

Concordia University  
Electrical and Computer Engineering Department

**A Tool for  
Global Motion Estimation and Compensation  
for Video Processing**

by

Olivier Adda  
Nicolas Cottineau  
Mahmoud Kadoura

**Final Year Project ELEC/COEN 490  
Final Report**

Project Supervisors:

Dr. Aishy Amer and Dr. William E. Lynch

May 5, 2003

## Abstract

This report gives a study of three approaches for global motion estimation to design a tool meant to estimate and compensate global motion in a video sequence. Global motion estimation is a process to estimate the motion of the background in a video sequence caused mainly by the camera motion. It is an important task in variety of video processing applications, such as coding, segmentation, mosaicing, video surveillance. These three approaches are meant to be applied in coding but our investigation aims to apply them for video segmentation and surveillance.

The first method of global motion estimation is based on the minimization of the prediction error, the sum of squared difference (SSD), by a gradient descent applied over a pyramid of two input images. The second one is an improvement of the first one and consists of minimizing the sum of absolute difference (SAD) instead of the SSD in the initial matching. Beside that, it uses prediction and earlier termination. Finally, the third approach is totally different and is based on two stages: a feature tracking followed by a estimation of the global translation parameters using a robust M-approach (or maximum-likelihood approach) followed by a differential refinement stage based on an iterative descending Gauss-Newton minimization of the SAD, which gives all the global motion parameters. The robust M-estimator is used to distinguish between reliable measures and outliers.

In this report we will show the results we obtained by the three algorithms and compare them by using three criteria: the speed, the mean absolute difference error and the robustness. As will be shown, the three algorithms have similar performance in the case of translation, but the one using prediction and the last one based on feature tracking are faster than the first algorithm. The feature tracking seems to be very efficient for global motion estimation applied for video surveillance.

**Keywords:** *Video processing, global motion estimation, compensation, gra-*

*gradient descent, prediction, affine model, M-approach estimator, feature tracking.*

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Project objectives and tasks distribution</b>	<b>3</b>
2.1	Project objectives . . . . .	3
2.2	Tasks distribution for phase 3 . . . . .	3
<b>3</b>	<b>Problem statement</b>	<b>6</b>
3.1	Motion estimation . . . . .	6
3.2	Camera motions . . . . .	7
3.3	Global motion estimation (GME) . . . . .	7
3.4	Difficulties in global motion estimation . . . . .	8
3.4.1	Global/local motion: . . . . .	8
3.4.2	Local changes: . . . . .	9
3.4.3	Global changes: . . . . .	9
3.5	Global motion compensation . . . . .	9
<b>4</b>	<b>Modeling</b>	<b>11</b>
<b>5</b>	<b>Solution approaches</b>	<b>14</b>
5.1	GME technique using hierarchical implementation and gradient descent (Algo. 1) . . . . .	14
5.2	Improved GME using earlier prediction (Algo. 2) . . . . .	17
5.3	Real Time GME using feature tracking (Algo.3) . . . . .	20
5.3.1	Feature matching and initial estimation . . . . .	20
5.3.2	Differential affine estimation . . . . .	21
5.3.3	Wrapping . . . . .	22
<b>6</b>	<b>Results</b>	<b>23</b>
6.1	Implementation . . . . .	23

6.2	GME Technique using hierarchical implementation and gradient descent (Algo.1) . . . . .	25
6.2.1	Determination of the parameters . . . . .	25
6.2.2	Results . . . . .	25
6.3	Improved GME Technique using earlier prediction (Algo.2) . . . . .	30
6.3.1	Implementation and Determination of the parameters . . . . .	30
6.3.2	Results . . . . .	30
6.4	Real time technique using feature tracking (Algo.3) . . . . .	32
6.4.1	Determination of the parameters . . . . .	32
6.4.2	Results . . . . .	35
6.5	Comparison . . . . .	38
6.5.1	Criteria . . . . .	38
6.5.2	Results and analysis . . . . .	38
<b>7</b>	<b>Conclusion and future work</b>	<b>44</b>
<b>A</b>	<b>Algorithm modules</b>	<b>45</b>
A.1	GME technique using hierarchical implementation and gradient descent	45
A.2	Improved GME Technique using earlier prediction . . . . .	46
A.3	Real Time GME using feature tracking . . . . .	47
A.4	Compensation and error modules . . . . .	48
<b>B</b>	<b>Implementing modules and functions</b>	<b>49</b>
	<b>References</b>	<b>52</b>

**List of Figures**

1	Design of the tool. . . . .	2
2	Typical types of camera motion. . . . .	8
3	Projection of a moving object. . . . .	11
4	A typical 2-D motion field. . . . .	12
5	Motion fields corresponding to (a)camera zoom (b)camera rotation. . .	13
6	Hierarchical estimation. . . . .	15
7	Block diagram for the case of a three-level hierarchical implementation	16
8	Block diagram of the second algorithm extracted from [1]. . . . .	19
9	Block diagram of the proposed algorithm extracted from [2]. . . . .	20
10	Car sequence: Frame n.15 . . . . .	23
11	Prlcar sequence: Frame n.12 . . . . .	24
12	Tennis sequence: Frame n.4 . . . . .	24
13	Atp sequence: Frame n.5 . . . . .	24
14	Simulations Car sequence, algo.1. . . . .	26
15	Simulations Prlcar sequence, algo.1. . . . .	27
16	Simulations Tennis sequence, algo.1. . . . .	28
17	Simulations Atp sequence, algo.1. . . . .	29
18	Simulation Prlcar sequence, algo.2 . . . . .	31
19	Selection of the number of features. . . . .	33
20	Difference image for different numbers of features. . . . .	34
21	Simulations Car sequence, algo.3 . . . . .	35
22	Simulations Prlcar sequence, algo.3 . . . . .	36
23	Simulations Atp sequence, algo.3 . . . . .	37
24	Comparison for Car . . . . .	39
25	Comparison for Prlcar . . . . .	39
26	Comparison for Tennis . . . . .	39
27	Comparison for Atp . . . . .	40

# 1 Introduction

Video processing is today one of the most important research field among multimedia types because of its large number of applications. Video processing consists of manipulation of visual data in order to analyze it, compress it, or segment it, for example. Some applications of video processing are: remote vision, video conference and video surveillance.[3].

The goal of our project is to design a tool for global motion estimation and compensation, which is aimed to be applied to video surveillance and segmentation [4]. From a sequence of images our tool should be able in a first step to estimate the global motion and in a second step to compensate it.

In this report, in section 2 we will define video processing, motion estimation and more particularly global motion estimation. In section 3 we will describe different methodologies to estimate global motion and some models to the problems. In section 4 we will present three different approaches for the estimation stage [5], [1], [2]. In section 5, we will the results and a comparison of these three algorithms. Section 6 concludes this report. In appendix several implemented functions are described.

Briefly, our tool treats an input of two images, estimates the global motion between both of them and from the first one predicts the second one called compensated image, as described in Fig. 1.

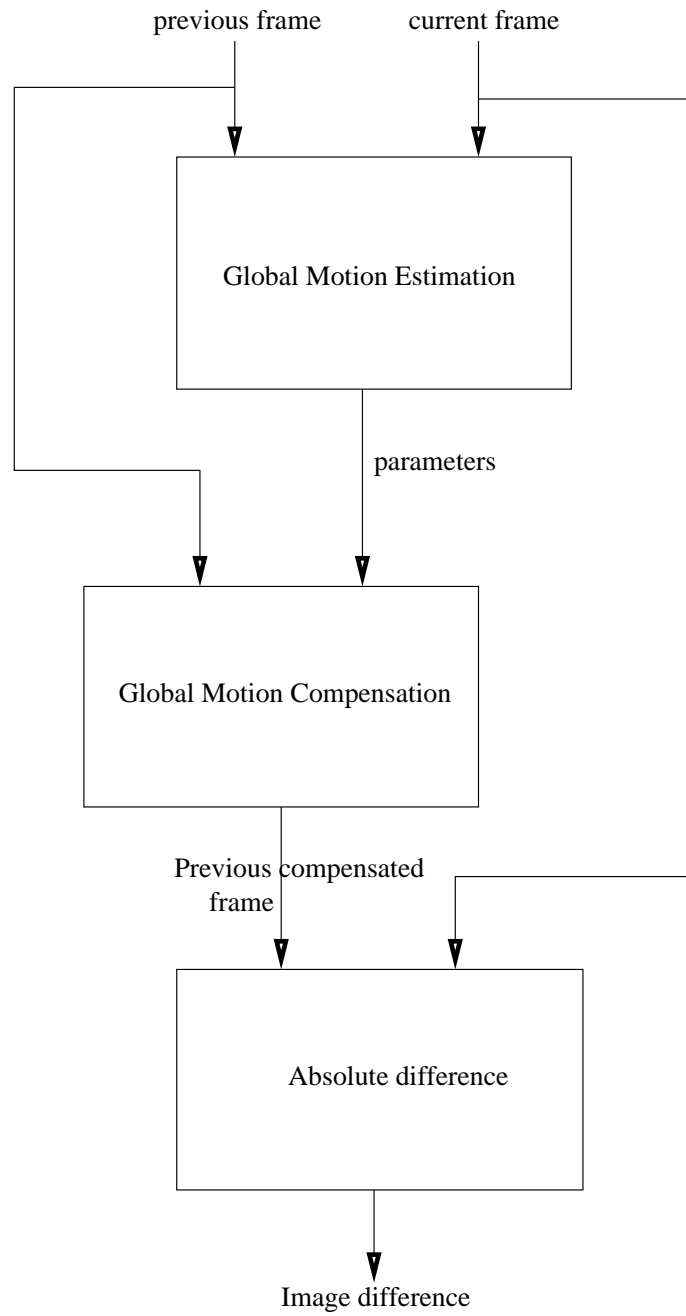


Figure 1: Design of the tool.

## 2 Project objectives and tasks distribution

### 2.1 Project objectives

- To implement three algorithms for Global Motion Estimation.
- To implement a program for Global Motion Compensation.
- To simulate our algorithms on different kind of video sequences.
- To compare the results.

### 2.2 Tasks distribution for phase 3

C: Completed

N: Not completed

- Nicolas Cottineau
  - Algorithm 1:
    - \* Affine part (C)
    - \* Tests (C)
  - Algorithm 2:
    - \* Implementation (C)
    - \* Tests (C)
  - compensation (C)
  - Report:
    - \* Algorithm 2 (C)
    - \* Results of algorithms 1 and 2 (C)
    - \* Modules for algorithms 1 and 2 (C)
    - \* Conclusion (C)
    - \* Editing in Latex (C)

- Final correction and editing (C)
- Final presentation (algo 1 and 2) (C)
- Mahmoud Kadoura
  - Algorithm 3:
    - \* Understanding and analysis (C)
    - \* Implementation of the differential affine estimation (N)
  - report
    - \* Theory (C)
    - \* Algorithm 1 (C)
    - \* Rereading (C)
  - Final presentation (Introduction and theory) (C)
- Olivier Adda
  - Algorithm 3:
    - \* Understanding and analysis (C)
    - \* Implementation of the feature tracking (C)
    - \* Implementation of the estimation of the global translation parameters (C)
    - \* Tests (C)
  - Report:
    - \* Introduction (C)
    - \* Abstract (C)
    - \* Algorithm 3 (C)
    - \* Results of algorithm 3 (C)
    - \* Comparisons (C)

- \* Modules for algorithm 3 (C)
- Final correction and editing (C)
- Final presentation (algo 3) (C)

### 3 Problem statement

Among the goals of the video processing, is to describe changes between consecutive images of a video sequence. In fact, it deals with describing or "estimating" the object motions and other effects, such as illumination changes or camera motion. A video sequence is composed by a continuation of images. An image is constituted of pixels and between two images, we can decompose the motion of any pixel into global motion and local motion. Mainly, global motion is constituted by the camera motion whereas local motion concerns object motion. Local or object motion is the movement of some pixels or blocks in the frame and not all of the pixels of the image.

With motion estimation we can for example:

- ⇒ understand the content of an image sequence (video analysis).
- ⇒ reduce temporal redundancy of video (video compression)
- ⇒ stabilize video by "removing" noisy global motion (video stabilization)

In this study, we will focus on estimating camera motion. Therefore in the following sections, the use of the term global motion refers to camera motion.

#### 3.1 Motion estimation

Motion estimation is one of the topics having attracted many research activities in the field of video processing. By using different motion estimation techniques, it is possible to estimate object motion or camera motion observed in video sequence.

If multiple objects moving in different direction are captured by a video camera, these object's movement can be described only by terms of local motion. But in the case of a non-stationary camera, which means that the camera moves during the capture such as track, pan and zoom (defined in section 2.2), individual local motion vectors are "contaminated" by the camera motion. This is why it is necessary to estimate first camera movement then to compensate it and finally to recompute local motion vectors by considering the estimated camera motion [6].

Motions can be described either by a 2-D motion model or by a 3-D motion model. Two-Dimensional motion estimation is an important part of any video processing system. 2-D motion estimation is often the prerequisite step for 3-D structure and motion estimation. Motion estimation methods can be very different according to the desired application. Then the estimated motion vectors can be used to produce a motion-compensated prediction of a frame for various applications.

Three-Dimensional motion estimation is used to describe the motion of an object in 3-D space, for example in computer vision applications for object tracking or in object-based video coding for object tracking and motion compensation.

### 3.2 Camera motions

The following definitions are extracted from [7]:

- **Track:** Horizontal translation.
- **Boom:** Vertical translation.
- **Dolly:** Translation in the direction of the optical camera axis.
- **Pan:** Turning around the vertical axis of the camera.
- **Tilt:** Turning around the horizontal axis of the camera.
- **Roll:** Rotation around the optical axis of the camera.
- **Zoom:** The camera changes its focal length.

These camera motions are shown in Fig. 2.

During this project, we essentially focused on pan (rotation), zoom (dolly and zoom) and translation (track and boom).

### 3.3 Global motion estimation (GME)

The motion field between two images can be described by a translation, a geometric transformation, an affine mapping, or a projective mapping. In the case of a camera

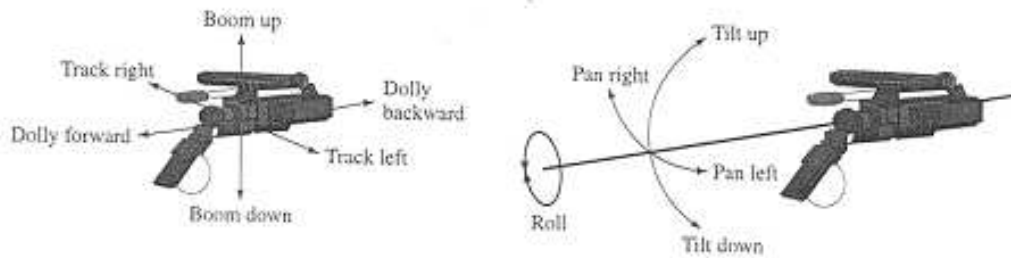


Figure 2: Typical types of camera motion.

motion, this model can be applied to the entire frame [7]. In general, there are at least two objects in a video sequence: a background and an object. But in the case of a small object motion, it is possible to well approximate the motion by a global model. For example, if the camera motion is fast we do not take into account the local motion. It is always useful to estimate camera motion.

There are two main global motion estimation approaches [7]: One consists of minimizing the prediction errors under a given set of motion parameters and then estimating global motion parameters. The other one consists of first finding pixel motion vectors and after determining the global motion model by regression methods.

### 3.4 Difficulties in global motion estimation

#### 3.4.1 Global/local motion:

Pixels in the frames may not have only global motion but both global and local motion vectors if the scene contains moving objects. Not all of the pixels of the same frame have the same global motion. To solve this problem, a “robust estimation” technique is used. Basically, pixels are separated into two groups: the inliers, which have a global motion and the outliers, the others. First the global motion is determined as if there were only inliers and then the prediction error is calculated for each pixel and a threshold is fixed, if the error is higher than this threshold this pixel is in the outlier

group. This process is iterated until there is no more outliers. This method is applied in the techniques that are used in our algorithms of GME.

### 3.4.2 Local changes:

Including:

- Local illumination changes.
- Object shadow.
- Light reflections.

### 3.4.3 Global changes:

Including:

- Global illumination changes.

## 3.5 Global motion compensation

Once global motion estimation is done, we can compensate or "remove" global motion (Fig. 1). Global motion compensation (GMC) is the process of "removing" unwanted global motion while preserving object motion. GMC is implemented by predicting a frame from the previous frame based on the estimated global motion [4]. GMC is an important tool in video processing:

- To improve picture quality for scenes with large global motion (video enhancement [4]). In contrast to scenes with arbitrary motion, the human eye is able to track detail in case of global motion. Thus GMC allows improving quality in many scenes.
- For video segmentation [4].
- For video coding [7] and [6].
- Video conferencing [6].

- Video surveillance [4].
- Video retrieval and database [4].

## 4 Modeling

Motions can be described either by a 2-D motion model or by a 3-D motion model. Two-Dimensional motion estimation is an important part of any video processing system. 2-D motion estimation is often the prerequisite step for 3-D structure and motion estimation. Three-Dimensional motion estimation is used to describe the motion of an object in 3-D space, for example in computer vision applications for object tracking or in object-based video coding for object tracking and motion compensation.

In many applications, we can represent the motions by using two-dimensional models. Figure 3 shows the projection of a moving object onto an image plane, and the relation between 3-D and 2-D motions.

When an object point is moved from  $X = x, y, z$  at time  $t_1$  to  $X' = (x + dx, y + dy, z + dz)$  at time  $t_2 = t_1 + dt$ , its projected image is moved from  $x = (x, y)$  to  $x' = (x', y') = (x + dx, y + dy)$ . The 3-D motion vector at  $X$  is the 3-D displacement  $d(X, t_1, t_2) = (X' - X)$  and the 2-D motion vector at  $x$  is the 2-D displacement  $d(x, t_1, t_2) = (x' - x)$ .

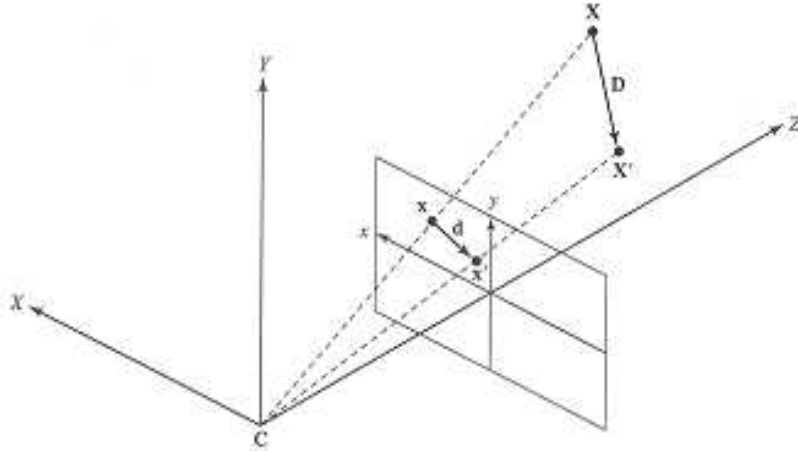


Figure 3: Projection of a moving object.

The motion field from  $t_1$  to  $t_2$  is represented by  $d(x, t_1, t_2)$  of all image positions  $x$  at time  $t_1$ . We can draw the motion field by a vector graph, as shown in figure 4, where the direction and magnitude of each arrow correspond to the direction and

magnitude of the motion vector at the pixel where the arrow takes its origin.

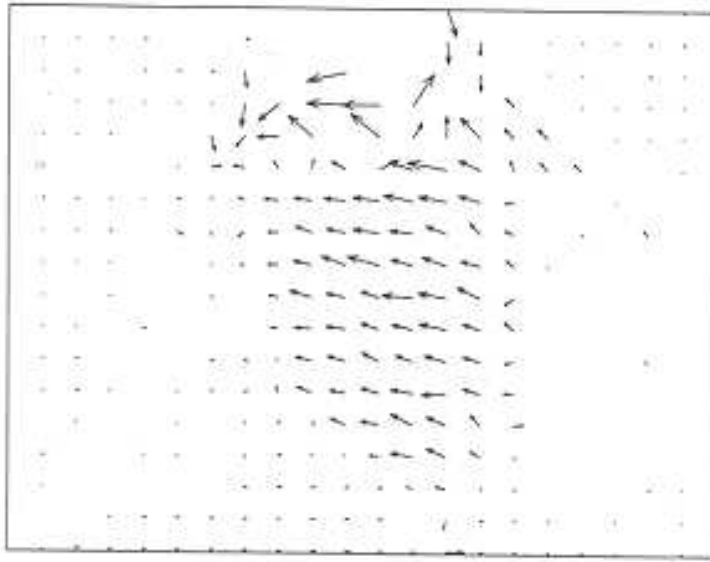


Figure 4: A typical 2-D motion field.

The motion fields corresponding to a camera zoom and camera rotation are represented on figure 5.

Global motion can be estimated using different models such as translation, geometric transformation, affine mapping, or projective mapping (see section 2.3). One of the most frequently used model is the 6-parameter affine model. To describe the global motion field, a 6-parameter affine motion model is used in our simulations:

$$x' = a_0 + a_1 \times x + a_2 \times y$$

$$y' = a_3 + a_4 \times x + a_5 \times y$$

where  $[x, y]$  is the position of a pixel in the current image,  $[x', y']$  is the position of the corresponding pixel in the reference image, and  $[a_0, a_1, a_2, a_3, a_4, a_5]$  are the global motion parameters.

Basically,  $a_0$  and  $a_3$  describe translation while  $a_2$ ,  $a_1$ ,  $a_4$  and  $a_5$  describe rotation and zooming. This affine model can deal with the majority of motion types encountered in video processing applications.

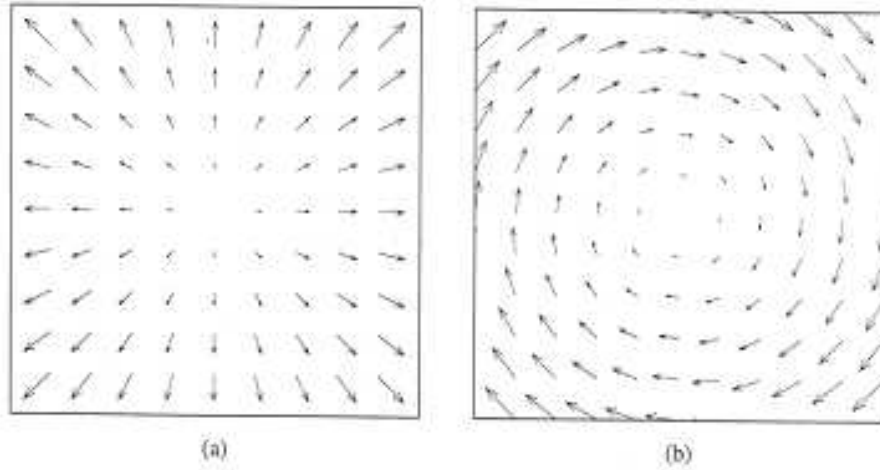


Figure 5: Motion fields corresponding to (a) camera zoom (b) camera rotation.

Typically the 6 parameters are initialized and then affined by different methods that we are going to describe.

We will see that generally first the translation parameters are determined and then from these parameters we affine to get the other ones and to find more accurate translation parameters.

Typically to get the parameters, we minimize either the sum of square differences (SSD) or the sum of absolute differences (SAD) between the current frame  $I$  and the motion compensated previous frame  $I'$ .

$$SSD = \sum_{x,y \in R} (I(x,y) - I'(x',y'))^2$$

$$SAD = \sum_{x,y \in R} |I(x,y) - I'(x',y')|$$

## 5 Solution approaches

In our study, we evaluated three solution approaches to GME. Two approaches are based on gradient descent and hierarchical implementation. The third approach is different in the sense it uses features of the image for estimation.

### 5.1 GME technique using hierarchical implementation and gradient descent (Algo. 1)

In this algorithm [5], called “Efficient, robust, and fast global motion estimation for video coding”, we propose an efficient method for global motion estimation from image sequences. The method is generic in that it can accommodate various global motion models, from simple translation to six-parameter affine model (see section 3). This GME technique is designed to minimize the sum of squared differences (SSD) between the current frame and the motion compensated previous frame.

This algorithm consists of three stages (see Fig. 7 and Fig. 6). First a three-level low-pass image pyramid is built in order to improve convergence and reduce computational complexity. Low-pass filters and downsampling are used to build this pyramid. This technique is called “a three-level hierarchical implementation”.

Then, an initial translation is estimated with full-pixel precision at the top of the pyramid. This initial estimation is obtained by a matching technique, which minimizes the SAD by using a modified n-step search [8].

In the third stage, a gradient descent(explained below) is executed at each level of the pyramid starting from the initial translation at the coarse level. The motion parameters are computed by minimizing the SSD. Since the dependence of SSD on motion parameters is nonlinear, we use the following iterative procedure:

$$a^{(t+1)} = a^{(t)} + H^{(-1)}b$$

where:

- $a$  is the motion parameter at  $t$  and  $t + 1$  iterations respectively.

- $H$  is a  $n \times n$  matrix equal to one-half times the Hessian matrix of SSD and is usually referred to as the curvature matrix.
- $b$  is a  $n$ -element vector equal to minus one-half times the gradient of SSD.
- $n$  is the number of parameters of the model (in this study six).

The parameters  $a_0$  and  $a_1$  are first initialized with the translation vector estimated in the initial matching stage. The gradient descent starts at the top level of the pyramid, then follows at the subsequent levels in a top-down approach. At each level, the gradient descent is iterated until a suitable convergence criterion is met. Each iteration of the gradient descent consists of the following steps:

- 1) matrix  $H$  and vector  $b$  are computed.
- 2) the system defined above is solved using Singular Value Decomposition (SVD) [9]. This method calculates  $H^{(-1)}b$ .
- 3) resulting update terms  $a$  is added to the current set of parameters:

$$a^{(t+1)} = a^{(t)} + \delta a$$

where  $\delta a = H^{(-1)}b$

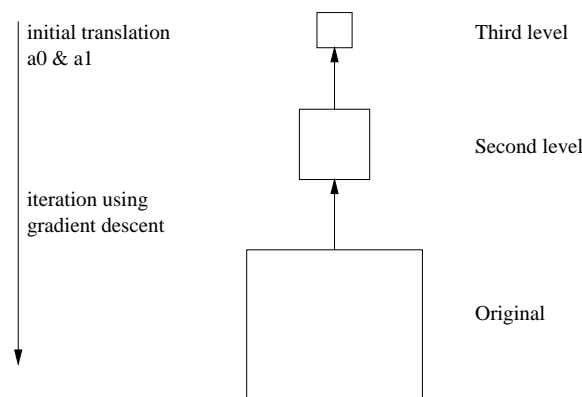


Figure 6: Hierarchical estimation.

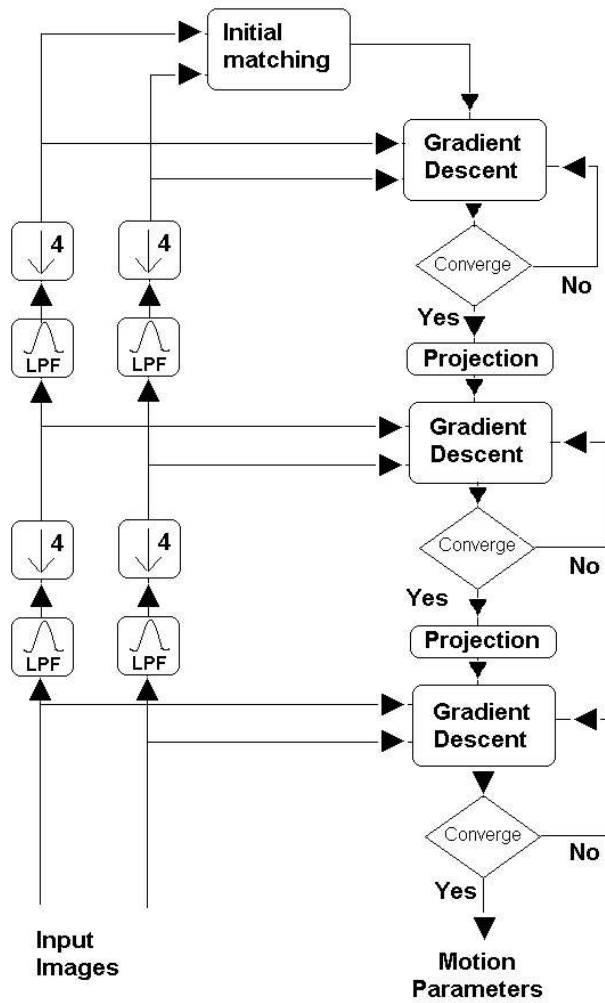


Figure 7: Block diagram for the case of a three-level hierarchical implementation

## 5.2 Improved GME using earlier prediction (Algo. 2)

The second proposed approach is an algorithm called “Improved Global Motion Estimation Using Prediction and Early Termination” [1].

This algorithm is very closed to the first one but uses prediction and early termination of the iterative process. In this approach we keep the 6-parameters affine model :

$$x' = a_0 + a_1 \times x + a_2 \times y$$

$$y' = a_3 + a_4 \times x + a_5 \times y$$

as defined in section 3. The goal is here to minimize the sum of squared intensity error between the current frame  $I$  and the reference frame  $I'$  :

$$E = \sum_{x,y \in R} (I(x,y) - I'(x',y'))^2$$

This is similar to the first method (described in the section 4.1), a three-level hierarchical implementation, with the initial stage at the top of the pyramid and a gradient descent at each level (Fig. 8) to finally get our global motion parameters.

Nevertheless, the first approach proposed a method to find the initial translation parameters  $(a_0, a_3)$ , but this method provides no rotation or zooming motion. Thus after the initial part:  $a_1 = a_5 = 1$  and  $a_2 = a_4 = 0$  .

The first improvement proposed is to use the previous motion parameters calculated as Motion Parameter Vector Prediction. Since global motion is mostly camera motion, we assume continuous camera motion over time. Then different predictors are used: the past motion vector, the acceleration motion vector which is a function of the two previous frame motion vector, a historic maximum motion vector based on the five previous frames, a historic minimum motion vector based on the five previous frames, a long time average motion vector and a zero motion vector.

The aim is to determine the best predictor according to the calculation of the SAD. To implement that in an efficient way, we first assume that  $a_1 = a_5 = 1$  and  $a_2 = a_4 = 0$ , and we find the best SAD with the translation parameters of all motion

vectors (MV). Once  $(a_0, a_3)$  are fixed then the SAD with the other parameters are calculated. In this way the complexity is less than if the predictors are calculated all at once.

At the end of the initialization, our predictor is the center of the gradient descent at the top level of the pyramid (Fig. 8).

A second change is a new termination criteria in the gradient descent but it can be done at the price of getting less accurate global motion parameters, indeed the process is stopped when the reduction in the SSD is less than a threshold.

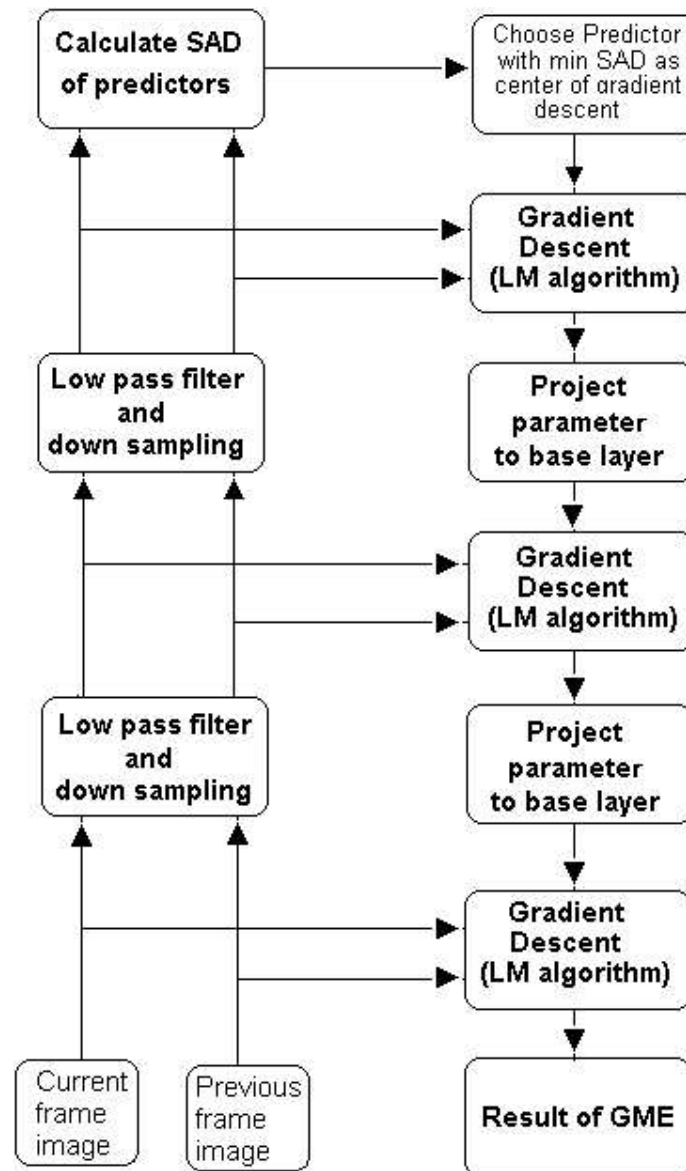


Figure 8: Block diagram of the second algorithm extracted from [1].

### 5.3 Real Time GME using feature tracking (Algo.3)

We tested another approach described in the paper untitled "Real Time Global Motion Estimation from MPEG-4 video encoder" [2].

To estimate global motion, the algorithm combine "a rough, wide-range initial step followed by a differential affine estimation". The algorithm is described on the Fig. 9 where  $I_0$  is the current image,  $I_1$  is the reference image,  $I_P$  is the predicted image and  $E_T, E_A$  are the motion parameters (translation, affine).

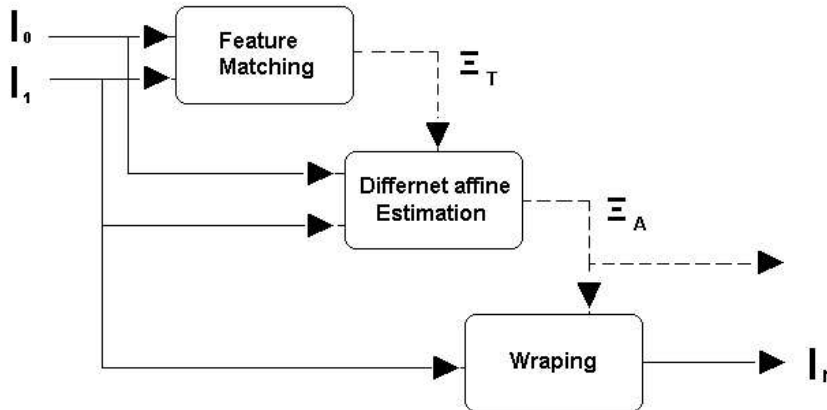


Figure 9: Block diagram of the proposed algorithm extracted from [2].

In this study, we will only implement the first step: "feature matching and initial estimation".

#### 5.3.1 Feature matching and initial estimation

First, a set of features is selected. Then a set of motion vectors is obtained by tracking these features. The feature selection process and the tracking process were defined by Shi and Tomasi [10]. A feature is a small window containing an image part that can be reliably tracked by a block matching algorithm.

In fact to have a reliable tracking of features, we must have a well structured image areas. Therefore before selecting the features, we filter one of the two images ( $I_1$ ) involved in each estimation by a Laplace Operator (an edge detection procedure)

with its FIR filter coefficients  $[1, -2, 1]$  in both directions. Then to track the features, a full search block matching is applied.

After that it is necessary to eliminate disturbing influences caused by differently moving foreground objects. To do that a robust M-estimator is used [11]. This estimator, also called maximum-likelihood operator, is applied to distinguish between reliable measures and outliers. The tracking process allows to get the displacement in  $x$  and in  $y$  of each feature, respectively  $v_x^{(m)}$  and  $v_y^{(m)}$ . We initialize the translation parameters  $a_0$  (in  $x$ ) and  $a_3$  (in  $y$ ) as the averages of respectively the  $v_x$  and of the  $v_y$  on all the features.

Then we calculate the difference  $\epsilon^{(m)}$  between the mean translation ( $a_0$  and  $a_3$ ) and each individual measure ( $v_x^{(m)}, v_y^{(m)}$ ):

$$\epsilon^{(m)} = |v_x^{(m)} - a_0| + |v_y^{(m)} - a_3| \quad (2.1)$$

Then the mean error is:

$$\mu_\epsilon = \frac{1}{M} \sum_{m \in M} \epsilon^{(m)} \quad (2.2)$$

with  $M$  = number of tracked features.

After that we affect some weighting functions for each individual measure (Tukey's Biweight):

$$\omega(\epsilon) = \begin{cases} (1 - (\frac{\epsilon}{c\mu_\epsilon})^2)^2 & \epsilon < c\mu_\epsilon \\ 0 & \epsilon \geq c\mu_\epsilon \end{cases} \quad (2.3)$$

Then we affine the values of  $a_0$  and  $a_3$ :

$$\begin{pmatrix} a_0 \\ a_3 \end{pmatrix} = \frac{1}{\sum_m \omega^{(m)}} \begin{pmatrix} \sum_m \omega^{(m)} v_x^{(m)} \\ \sum_m \omega^{(m)} v_y^{(m)} \end{pmatrix} \quad (2.4)$$

This algorithm has a fast convergence. Thus typically around 5-7 iterations are required.

### 5.3.2 Differential affine estimation

After predicting the global translation a higher order refinement takes place based on the SAD. In fact we calculate for each pixel defined by its position  $p = [x, y]$ :

$$\epsilon(x, y) = I_1(x, y) - I_0(x', y') \quad (2.5)$$

with:

$$x' = a_0 + a_1 \times x + a_2 \times y$$

$$y' = a_3 + a_4 \times x + a_5 \times y$$

where  $[x, y]$  is the position of a pixel in the current image,  $[x', y']$  is the position of the corresponding pixel in the reference image, and  $[a_0, a_1, a_2, a_3, a_4, a_5]$  are the global motion parameters.

Then we apply the M-Estimator approach [11] in which a function  $\rho$  of the residuals with a scale  $\mu$  is used to minimize  $\epsilon$ :

$$\min \sum_{p \in R} \rho(\epsilon(p), \mu) \quad (2.6)$$

To solve that an iterative Gauss-Newton minimization is applied.

The weighting function of each point contributing to the estimation can be expressed as  $\omega(\epsilon) = (d\rho/d\epsilon)/\epsilon$ . Nevertheless, we can simplify this expression by reducing it to a single binary decision:

$$\omega(\epsilon) = \begin{cases} 1 & \epsilon^2 < \mu \\ 0 & \epsilon^2 \geq \mu \end{cases} \quad (2.7)$$

with  $\mu$  the mean square of all the  $N$  considered points within the region  $R$ . However, to reduce the influence of unstructured areas, we do a priori an evaluation of the spatial gradients  $(I_x, I_y)$  and use only those pixels ( $\nu$ ) in the estimation which satisfy:

$$\nu = \begin{cases} 0 & (|I_x| < d) \cup (|I_y| < d) \\ 1 & \text{otherwise} \end{cases} \quad (2.8)$$

A threshold  $d$  between 10 and 20 leaves only those pixels which lead to a fast convergence of the Gauss-Newton algorithm.

### 5.3.3 Wrapping

According to our understanding of the algorithm, the wrapping consists of global motion compensation (see section 2.5).

## 6 Results

### 6.1 Implementation

To analyze our results and to compare the algorithms, we did a compensation module and a difference module which calculate and represent the absolute difference between the current image and the previous compensated image.

In all our tests, we used a sun4u, SunOS Release 5.7 Version *Generic106541 – 15 [UNIX(R)SystemVRelease4.0]*

We tested the three algorithms on different sequences:

- “Car” which include mainly a camera translation to the left. Number of frames: 20 (Fig.10).



Figure 10: Car sequence: Frame n.15

- “Pr1car” which include a camera translation to the right but with another background. Number of frames: 20 (Fig.11).
- “Tennis” which include a zoom motion. Number of frames: 10 (Fig.12).
- “Atp” which include a sequence with almost no global motion but a lot of different local motions. Number of frames: 20 (Fig.13).



Figure 11: Prlcar sequence: Frame n.12



Figure 12: Tennis sequence: Frame n.4

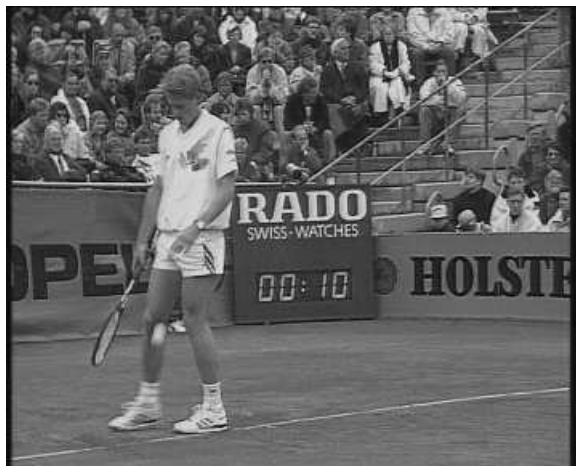


Figure 13: Atp sequence: Frame n.5

## 6.2 GME Technique using hierarchical implementation and gradient descent (Algo.1)

### 6.2.1 Determination of the parameters

In this first algorithm, there are different parameters that we can change. Firstly, concerning the initial matching which, the sum of absolute difference is computed for 25 search points in the top level of the image pyramid. We put a threshold to exclude the part of the points where the error is too big. In our case we fixed this threshold to 12. Moreover, the same threshold is used in the gradient descent and we fixed it to 15 after several tries.

For instance if we fixed the threshold to 5 we get the difference with a many errors where it is impossible to recognize anything. Concerning the gradient there are also a maximum number of iterations (here 20) and two other thresholds : a threshold  $\epsilon_1$  for the update of the translation parameters and a threshold  $\epsilon_2$  for the update of the remaining parameters. We fixed  $\epsilon_1 = 0.1$  and  $\epsilon_2 = 0.001$  after tries.

For the rest of our tests we will keep this values.

### 6.2.2 Results

- **Car** : We find for this sequence  $a_0$  changes from 9.1 to 6.6 while  $a_3$  oscillates between  $-0.5$  and  $1$ . the other motions are neglected.

We can see here in Fig. 14 very clearly the edges of the car which has a different local motion compared to the global motion.

- **Prlcar** : Here we get, for  $a_0$  values between  $-13$  and  $-16.5$  throughout 20 images and for  $a_1$  between  $-0.6$  to  $0.6$

Here in Fig. 15 we can notice that even if the shape of the car is clear, there are more errors in the background due to the details contained in the background and to the motion. We will see in the section 5.5 that the error is larger than for the first sequence (“car”) as expected because of the background.



(a) Previous frame.



(b) Current frame.



(c) Compensated previous frame.



(d) Difference frame.

Figure 14: Simulations for Car frame n.15.



(a) Previous frame.



(b) Current frame.



(c) Compensated previous frame.



(d) Difference frame.

Figure 15: Simulations for Prlcar frame n.12.



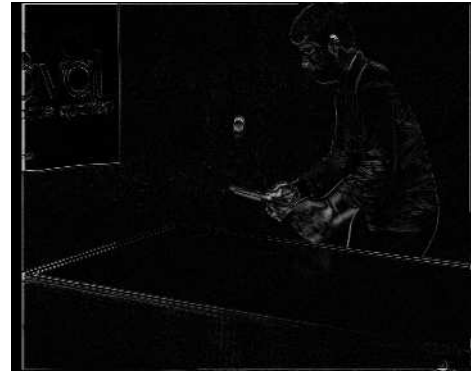
(a) Previous frame.



(b) Current frame.



(c) Compensated previous frame.



(d) Difference frame.

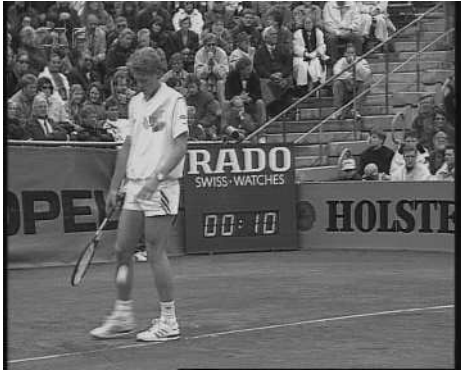
Figure 16: Simulations for Tennis frame n.4.

- **Tennis** : here the values for the parameters are about 4 for  $a_0$ , 1.5 for  $a_3$ , 0.9 for  $a_1$  and  $a_5$  and 1 for  $a_2$  and  $a_4$ .

We can see in Fig. 16 the ball which has a local motion. Moreover in the compensated sequence we get a zoom as expected. For this sequence we get very good results and in the difference sequence we can see the ball which has local motion but because of the luminance and the printing it is not very clear on the figure.

- **Atp**

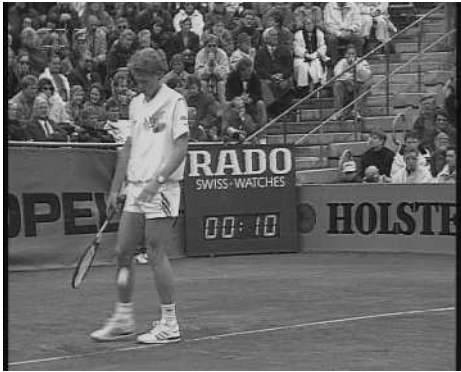
Concerning this sequence the results are quite bad because there are too many



(a) Previous frame.



(b) Current frame.



(c) Compensated previous frame.



(d) Difference frame.

Figure 17: Simulations for Atp frame n.5.

local motions. In Fig. 17 which is a difference image between a previous compensated image and the current image, there are many details that appear.

## 6.3 Improved GME Technique using earlier prediction (Algo.2)

### 6.3.1 Implementation and Determination of the parameters

In the second algorithm, which is an improvement of the first one, there are still the same thresholds and another parameter that corresponds to the predictors historic. In the implementation we choose to take into account the seven last motion vectors or the five last motion vectors. In fact because of the accuracy of the results, seven and five last motion vectors have almost the same results in terms of errors so we took the five last motion parameters (gain in complexity).

Knowing that we need at least five motion parameters at the beginning of the sequence, thus we applied the first algorithm to determine the five first motion parameters. We also changed the threshold  $\epsilon_2$  to 0.05 even if we loose a little bit in accuracy, as described in the algorithm (section 4.2).

### 6.3.2 Results

In fact for this algorithm the visual results are very closed to the first one, so we do not show the frames. Nevertheless, we can look at the results for the prlcar sequence (Fig. 18).

We can notice in Fig. 18 that the car is very clear but in the background we see some errors in the windows. In fact when we look at the compensated frame we see that the background is at the same place as in the current frame but there is a problem with the car. That is normal because we compensate only the global motion (here the background) and not the local motion (here the car) so that is why the car in the compensated frame is not at the good place.



(a) Previous frame.



(b) Current frame.



(c) Compensated previous frame.



(d) Difference frame.

Figure 18: Simulation for Pr1 car sequence.(frame n.12)

## 6.4 Real time technique using feature tracking (Algo.3)

### 6.4.1 Determination of the parameters

The first stage of the algorithm is the feature selection process and the feature tracking. The tracker [12] is described in the paper written by Shi and Tomasi [10]. Briefly, good features are located by examining the minimum eigenvalue of each  $2 \times 2$  gradient matrix, and features are tracked using a Newton-Raphson method of minimizing the difference between the two windows. Multiresolution tracking allows for even large displacements between images.

In our simulations, we have the possibility to change the number of selected features, this number  $N$  is one of the parameters ( $N \leq 1001$ ). The three images in Fig. 19 show the selection of the features for  $N = 100, 500$  and  $1000$  (the features are marked by small rectangles).

These images come from the video sequence "car", on which we applied the selection process. We can notice that the features follow the edges included in the image. Indeed the feature selection process is based on calculating the gradients.

After the feature selection, the tracker writes a table with the locations in  $x$  and  $y$  of each feature and for each feature it gives a value of "goodness". Then it tracks these features from the first image to the second image and writes in the table the new locations and if the feature is successfully tracked, then its value becomes zero; otherwise, it becomes negative. In this case, the feature is replaced and its new locations and value are overwritten.

After the tracking, the estimation process is applied by taking into account the features which have a value equal to zero, i.e., the successfully tracked features between the previous and the current image.

Currently, only the estimation of the translation parameters  $(a_0, a_3)$  is implemented.

Besides the choice of the number of features has effects on the estimation. Indeed we can see, from the figure 20, the influence of this parameter. These images represent

(a)  $N = 100$ (b)  $N = 500$ (c)  $N = 1000$ 

Figure 19: Feature selection.

(a)  $N = 100$ (b)  $N = 500$ (c)  $N = 1000$ 

Figure 20: Difference image for different numbers of features.

the absolute difference between the previous compensated frame and the current frame.

As expected the less is the number of features, the worse is the estimation. Therefore some errors appear in the difference image (Fig. 20).

Thus to get a good estimation, we should keep a high number of features. Of course by increasing the number of features, the time of process is a little bit longer but a high number of features like 1000 ensures the validity of the estimation. The selection of 500 features gives also good results but by precaution we choose 1000 features.

In the following tests,  $N$  is equal to 1000.



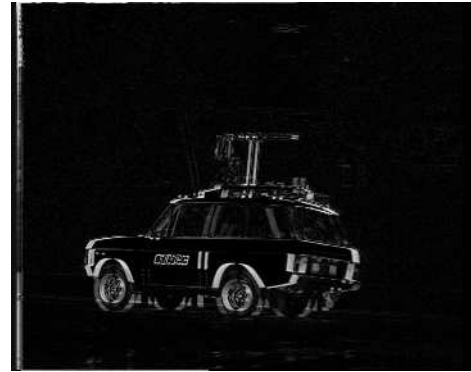
(a) Previous frame.



(b) Current frame.



(c) Compensated previous frame.



(d) Difference frame.

Figure 21: Simulation for Car sequence.(frame n.15)

The tuning constant used in the equation (2.3) is one of the parameters of our algorithm. This one is set to 1.0.

#### 6.4.2 Results

We tested our algorithm of translation global motion estimation on different video sequences with translation global motion.

- **Car:**

We find for this video sequence values between 8.44 and 6.71 for  $a_0$  and between  $-0.28$  and  $0.95$  for  $a_3$ . For this video sequence, we get subjectively a compensated sequence similar to the original. When we look at the error sequence, we



(a) Previous frame.



(b) Current frame.



(c) Compensated previous frame.



(d) Difference frame.

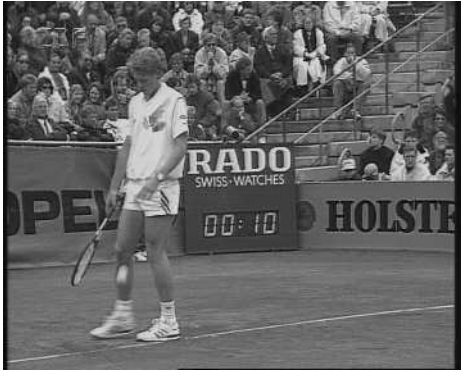
Figure 22: Simulation for Prlcar sequence.(frame n.12)

can distinguish the shape of the car (Fig. 21). This error is due to the motion of this object. The other errors are very small and may be partially linked with the little rotation motion in the sequence which is not estimated yet or more simply because the estimation cannot be perfect.

- **Prlcar:**

We find for this video sequence values between  $-16.17$  and  $-13.21$  for  $a_0$  and between  $-0.54$  and  $0.54$  for  $a_3$ .

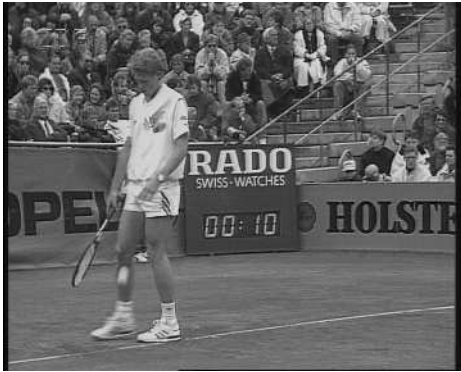
For this video sequence, we get a compensated sequence quite similar to the original. When we look at the error sequence (Fig. 22), we can distinctly see



(a) Previous frame.



(b) Current frame.



(c) Compensated previous frame.



(d) Difference frame.

Figure 23: Simulation for Atp sequence.(frame n.5)

the shape of the car and a little bit less the shape of the background. As previously, the shape of the car is due to the motion of the object. However, the edges of the background should not appear in the error sequence. That maybe partially caused by the rotation motion in the sequence. But in this case, several imperfections appear.

That can be explained by the fact that the background and the light are different than in the case of the car sequence.

- **ATP:**

In this case, a lot of errors appear in the error sequence (Fig. 23) (Note: The

print is not good). The estimation is not good for this kind of video. In this one, there are too much objects in the background.

## 6.5 Comparison

### 6.5.1 Criteria

For the comparison we use three main criteria

⇒ Mean Absolute Error (MAE):

For the error we work with the mean absolute difference defined as follow:

$$MAE = \frac{1}{M} \sum_{x,y \in R} |I(x,y) - I'(x',y')|$$

where:  $M$  is the number of considered pixels.  $R$  is the set of the image pixels.  $I$  is the current image and  $I'$  is the previous compensated image.

⇒ Robustness:

There is no unique definition of robustness in the literature. In general an algorithm is said to be robust when it gives similar estimation even if the input images are affected by outliers such as noise, local motion, background motion and local illumination changes.

⇒ Efficiency (speed):

To determine the speed efficiency we used the *clock* utility of the programming language C.

### 6.5.2 Results and analysis

⇒ MAE



(a) Algo 1.

(b) Algo 2.

(c) Algo 3.

Figure 24: The difference image for Car sequence.(frame n.15)

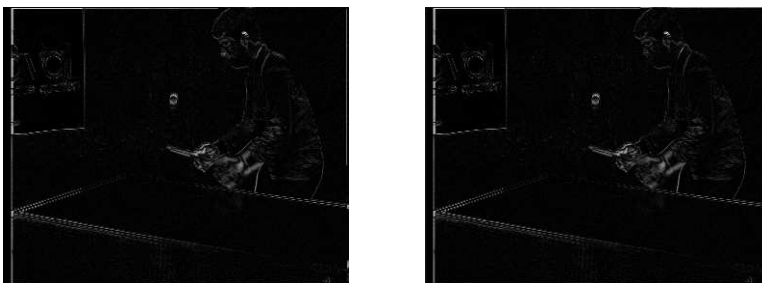


(a) Algo 1.

(b) Algo 2.

(c) Algo 3.

Figure 25: The difference image for Prlcar sequence.(frame n.12)



(a) Algo 1.

(b) Algo 2.

□Algo 3, not available.

Figure 26: The difference image for Tennis sequence.(frame n.4)



(a) Algo 1.

(b) Algo 2.

(c) Algo 3.

Figure 27: The difference image for Prlcar sequence.(frame n.5)

In figures Fig 24, Fig 25, Fig 26 and Fig 27, we see the difference images for all the sequences.

**Table 1**

<i>Error</i>	<b>Car</b>	<b>prlCar</b>	<b>Tennis</b>	<b>Atp</b>
<b>algo.1</b>	10.9	15.2	7.6	10.12
<b>algo.2</b>	10.9	15.3	8.7	10.2
<b>algo.3</b>	11.4	15.4	X	10.6

First of all, the value of the error gives only an idea of the level of difference between the previous compensated image and the current image. When the estimation is well done, this value is low. However we can have a small error but the estimation can be quite bad, particularly when there are a lot of objects in the image.

Indeed we can see from the Table 1, when the estimation is good (car sequence, any algorithm), the error is around 11. In the prlcar sequence, we saw that the estimation was not as good as in the Car sequence. Therefore, the error is higher (around 15, any algorithm). In this sequence we have background changes and that is why we can notice many errors in the background.

For the atp sequence, as we said before the error is low (around 10.5, any algorithm) but the estimation is not good when we subjectively view the difference sequence. This is caused by the high number of objects in the image. Indeed the least object motion in the background may cause an error in the estimation.

Besides if we compare the three algorithms for sequences in which the main global motion is a translation (prlcar and car), the results are quite similar. However the third one gives a little bit less good results for the car sequence (11.4 instead of 10.9). This difference may be caused by the small rotation motion in the sequence that is not estimated in the third algorithm yet and by

the fact that the translation parameters are not totally affined (the algorithm is not finished yet).

For the tennis sequence in which we have a zoom, the two first algorithms give a small error (around 8) which corresponds to a good estimation as we can see on the difference sequence. Nevertheless the second algorithm gives a higher error than the first one. This little difference was expected because the second algorithm increases the speed at the price of getting less accurate global motion parameters.

⇒ Robustness

Based on the definition of robustness (see section 5.5.1), the three algorithm seem to be robust. Besides the third algorithm was expected to be robust because it uses a *hard-threshold robust estimator* [7] in the differential affine estimation part, and a *soft-threshold robust estimator* [7] in the feature matching part. These estimators eliminate the outliers and keep only the inliers to make the estimation. The basic idea of a robust estimator is to consider the pixels that are governed by global motion as inliers, and the remaining pixels as outliers. However, more investigations are needed to confirm our observations specially concerning the noise.

⇒ Efficiency (clock)

**Table 2**

<i>Time(s/img)</i>	<b>Car</b>	<b>prlCar</b>	<b>Tennis</b>	<b>Atp</b>
<b>algo.1</b>	9.9	6.5	6.9	5.1
<b>algo.2</b>	3.6	2.2	3	1.9
<b>algo.3</b>	2.4	2.5	X	2.2

Concerning the speed of the algorithms, we can see on the Table 2, that as expected the second algorithm is much faster than the first one for any sequence

(2.2 s/img instead of 6.5 s/img for the prlcar sequence). We have a gain in time of around 2.5 between the two first algorithms. The third algorithm seems to be very fast and as fast as the second algorithm or maybe faster (2.4s/img instead of 3.6s/img for the car sequence) but right now we have only the estimation of the translation parameters for the third algorithm. Therefore we cannot affirm that the last algorithm is faster than the second one.

## 7 Conclusion and future work

In this report, we presented the results of the implementation of three different global motion estimation methods. Currently we can compare these three algorithms only from global translation motion because the third algorithm is not fully implemented yet. However, we can say that the second algorithm gives a similar results as the first algorithm but faster for any kind of sequences. The third algorithm seems to give less good results (MAE) than the others but seems to be a little bit faster than the second algorithm. The use of feature tracking seems to be a good idea because in video surveillance we need a fast estimation and a “perfect” estimation is not expected.

Even if the error is quite high for some sequences, when we look at the difference image we can easily recognize objects which have local motion. Indeed, our tool may be used as an initial step for video object segmentation.

However, the third algorithm have to be completed specially that the first results that were obtained for the translation motion were good and very fast. Another part which can be very interesting to study now is the object segmentation part, after estimating the global motion and compensating it.

## A Algorithm modules

### A.1 GME technique using hierarchical implementation and gradient descent

For the first there is one main function :

#### - GlobMotAff()

This function calculates the 6 affine global motion parameters and inside, it builds the low pass image pyramid, then estimates the initial translation parameters, and then executes the gradient descent to get the global motion parameters.

- **inputs:** previous frame, current frame, four thresholds, the maximum number of iterations.
- **output:** an array with the 6 affine motion parameters.

#### Parameters in the pcf file

<b>Input File</b>	- A video sequence
<b>Output Files</b>	- the compensated sequence - the difference sequence
<b>Numerical parameters</b>	- Number of the first image - Number of the last image - Threshold 1(as explained in section 5.2.1) - Threshold 2(as explained in section 5.2.1) - $\epsilon_1$ (as explained in section 5.2.1) - $\epsilon_2$ (as explained in section 5.2.1) - Maximum number of iteration (in the gradient descent)

#### - Vector()

This function represents the translation motion (horizontal and vertical) by arrows.

- **inputs:** Previous frame, translation parameters.
- **output:** An image with arrows to represent the translation motion (horizontal and vertical).

## A.2 Improved GME Technique using earlier prediction

Here, there are different functions because for the first five frames we calculate the parameters with the first algorithm, and then we use the improvement algorithm:

### - GlobMotAff2()

This function calculates the 6 affine global motion parameters and inside it builds the low pass image pyramid, then estimates the initial translation parameters, and then executes the gradient descent to get the global motion parameters.

- **inputs:** previous frame, current frame, four thresholds, the maximum number of iterations and the number of the frame.
- **output:** a matrix with the 6 affine motion parameters of all the previous frames plus the new calculated parameters.

### - Predict\_Motion()

This function does the same operations as GlobMotAff2 but with a new initial part based on predictors parameters. We use it from the 6th frame to the last one.

- **inputs:** previous frame, current frame, the motion parameters predictors (past MV, acceleration MV, zero MV, long time average MV) and the number of the frame.
- **output:** a matrix with the 6 affine motion parameters of all the previous frames plus the new calculated parameters.

**Note that for the first and the second algorithm the pcf file is the same.**

### A.3 Real Time GME using feature tracking

In the estimation of the global translation parameters described in the third algorithm, we can define two distinct parts: the feature tracking and the estimation.

#### a) The feature tracking process:

The feature tracking is realized by the following main functions:

##### - **KLSelectGoodFeatures()**

It finds the N best features in an image and stores them in a feature list in descending order of goodness. N is the size of the feature list.

- \* **inputs:** tracking context, values of the pixels of an image, number of columns and number of rows.
- \* **output:** list of the selected features with locations in x, in y and value of goodness.

##### - **KLTrackFeatures()**

It tracks the features from the first image to the second image and writes the feature in a list. If a feature is successfully tracked, then its value becomes zero; otherwise, it becomes negative and it is lost.

- \* **inputs:** tracking context, values of the pixels of the previous and the current image, number of columns and number of rows.
- \* **output:** list of the selected features with new locations in x, in y and new value of goodness.

##### - **KLReplaceLostFeatures()**

It calls the same underlying computation as **KLSelectGoodFeatures()** to find all the features in the image and rank them accordingly. Then, if k features have been lost, the k best features are used to replace them.

##### - **KLStoreFeatureList()**

It stores a feature list as the  $i$ th column of a feature table, where  $i$  is given by the third parameter.

- \* **inputs:** a feature list, the number of frame.
- \* **input-output:** a feature table.

#### b) The estimation process:

The estimation is realized by the main function:

##### - **estim()**

It estimates the global translation parameters between the current image and the previous image by applying the algorithm.

- \* **inputs:** the feature table, the number of features, the tuning constant and the number of the previous frame
- \* **output:** the array of parameters with the new values of  $a_0$  and  $a_3$

#### Parameters in the pcf file

<b>Input File</b>	- A video sequence
<b>Output Files</b>	- the compensated sequence - the compensated sequence
<b>Numerical parameters</b>	- Number of the first image - Number of the last image - Window size for the feature tracking (x-direct) - Window size for the feature tracking (y-direct) - Tuning constant

### A.4 Compensation and error modules

#### - **M\_Compensated\_Pic()**

It calculates the compensated image from the previous image and the global motion parameters.

- **inputs:** The previous image and the global motion parameters.
- **output:** The compensated image.

#### - Accuracy()

It calculates the difference image between the current image and the previous compensated image and calculates the mean absolute difference too.

- **inputs:** The previous compensated image and the current image.
- **outputs:** The difference image and the mean absolute difference.

#### - M\_Compensated\_Pic2()

It calculates the compensated image from the previous image and the global motion parameters for the second algorithm. The output is a matrix with the 6 parameters for all the frames.

- **inputs:** The previous image and the global motion parameters.
- **output:** The compensated image.

## B Implementing modules and functions

### • Algorithm 1

- **Low Pass Filter:** Implemented and tested.
- **Downsampler:** Implemented and tested.
- **Initial matching:** Based on an existing code, we adapted it and changed it to apply on our algorithm.
- **Gradient descent:**
  - \* Calculation of the different matrices Based on a code, we understood and adapted it to apply on our algorithm.

- \* SVD function Based on a library from internet, but we had to change all the interface to test it and to apply it on our algorithm.

- **Algorithm 2**

- **Low Pass Filter and Downsampler** Same functions as in the first algorithm.
- **Calculation of Predictors** Implemented and tested.
- **Gradient descent** Same functions as in the first algorithm.

- **Algorithm 3**

- **Laplace operator:** Implemented and tested.
- **Feature matching and initial estimation:** We implemented and tested the different mathematical equations involved in this part.
- **Feature selection and tracking:** Code got from internet. We understood it, compiled it, tested it and inserted it in our program (adaptation of the inputs and outputs).
- **Differential affine estimation:** We implemented and tested the equation (2.5) (difference image).

- **Compensation**

- **Compensation program:** Implemented and tested.

- **Difference image**

- **Difference image program:** Implemented and tested.

- **MAE**

- **MAE program:** Implemented and tested.

- **Arrows**

- **Arrows program:** Implemented and tested.

## References

- [1] M. F. F. Wing Cheong Chan, Oscar C. AU, “Improved global motion estimation using prediction and early termination,” *IEEE ICIP*, vol. II, pp. 285–288, January 2002.
- [2] B. S. H. Richter, A. Smolic and E. Muller, “Real time global motion estimation for an mpeg-4 video encoder,” *Proc. PCS’2001, Picture Coding Symposium*, Apr. 2001.
- [3] H.-M. Hang and J. W. Woods, *Handbook of visual communication*. Academic Press, 1995.
- [4] A. Amer, *Object and Event Extraction for Video Processing and Representation in On-Line Video Applications*. PhD thesis, INRS-Télécommunications, Univ. du Québec, Dec. 2001.
- [5] F. Dufaux and Janusz Konrad, “Efficient, robust and fast global motion estimation for video coding,” *IEEE Transactions on Image Processing*, vol. 9, pp. 497–501, Mar. 2000.
- [6] B. Furht, J. Greenberg, and R. Westwater, *Motion estimation algorithms for video compression*. Kluwer Academic Publishers, 1997.
- [7] Y. Wang, J. Ostermann, and Y.-Q. Zhang, *Video Processing and Communication*. Prentice Hall, 2002.
- [8] A. H. T. Koga, K. Linuma and Y. Lijima, “Motion compensated interframe coding of video conferencing,” *proc. Nat. Telecommun. Conf.*, pp. G5.3.1–G5.3.5, Dec. 1981.
- [9] S. T. W.H. Press, B.P. Flannery and W. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge Univ. Press, 1988.

- [10] J. Shi and C. Tomasi., “Good features to track,” in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 593–600, July 1994.
- [11] A. Smolic and J.-R. Ohm, “Robust global motion estimation using a simplified m-estimator approach,” *Proc. ICIP2000, IEEE international Conf. on image processing*, Sept. 2000.
- [12] vision.stanford.edu, “Klt: An implementation of the kanade-lucas-tomasi.” <http://vision.stanford.edu/birch/klt/index.html>, 1998.