are available at the same time and the parallelism is not affected by some complex computations.

One can observe that the tangential distance d hasn't been taking into account. In fact, this measurement doesn't bring any usefull information that should increase the robustness of the matching when the euclidean distance test exists. Its computation can be ignored in order to decrease the hardware complexity.

4 : EXPERIMENTAL ASSESMENT.

The token tracker above presented has been applied on a robotics configuration. In these experiments, a camera is fixed on the robot arm and moves around an object located in the workspace of the robot.Each token in the flow model is assigned an identification number when it is created. This number may be used to identify tokens in snapshots of the flow model.

Figure 4 shows the raw image number 25 and 45 of a sequence in which the camera follows a roughly circular preprogrammed trajectory around an object (a hexagonal Rubik's puzzle).
Figure 5 displays a trace of the correspondance between snapshot dumps of the model after images (25 and 30), (30 and 35), (35 and 40) and (40 and 45). The line segments in each trace show the displacement of the midpoint of each token in image coordinates.

The reliability and robustness of this experimentation is displayed in figure 6 which indicates the matching results for this sequence of images. For each image, the number of tokens in the flow model as well as the number of good and false matches is indicated. This figure stresses the fact that the system is capable of coping with occlusions and degradations due to noise and robot vibrations.

5 : CONCLUSION.

The token tracker is a simple process which provides an elegant and reliable solution to the problem of image flow measurement and image correspondance. The resulting flow model is robust; it is able to tolerate image disturbance due to noise and photometric effects. Token tracking is made possible by maintaining an explicit estimate of the precision as spatial and dynamic attributes using a Kalman Filter.
Making some assumptions, computaion of matrix inversions can be avoided and Kalman filter can be easily implemented in hardware. A suitable representation for tokens allows a simple matching and determination of the search area. This token tracker, that is expected to deal with 10 images per second, is under hardware development.

ACKNOWLEDGMENTS.

This work was performed as part of ESPRIT project P940 in collaboration with R. DERICHE and O. FAUGERAS from INRIA.
The authors want to thank Prof J.L CROWLEY from LIFIA for having introduce the concept of the flow model and the parametric line segment representation and for their constructive suggestions during this work.

BIBLIOGRAPHY.

BER 85 M. BERTHOD
Polygonal approximation of edge chains.
Internal Report INRIA, 1985

CAN 86 J. CANNY
A computational approach to edge detection.
IEEE trans. on PAMI, Vil PAMI-8, N° 6, 1986.

CRO 88 J.L CROWLEY, P.STELMASZYK, C.DISCOURS
Measuring image flow by tracking edge-lines.
Int Conf on Computer Vision, Dec 1988.

DER 87 R. DERICHE
Optimal edge detection using recursive filtering.
First International Conference on Computer Vision.
London (UK), 8-12 June 1987.

GIR 86 G.GIRAUDON
An efficient edge chaining algorithm
Internal Report INRIA, 1986

LAB M. LABARRERE, J.P KRIEF, B.GIMONET
Le filtrage et ses applications.
SUP'AERO, CEPADUES EDITIONS.

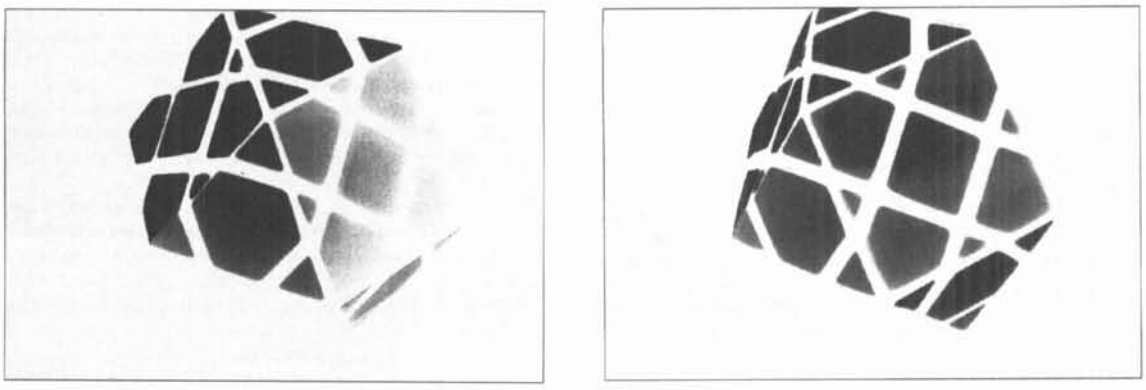

Figure 4
Raw images 25 and 45

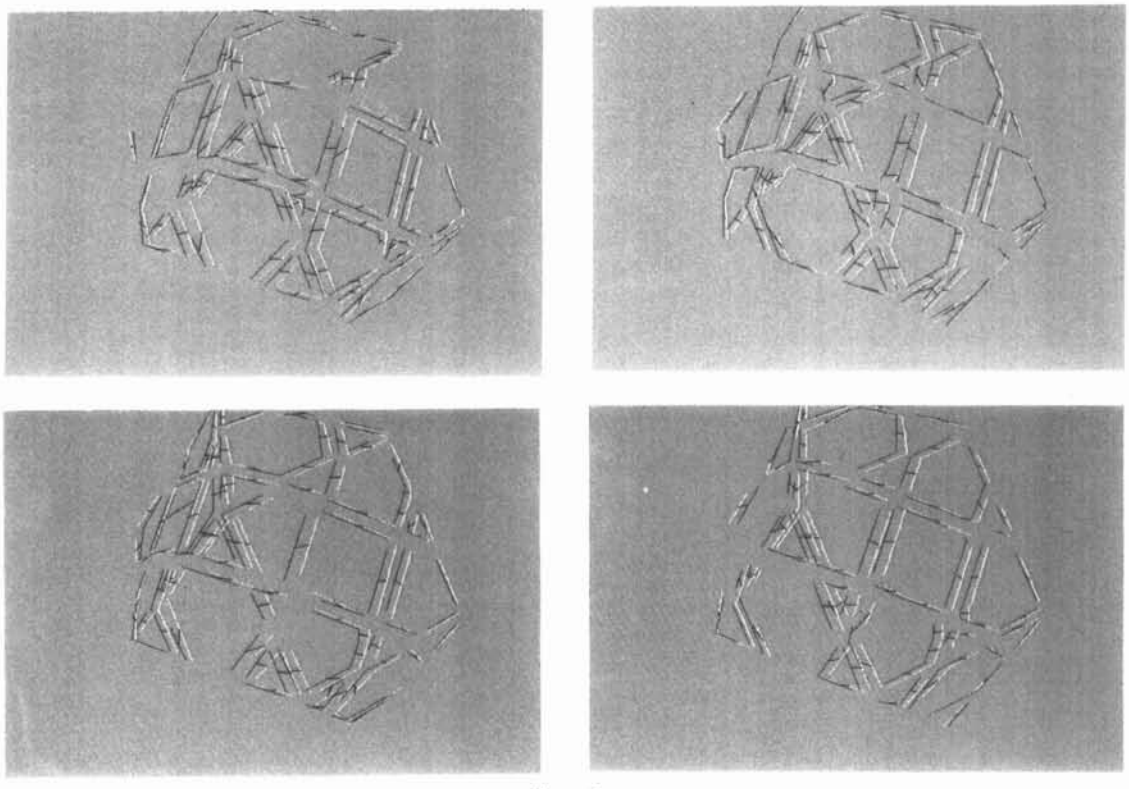

Figure 5
Matches between images (25, 30), (30, 35), (35, 40), (40,45)

Images	Nb of Tokens	Nb of matches	Falses matches
25 - 30	136	84	0
30 - 35	140	89	0
35 - 40	139	86	3
40 - 45	136	75	0

Figure 6
Matching evaluation