Fast gradient methods based global motion estimation for video compression

A. Averbuch[1], Y. Keller[1,2]

[1]School of Computer Science, Tel Aviv University, Tel Aviv 69978 Israel

[2]Dept. of Electrical Engineering Systems
Tel-Aviv University
Tel-Aviv 69978, Israel

Abstract

This paper presents a fast global motion estimation (GME) algorithm based on gradient methods (GM), which can be used for real-time applications, such as in MPEG4 video compression. This approach improves the existing state-of-the-art GME algorithms by introducing two major modifications: first, only a small subset (down-to 3%) of the original image pixels is used in the estimation process. Second, a warp-free formulation of the basic GM is derived, further decreasing the computational complexity. Experimental results show no loss of GME accuracy and compression efficiency compared to the MPEG-4 verification model, while reducing the computation complexity of the GME by a factor of 20.

Keywords: Global motion estimation, Gradient methods, MPEG-4, sprites, video coding
EDICS Category={2-ANAL, 2-MOTD}

1 Introduction

MPEG-4 is a new video compression standard providing core technologies for efficient storage, transmission and manipulation of video data in multimedia environments [11, 20]. Motion

estimation algorithms calculate the motion between successive video frames and predict the current frame from previously transmitted frames using the motion information [7]. Global motion estimation (GME) algorithms estimate a single parametric motion model [9] for the whole frame which can be used within MPEG-4 to produce either *static* or *dynamic sprites*. Static sprites [17] are mosaics containing the visual information of the objects which were visible over the sequence. While various mosaic generation algorithms were developed [3, 4, 6, 14], their applicability to general purpose video compression applications is limited by the significant delay incurred by frames accumulation and mosaic image coding (as intra frames) [18]. Furthermore, the 8-parameters projective motion model used by the MPEG-4 coding standard is suitable for a restricted range of camera motions [4]. Thus, each static sprite can be only used for a single short video segment. Therefore, this paper concentrates on dynamic sprites while its results are also applicable to static sprites. The dynamic sprite [21] coding scheme utilized by the MPEG-4 verification model, estimates the motion between consecutive frames using a 6-parameters affine motion model. A sprite is generated every time step by warping the previous frame according to the motion parameters and used as a reference frame [19]. Further improvement is achieved by first estimating both the global and local motions (using block matching) and then coding each macro-block using the motion estimation mode which results in a lower prediction error [15, 16].

A comprehensive comparative survey by Barron et. al. [1] found the family of gradient-based motion estimation methods (GM), originally proposed by Horn and Schunck [2], to perform especially well. The purpose of the GM algorithm is to estimate the parameters vector $\underline{P}$ associated with the *parametric image registration* problem [3]. A critical implementation issue concerning the GME, is its significant computational complexity, making it useless for real-time encoding application, especially when implemented on low-power devices such as PDAs and cellular phones. This paper offers two modifications to the GM algorithm, reducing its computational complexity by 20 times. First, only a small, selective sub-set of the image pixels named *Dominant Pixels,* is used by the GM algorithm. Second, the warp-free formulation of the GM algorithm (WFGM) [13] allows for further complexity reduction. These two algorithms are complexity-wise complementary: each of them reduces the complexity of a different component within the original GM algorithm. Experimental results demonstrate the significant complexity reduction while maintaining the GME accuracy and video compression efficiency.

The regular GM based GME algorithm is presented in section 2, while the Selective integration based GM (SIGM) and Warp-free GM (WFGM) are introduced in sections 3 and 4 respectively. The resulting algorithm, Fast GM, is presented in section 5, while experimental results are given in section 6.

2 Gradient method based motion estimation

GM methodology [8, 10] estimates the motion parameters $\underline{P}$ by minimizing the intensity discrepancies between input images $I_1(x,y)$ and $I_2(x,y)$

$$\underline{P}^* = \arg\min_{\underline{P}} \sum_{\left(x_i^{(1)},y_i^{(1)}\right)\in S} \left(I_1\left(x_i^{(1)},y_i^{(1)}\right) - I_2\left(\underline{P},x_i^{(2)},y_i^{(2)}\right)\right)^2 \tag{2.1}$$

where

$(x_i^{(1)},y_i^{(1)})$, $(x_i^{(2)},y_i^{(2)})$	the coordinates of the ith pixel common to I_1 and I_2, respectively
S	the set of coordinates of pixels that are common to I_1 and I_2 in I_1's coordinates.
$\underline{P}$	the estimated motion parameters vector.

The solution of Eq. (2.1) is based on a pixel-wise first order Taylor expansion of I_1 in terms of I_2 as a function of the parameters vector $\underline{P}$

$$I_1(x_i^{(1)},y_i^{(1)}) = I_2(x_i^{(2)},y_i^{(2)}) + \sum_{Pj\in\underline{P}} \frac{\partial I_2(x_i^{(2)},y_i^{(2)})}{\partial P_j} P_j \tag{2.2}$$

Eq. (2.2) can be considered the linearization step of a Gauss-Newton nonlinear minimization [22] of Eq. (2.1). By gathering the pixel-wise equations similar to Eq. (2.2), an equation set is formed and solved for $\underline{P}$ [3, 5].

$$\underline{P} = \left(\underline{\underline{H}}^t \underline{\underline{H}}\right)^{-1} \underline{\underline{H}}^t \underline{I_t} \tag{2.3}$$

where

$$\underline{I_t} = \left((I_t)_1 \dots (I_t)_n\right)^T \tag{2.4}$$

and

$$(I_t)_i = I_1\left(x_i^{(1)},y_i^{(1)}\right) - I_2\left(x_i^{(2)},y_i^{(2)}\right) \tag{2.5}$$

The partial derivatives according to the motion parameters are calculated using the derivative chain rule

$$\begin{aligned} \underline{\underline{H}}_{i,j} &= \frac{\partial I_2\left(x_i^{(2)},y_i^{(2)}\right)}{\partial P_j} \\ &= \frac{\partial I_2\left(x_i^{(2)},y_i^{(2)}\right)}{\partial x_2}\frac{\partial x_i^{(2)}\left(x_i^{(1)},y_i^{(1)},\underline{P}\right)}{\partial P_j} + \frac{\partial I_2\left(x_i^{(2)},y_i^{(2)}\right)}{\partial y_2}\frac{\partial y_i^{(2)}\left(x_i^{(1)},y_i^{(1)},\underline{P}\right)}{\partial P_j} \end{aligned} \tag{2.6}$$

where $\left(x_i^{(2)}, y_i^{(2)}\right)$ are related to the parametric motion model used. For the affine motion model we get

$$\begin{aligned} x_i^{(2)} &= a \cdot x_i^{(1)} + b \cdot y_i^{(1)} + c \\ y_i^{(2)} &= d \cdot x_i^{(1)} + e \cdot y_i^{(1)} + f \\ \underline{P} &= (a,b,c,d,e,f) \end{aligned} \tag{2.7}$$

Due to the non-linear nature of Eq. (2.1), it is solved iteratively where Eq. (2.3). The basic GM iteration, which is marked as "*Single Iteration*" and iterative refinment phase are presented in Fig. 1.

In order to improve the convergence properties, a standard Gaussian pyramid [3] is constructed using scaling factors of 2 or 3 [6, 10]. Hence, the GM algorithm starts at the coarsest resolution scale of the pyramid, then follows the subsequent levels in a coarse-to-fine approach. At each resolution scale, Eq. (2.3) is iterated until a maximal number of iterations is reached or the magnitude of the update of translation parameters reaches a predetermined threshold. Finally, when the procedure stops at the finest resolution scale, the final motion parameters are obtained. In video compression applications the induced relative motion is usually small, therefore, less than 10 iterations are needed for an accurate registration and convergence. In situations where larger motion is anticipated, a bootstrapping procedure can be applied [9].

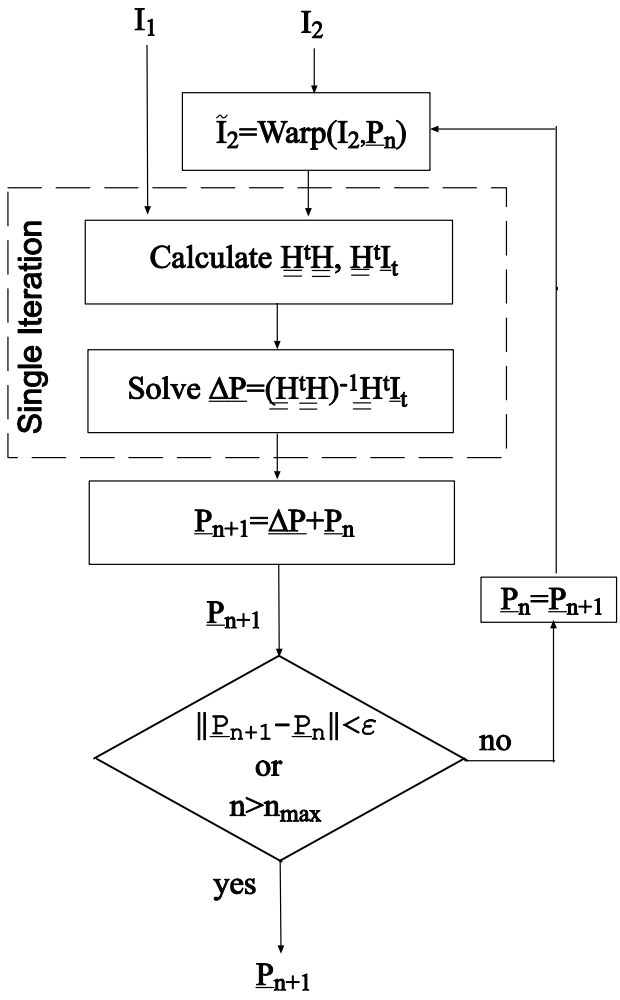

Figure 1: Block diagram of the basic and iterative GM formulations. For $n = 0$, $\underline{P}_0$ is given as an initial guess and $\underline{\Delta P}$ is the iterative update after each iteration.

3 Selective integration based GM (SIGM)

The evaluation of Eq. (2.3) at each GM iteration uses S, the complete set of pixels common to I_1 and I_2. Hence, the resulting equation is highly overdetermined. At QCIF frame size (176×144 pixels) there are 25,344 equations, as opposed to 6 or 8 unknown parameters related to affine and perspective motion models, respectively. The Selective integration based GM (SIGM) evaluates Eq. (2.3) using $\widehat{S}$, a small subset (down to 3%) of S. Following the GM scheme in section 2, the SIGM algorithm uses a multiresolution pyramid of the input images, where the pixel set $\widehat{S}$ is chosen at the coarsest resolution scale, and tracked using a

Figure 2: An example of the locations of the subset of pixels used for the global motion computation. The pixel set is superimposed on the first frame of the Stefan video sequence.

coarse-to-fine formulation. The pixel subset selection process is described in section 3.1 No other modifications are made to the procedure in section 2. It is used in the initialization and multiscale embedding of the SIGM, which is presented in section 3.2 and illustrated in Fig. 3. The iterative formulation is described in section 3.3 and Fig. 4.

3.1 Pixel subset selection

The pixel subset $\widehat{S}$ is chosen by finding the pixels having the largest gradient magnitude. In order to avoid numerical instabilities caused by the concentration of the subset $\widehat{S}$ in small image regions. The image is divided into 100 sub-regions, where at each sub-region the top 10% are chosen. A similar approach was used for feature tracking by Dellaert et-al [12], while Wei et-al [23] improved the GM robustness by using selective integration where the selection criterion was based on temporal gradient sorting. This procedure was also used by [9]. Figure 2 illustrates the proposed pixel selection process, which was applied to the first frame of the Stefan sequence. The uniform distribution of the feature points is evident.

3.2 SIGM multiscale scheme

This section presents the multiscale SIGM registration scheme which uses the iterative refinment algorithm in section 3.3.

1. Similarly to section 2, a resolution pyramid of the input images, I_1 and I_2, is constructed.

2. At the lowest resolution level $Scale = 0$, N pixels in I_2, having the largest gradient magnitude are added to the pixel set $\widehat{S}_2\,(Scale)$, following the procedure described in section 3.1 and Fig. 2.

3. The pixel set $\widehat{S}_2\,(Scale)$ is used as an input to the iterative refinment algorithm in section 3.3, where the initial estimate of the motion can be either set to zero or calculated according to the motion of the previous frames.

4. The pixel set $\widehat{S}_2\,(Scale)$ and the result of step 2 are upscaled and used as an input to the calculation of steps 2 at a higher resolution scale, until the original image size is reached.

3.3 SIGM iterative refinment

At each resolution scale the initial estimate of the registration is refined using the following procedure:

1. At the first iteration in each resolution scale ($n = 0$), the matrix $\left(\underline{\underline{H}}^t\,\underline{\underline{H}}\right)$ is calculated according to Eqs. (2.5) and (2.6) using the pixel set $\widehat{S}_2$.

2. The pixel set $\widehat{S}_1$, which is a warping of I_1 towards the pixel set $\widehat{S}_2$, is calculated using the inverse of the current estimate $\underline{P}_n^{-1}$, $n \geq 0$. For $n = 0$, $\underline{P}_0$ is given as input.

3. If a pixel in $\widehat{S}_2$, does not have a corresponding pixel in I_1, it is extracted from $\widehat{S}_2$ and its contribution to the matrix $\left(\underline{\underline{H}}^t\,\underline{\underline{H}}\right)$ is subtracted.

4. The vector $\left(\underline{\underline{H}}^t\,\underline{I}_t\right)$ is calculated according to $\widehat{S}_1$ and $\widehat{S}_2$.

5. Equation (2.3) is solved for $\underline{\Delta P}$ using $\left(\underline{\underline{H}}^t\,\underline{\underline{H}}\right)$ and $\left(\underline{\underline{H}}^t\,\underline{I}_t\right)$ calculated above.

6. $\underline{\Delta P}$, the outcome of step 2 is used to update the solution

$$\underline{P}_{n+1} = \underline{\Delta P} + \underline{P}_n \qquad n \geq 0.$$

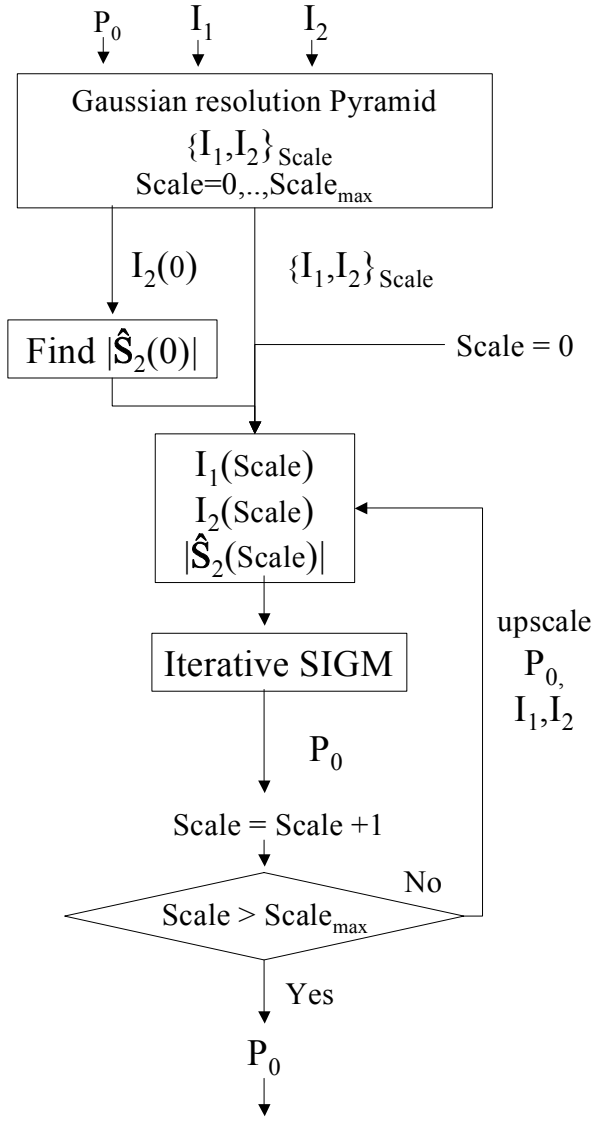

Figure 3: SIGM multiscale image registration flow chart. The scheme utilizes the iterative registration presented in Fig. 4.

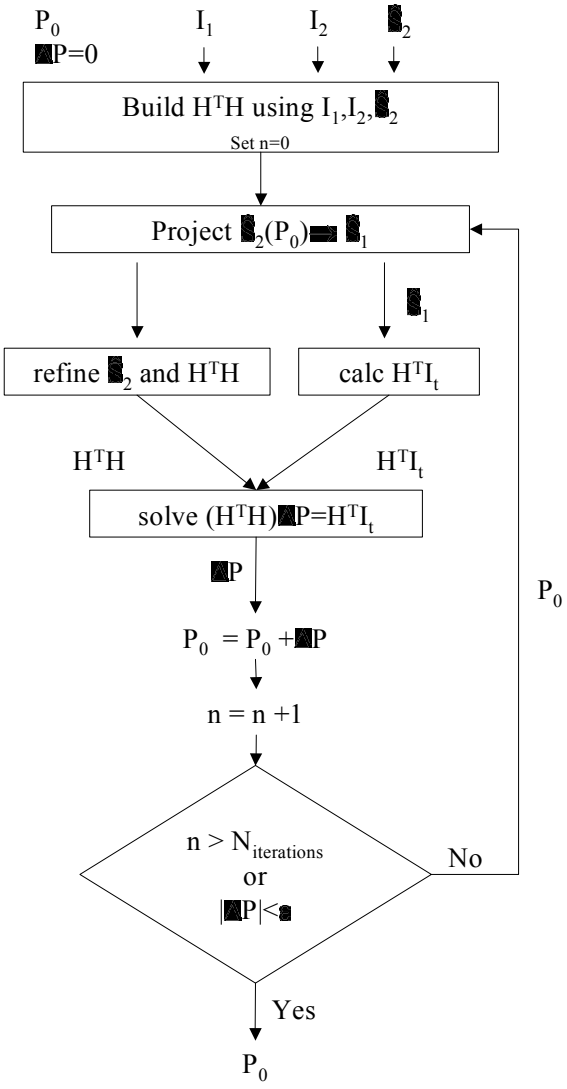

Figure 4: SIGM iterative image registration flow chart.

7. Step 2 is repeated until one of the following stopping criteria is met:

   (a) At most $N_{\max}$ iterations were performed

   or

   (b) The process is stopped if the translation parameters within the updated term $\underline{\Delta P}$ reach a predetermined threshold which corresponds to the required registration accuracy. A threshold of 0.1 pixel was used as a practical limit to the motion estimation accuracy [13].

3.4 Complexity analysis

Next we provide an estimation of the GM algorithm's complexity and a comparison to the SIGM. Let $C_{GM}(Scale)$ be the total complexity of a regular GM at a certain resolution scale $Scale$, then

$$C_{GM}(Scale) = K_{GM} |S_2| N_{Iterations} \tag{3.1}$$

where

K_{GM}	the number of operations per pixel.
$\|S_2\|$	the number of pixels in the set S_2.
$N_{Iterations}$	the number of iterations per resolution scale.

Hence, the total complexity of the iterative multi scale process becomes

$$\begin{aligned} C_{GM} &= \sum_{m=0}^{N_{scales}-1} C_{GM}(m) \\ &= \sum_{m=0}^{N_{scales}-1} \left(ScaleStep^2\right)^m \cdot C_{GM}(0) \\ &= N_{Iterations} \cdot K_{GM} \cdot |S_2(0)| \cdot \frac{(ScaleStep)^{2N_{scales}} - 1}{(ScaleStep)^2 - 1} \end{aligned} \tag{3.2}$$

where

$C_{GM}(0)$	the GM complexity at the coarsest resolution scale.
$ScaleStep$	the resolution scale step (the image's downscaling factor in each dimension).
$ScaleStep^2$	the ratio of the number of pixels between successive resolution scales.
$\|S_2(0)\|$	the number of pixel used for the GM estimation at the lowest resolution scale.

Therefore, the SIGM significantly reduces the GM complexity:

1. The matrix $\left(\underline{\underline{H}}^t \underline{\underline{H}}\right)_{SIGM}$ is calculated using a small subset of pixels $\widehat{S}_2$, which is much smaller than the pixel set S_2 used by the GM in section 2. The complexity reduction is

$$\frac{C_{\left(\underline{\underline{H}}^t \underline{\underline{H}}\right)_{SIGM}}}{C_{\left(\underline{\underline{H}}^t \underline{\underline{H}}\right)_{GM}}} = \frac{\left|\widehat{S}_2\right|}{|S_2|}. \tag{3.3}$$

2. I_2 remains static throughout the iterative solution process (I_1 is being warped). Hence, there is no need to recalculate the matrix $\left(\underline{\underline{H}}^t\underline{\underline{H}}\right)_{SIGM}$ in each iteration. It has to be calculated just once at the first iteration of each resolution scale.

In the SIGM case the matrix $\left(\underline{\underline{H}}^t\underline{\underline{H}}\right)$ has to be estimated just once, while $\left(\underline{\underline{H}}^t\underline{I_t}\right)$ has to be evaluated at each iteration

$$C_{SIGM}\left(Scale\right) = K_{GM}^{\left(\underline{\underline{H}}^t\underline{\underline{H}}\right)}\left|\widehat{S}_2\right|\left(1+\Delta\widehat{S}_2\right) + K_{GM}^{\left(\underline{\underline{H}}^t\underline{I_t}\right)}\left|\widehat{S}_2\right| N_{Iterations}, \forall rs \tag{3.4}$$

where

$K_{GM}^{\left(\underline{\underline{H}}^t\underline{\underline{H}}\right)}$	the complexity of estimating $\left(\underline{\underline{H}}^t\underline{\underline{H}}\right)$ using a single pixel.
$K_{GM}^{\left(\underline{\underline{H}}^t\underline{I_t}\right)}$	the complexity of estimating $\left(\underline{\underline{H}}^t\underline{I_t}\right)$ using a single pixel.
$\Delta\widehat{S}_2$	the change in the size of the set $\widehat{S}_2$.

The total SIGM complexity becomes

$$C_{SIGM} = \sum_{m=0}^{N_{scales}-1} C_{SIGM}\left(m\right). \tag{3.5}$$

Following Eq. (3.4), C_{SIGM} is not a function of the resolution scale $Scale$, therefore we have

$$\begin{aligned} C_{SIGM} &= \sum_{m=0}^{N_{scales}-1} C_{SIGM}\left(0\right) \\ &= C_{GM}\left(0\right)\cdot N_{scales}. \end{aligned} \tag{3.6}$$

For a typical global motion estimation in a video sequence (320×240), using three dyadic resolution scales ($N_{scales}=2, ScaleStep=2$) and 10 iterations ($N_{Iterations}=10$) we get

- $\left|S_2\left(0\right)\right| = 320\times240/4 = 19,200, \left|\widehat{S}_2\left(0\right)\right| = \left|S_2\left(0\right)\right|\cdot 10\% = 1,920$
- $\Delta\widehat{S}_2$ =10%
- $K_{GM}^{\left(\underline{\underline{H}}^t\underline{I_t}\right)}$ = ($No.motion$ parameters)+ (4 multiplications were needed to interpolate $\underline{I_t}$)
- $K_{GM}^{\left(\underline{\underline{H}}^t\underline{\underline{H}}\right)}$ = no. multiplications which were needed to evaluate $K_{GM}^{\left(\underline{\underline{H}}^t\underline{\underline{H}}\right)}$, taking into account that $\left(\underline{\underline{H}}^t\underline{\underline{H}}\right)$is symmetric.

	Translation	Affine	Projective
$\left(K_{GM}^{\left(\underline{\underline{H}}^t\underline{\underline{H}}\right)}, K_{GM}^{\left(\underline{\underline{H}}^t\underline{I_t}\right)}, K_{GM}\right)$	$(3, 7, 10)$	$(21, 10, 31)$	$(46, 12, 58)$
GM	$9.6 \cdot 10^6$	$30 \cdot 10^6$	$56 \cdot 10^6$
$SIGM$	$14 \cdot 10^4$	$23 \cdot 10^4$	$33 \cdot 10^4$
Complexity gain: $\frac{C_{GM}}{C_{SIGM}}$	≈ 70	≈ 130	≈ 170

Table 1: Performance comparison between the SIGM and GM showing a significant computational complexity reduction achieved by the SIGM.

The complexity analysis shown in Table 1 demonstrates a significant complexity reduction achieved by the SIGM especially for the advanced motion models, such as the affine and the projective. It should be noted that for the affine motion model we have

$$K_{GM}^{\left(\underline{\underline{H}}^t\underline{I_t}\right)} \cdot N_{Iterations} \gg K_{GM}^{\left(\underline{\underline{H}}^t\underline{\underline{H}}\right)} \tag{3.7}$$

hence, the affine motion estimation complexity is dominated by the complexity of evaluating $\underline{\underline{H}}^t\underline{I_t}$, which can be reduced using the Warp-free GM algorithm described in the section 4.

4 Warp-free motion estimation

In order to reduce the complexity of the GM algorithms, we reformulate Eq. (2.2) such that no warping is needed for the evaluation of $I_1(x_i^{(1)}, y_i^{(1)})$ while maintaining the same accuracy [13]. We start by rewriting the regular GM formulation for the 1D case in section 4.1. Then, the 1D warp-free reformulation for translational motion is presented in section 4.1, while its extension to 2D and general motion models is shown in section 4.3.

4.1 1D Gradient Methods formulation

We consider the registration of two one-dimensional discrete signals $I_1(x)$ and $I_2(x)$ sharing some common interval. Using a 1D formulation of section 2 we get

$$I_1\left(x_i^{(1)}\right) = I_2\left(x_i^{(2)}\right) \tag{4.1}$$

where $x_i^{(1)}$ and $x_i^{(2)}$ are the coordinates of the ith sample common to I_1 and I_2.

Assuming relative translational 1D motion we have

$$x_i^{(1)} = x_i^{(2)} + \Delta x. \tag{4.2}$$

Similar to section 2, Eq. (4.1) is solved by expanding I_2 in a first order Taylor expansion and solving for Δx

$$I_1(x_i^{(1)}) = I_2(x_i^{(2)}) + \frac{\partial I_2(x_i^{(2)})}{\partial x}\Delta x \tag{4.3}$$

where $I_1(x_i^{(1)})$ is evaluated at non-integral grid points using interpolation.

By gathering the pixel-wise equations

$$\underbrace{I_1\left(x_i^{(1)}\right) - I_2\left(x_i^{(2)}\right)}_{=\underline{\widehat{I}_t}} = \frac{\partial I_2(x_i^{(2)})}{\partial x}\Delta x. \tag{4.4}$$

an equation set is formulated

$$\underline{\underline{H}}\,\underline{P} = \underline{I_t} \tag{4.5}$$

and solved in the least-square sense similar to Eq. (2.3).

4.2 1D Warp-Free Gradient Methods formulation

Next we reformulate Eq. (4.3) to account for non-integral coordinate values. Let $x_i^{(1)}$ be the initial estimate of the ith common pixel, at a certain GM iteration. Then

$$x_i^{(1)} \triangleq \left\lfloor x_i^{(1)} \right\rfloor + \varepsilon_x, \qquad x_i^{(1)}, \varepsilon_x > 0. \tag{4.6}$$

Thus, instead of evaluating I_1 at $x = x_i^{(1)}$ we evaluate it at $x = \widehat{x}$

$$\widehat{x}_i^{(1)} \triangleq \begin{cases} \left\lfloor x_i^{(1)} \right\rfloor & |\varepsilon_x| \leqslant 0.5 \\ \\ \left\lfloor x_i^{(1)} \right\rfloor + 1 & |\varepsilon_x| > 0.5 \end{cases} \tag{4.7}$$

$$I_1(\widehat{x}_i^{(1)}) = I_2(x_i^{(2)}) + \frac{\partial I_2(x_i^{(2)})}{\partial x}\widehat{\Delta x} \tag{4.8}$$

where

$$\widehat{\Delta x} \triangleq \Delta x + \widehat{\varepsilon_x} \tag{4.9}$$

and $\widehat{\varepsilon_x}$ is the coordinate shift

$$\widehat{\varepsilon_x} \triangleq \widehat{x}_i^{(1)} - x_i^{(1)}. \tag{4.10}$$

Substituting Eq. (4.9) into Eq. (4.8) we get

$$\begin{aligned} I_1(\widehat{x}_i^{(1)}) &= I_2(x_i^{(2)}) + \frac{\partial I_2(x_i^{(2)})}{\partial x} \cdot \widehat{\varepsilon_x} \\ &= I_2(x_i^{(2)}) + \frac{\partial I_2(x_i^{(2)})}{\partial x} \cdot \Delta x + \frac{\partial I_2(x_i^{(2)})}{\partial x} \cdot \widehat{\varepsilon_x} \end{aligned} \tag{4.11}$$

$$\underbrace{I_1(\widehat{x}_i^{(1)}) - I_2(x_i^{(2)}) - \frac{\partial I_2(x_i^{(2)})}{\partial x} \cdot \widehat{\varepsilon_x}}_{=\widehat{I_t}} = \frac{\partial I_2(x_i^{(2)})}{\partial x} \cdot \Delta x \tag{4.12}$$

By comparing between Eqs. (4.3) and (4.12) we note that the left-hand-side of Eq. (4.3) uses 2 floating-point multiplications to interpolate the value of $I_1(x_i^{(1)})$ at non-integral coordinates, while the warp-free formulation in Eq. (4.12) uses a single floating-point multiplication.

4.3 Generalization to 2D signals and general motion models

Equation (2.2) is reformulated to estimate the input image at integral coordinates

$$I_1(\widehat{x},\widehat{y}) = I_2(x_i^{(2)},y_i^{(2)}) + \sum_{\substack{P_i \in \underline{P} \\ P_i \notin \Delta x, \Delta y}} \frac{\partial I_2(x_i^{(2)},y_i^{(2)})}{\partial P_i} P_i + \frac{\partial I_2(x_i^{(2)},y_i^{(2)})}{\partial (\Delta x)} \widehat{\Delta x} + \frac{\partial I_2(x_i^{(2)},y_i^{(2)})}{\partial (\Delta y)} \widehat{\Delta y} \tag{4.13}$$

where the shifted coordinates $(\widehat{x},\widehat{y})$, are defined similarly to Eq. (4.9)

$$\begin{aligned}\widehat{x} &\triangleq \begin{cases} \left\lfloor x_i^{(1)} \right\rfloor & |\varepsilon_x| \leqslant 0.5 \\ \left\lfloor x_i^{(1)} \right\rfloor + 1 & |\varepsilon_x| > 0.5 \end{cases} \\ \widehat{y} &\triangleq \begin{cases} \left\lfloor y_i^{(1)} \right\rfloor & |\varepsilon_y| \leqslant 0.5 \\ \left\lfloor y_i^{(1)} \right\rfloor + 1 & |\varepsilon_y| > 0.5 \end{cases}\end{aligned} \tag{4.14}$$

and $\left(\widehat{\Delta x},\widehat{\Delta y}\right)$ is defined similarly to Eq. (4.9)

$$\begin{aligned}\widehat{\Delta x} &\triangleq \Delta x + \widehat{x} - \left\lfloor x_i^{(1)} \right\rfloor \\ \widehat{\Delta y} &\triangleq \Delta y + \widehat{y} - \left\lfloor y_i^{(1)} \right\rfloor.\end{aligned} \tag{4.15}$$

Substituting Eqs. (4.14) and (4.15) into Eq. (4.13) we get

$$\begin{aligned} I_1(\widehat{x},\widehat{y}) - I_2(x_i^{(2)},y_i^{(2)}) = & \sum_{\substack{P_i\in\underline{P} \\ P_i\notin\Delta x,\Delta y}} \frac{\partial I_2(x_i^{(2)},y_i^{(2)})}{\partial P_i} P_i \\ & + \frac{\partial I_2(x_i^{(2)},y_i^{(2)})}{\partial(\Delta x)} \left(\Delta x + \widehat{x} - \left\lfloor x_i^{(1)} \right\rfloor\right) \\ & + \frac{\partial I_2(x_i^{(2)},y_i^{(2)})}{\partial(\Delta y)} \left(\Delta y + \widehat{y} - \left\lfloor y_i^{(1)} \right\rfloor\right) = \\ \sum_{P_i\in\underline{P}} \frac{\partial I_2(x_i^{(2)},y_i^{(2)})}{\partial P_i} P_i & + \frac{\partial I_2(x_i^{(2)},y_i^{(2)})}{\partial(\Delta x)} \left(\widehat{x} - \left\lfloor x_i^{(1)} \right\rfloor\right) + \frac{\partial I_2(x_i^{(2)},y_i^{(2)})}{\partial(\Delta y)} \left(\widehat{y} - \left\lfloor y_i^{(1)} \right\rfloor\right) \end{aligned} \tag{4.16}$$

$$\underbrace{I_1(\widehat{x},\widehat{y}) - I_2(x_i^{(2)},y_i^{(2)}) - \left(\frac{\partial I_2(x_i^{(2)},y_i^{(2)})}{\partial(\Delta x)} \left(\widehat{x} - \left\lfloor x_i^{(1)} \right\rfloor\right) + \frac{\partial I_2(x_i^{(2)},y_i^{(2)})}{\partial(\Delta y)} \left(\widehat{y} - \left\lfloor y_i^{(1)} \right\rfloor\right)\right)}_{=\widehat{I}_t} =$$

$$\sum_{P_i\in\underline{P}} \frac{\partial I_2(x_i^{(2)},y_i^{(2)})}{\partial P_i} P_i. \tag{4.17}$$

4.4 Algorithm Flow

1. The matrix $\left(\underline{\underline{H}}^t \underline{\underline{H}}\right)_{WFGM}$ is identical to $\left(\underline{\underline{H}}^t \underline{\underline{H}}\right)_{GM}$ calculated by the regular GM algorithm using I_2 according to Eq. (2.6):

$$\left(\underline{\underline{H}}^t \underline{\underline{H}}\right)_{k,j}^{I_2} = \sum_i \frac{\partial I_2\left(x_i^{(2)}, y_i^{(2)}\right)}{\partial P_k} \frac{\partial I_2\left(x_i^{(2)}, y_i^{(2)}\right)}{\partial P_j}. \tag{4.18}$$

2. $\underline{I_{t_i}}$ is calculated according to Eq. (2.5) and it is shifted according to Eq. (4.17):

$$\widehat{\underline{I}}_{t_i} = \underline{I_{t_i}} - \left(\frac{\partial I_2(x_i^{(2)}, y_i^{(2)})}{\partial(\Delta x)} \left(\widehat{x} - \left\lfloor x_i^{(1)} \right\rfloor\right) + \frac{\partial I_2(x_i^{(2)}, y_i^{(2)})}{\partial(\Delta y)} \left(\widehat{y} - \left\lfloor y_i^{(1)} \right\rfloor\right) \right). \tag{4.19}$$

3. We solve the Eq. (4.19) for $\underline{P}$

$$\left(\underline{\underline{H}}^t \underline{\underline{H}}\right) \underline{P} = \underline{\underline{H}}^t \widehat{\underline{I}}_t. \tag{4.20}$$

4. The warp-free GM returns $\underline{P}$ as its result.

4.5 Complexity analysis

Following Eq. (4.17) and Table 2, we established that the warp-free GM reduced the complexity related to the evaluations the vector $\left(\underline{\underline{H}}^t \underline{I_t}\right)$ in Eq. (2.3). 2 floating point multiplications per entry are needed rather than 4. This improvement is insignificant when the regular GM is used, since its complexity is dominated by $K_{GM}^{\left(\underline{\underline{H}}^t \underline{\underline{H}}\right)}$. However, when the WFGM is implemented together with the SIGM a significant additional improvement is achieved.

5 Fast GM algorithm

In order to reduce the complexity of the GM algorithm, the formulations presented in Sections 3 and 4 were integrated into the Fast GM algorithm:

1. A multiscale pyramid is built following section 2.

	$C_{\underline{\underline{H}^t}\underline{I_t}}$	$C_{\underline{\underline{H}^t}\underline{I_t}}$ $N_{Param}=6, N_{Iterations}=10$ $\lvert S_2\rvert = 176\times 144, \frac{\lvert\widehat{S_2}\rvert}{\lvert S_2\rvert}=10\%$
GM	$\lvert S_2\rvert\, N_{Iterations}\,(N_{Param}+4)$	$2.5{\cdot}10^6$
WFGM	$\lvert S_2\rvert\, N_{Iterations}\,(N_{Param}+2)$	$2.0{\cdot}10^6$
$\frac{C_{WFGM}}{C_{GM}}$	$\frac{N_{Param}+2}{N_{Param}+4}$	0.8

Table 2: Performance comparison of affine motion. The regular GM was taken as the reference CPU complexity. The complexity related to the calculation of $\underline{\underline{H}^t}\underline{I_t}$ is reduced by 20%.

2. Utilizing the SIGM algorithm (section 3), a set of points $\widehat{S}_2$, is selected in the image I_2 at the coarsest resolution scale.

3. Starting with the coarsest resolution scale Eq. (2.3) is solved iteratively according to the WFGM in section 4.4.

4. The iterative and multiscale refinment were conducted according to the SIGM, where the motion estimation results and pixel set $\widehat{S}_2$, were propagated through the resolution pyramid in a coarse-to-fine manner.

Thus, the evaluation of $\underline{\underline{H}^t}\underline{\underline{H}}$ is optimized by the SIGM while the evaluation of $\underline{\underline{H}^t}\underline{I_t}$ is optimized by the WFGM. The complexity estimation presented in Table 3, shows the overall improvement, provided by the Fast GM, to be of $O(100)$. The complexity was estimated for the test case introduced in Table 1. Most of the improvement is achieved by the SIGM, while the WFGM accomplishes an additional improvement of 10-15%.

	Affine	Projective
GM	$30\cdot 10^6$	$56\cdot 10^6$
$SIGM$	$23\cdot 10^4$	$33\cdot 10^4$
$Fast\ GM$	$20\cdot 10^4$	$29\cdot 10^4$
$\frac{C_{Fast\,GM}}{C_{GM}}$	0.7%	0.5%
$\frac{C_{Fast\,GM}}{C_{SIGM}}$	87%	88%

Table 3: Performance comparison between the SIGM and GM: approximated number of multiplications needed for the scenario introduced in Table 1.

6 Experimental Results

Affine motion is utilized by the MPEG-4 video compression standard as its main GME motion model. In order to explore the Fast GM coding and its associated complexity performances, the proposed scheme was integrated into the MPEG-4 Verification Model software [11] and compared to the GME algorithm implemented in it, while no other modifications were made. Simulations have been carried out using the sequences shown in Fig. 5: Mobile, Stefan and Coastguard at CIF size (352×240). Both the Fast GM and regular GM used the 3-step initialization method described in [9]. Two resolution scales were constructed using a three-tap filter $\begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}$ and the spatial derivatives were approximated using the mask $\begin{bmatrix} \frac{1}{2} & 0 & -\frac{1}{2} \end{bmatrix}$. The iterative termination threshold at each resolution scale was a translation update of 10^{-1} or at most 10 iterations. The pixel subset $\left|\widehat{S}_2\right|$ used by the fast GM, was 10% of the pixels in the lowest resolution scale, which amounts to 2.5% of the pixels at the original resolution scale. The test sequences were encoded in interframe mode (IPPPPP...) and two fixed quantizer sizes $Q = 10$ and $Q = 31$. $Q = 31$ corresponds to measuring the Global motion compensation (GMC) error directly, while $Q = 10$ relates to a typical compression scenario. The coding efficiency results of the MPEG-4 using the Fast GM were compared to those of the regular GM and the non-GMC compression mode. The results shown in Fig. 6 and Table 4 establish that the Fast GM achieves the same coding efficiency as the regular GM while exhibiting 20 times less computational complexity. In both quantizer settings, the difference in the compressed frame size was no more than 2-3%. Figure 7.a frames 175-190 and 230–270, present a situation where the use of GME substantially improves the compression ratio as the encoded frame sizes of the GME modes (red and blue graphs) are considerably smaller then those of the non-GME mode (green graph). In these sub-sequences, significant global motion occurs within the Stefan sequence as the player rushes to the net. Thus, the GME is able to smooth the bitrate needed to encode the whole sequence at a fixed quantizer (quality). In the Coastguard sequence (Fig. 6.b) the improvement caused by the GME is less significant, as the global motion in this sequence is a slow uniform translational motion. This type of motions are efficiently compressed by the non-GME mode, since the MPEG-4 standard utilizes differential coding of motion vectors [11, 20]. In particular, Fig. 7.b presents a sub-sequence of the Coastguard, where there is no significant global motion, yet the Fast GM achieves similar results to the

regular GM using 20 fold less computational complexity. Similar results can be observed in Fig. 6.c (Mobile sequence) where a dominant, very slow translational background motion exists. For the Stefan and Coastguard sequence the average compression ratio was improved, while the compression of Mobile sequence resulted in a decrease of the compression ratio. This occurs since the GME mode is switched on\off on a macro-block basis within the MPEG-4 verification model, based on their reconstruction error. In this procedure the actual coding overhead of the GME parameters is not taken into account. Therefore, this problem can be easily overcome by using either a two-pass encoding process or a GME switching algorithm taking into account the coding overhead. Table 5 presents the experimental timing results of the proposed algorithm recorded. The execution time of the GME module were measured directly using the built-in Microsoft Visual C++ profiler [25]. Overall compression performance was measured by a stop-watch over a 300 frames period for each sequence. Our implementation uses standard C++ without any assembly-level optimizations and in order to avoid the performance bias caused by disk and memory caching [24], each test was run 10 times and the results were averaged. The results show that the proposed GME module achieves the expected computational complexity gains presented in Table 4. The overall performance was only improved by a factor of approximately 4, since other components within the compression algorithm became complexity-wise dominant. Several sizes of $\left|\widehat{S}_2\right|$ were tested, for $\frac{\left|\widehat{S}_2\right|}{|S_2|} = 20\%$ the compression results were identical to the results presented above, while for smaller pixel set sizes $\left(\frac{\left|\widehat{S}_2\right|}{|S_2|} = 5\%, 2\%\right)$ a decrease of the compression ratio was noticed. A Pentium PC P800MHz was used for the timing experiments.

We conclude that in sequences where a significant global motion is present, the Fast GM exhibits compression results similar to the regular GM, while using significantly less (upto 33 fold) less computational complexity. In Sub-sequences where using GME does not result in compression ratio improvement, the Fast GM may produce lower compression ratios compared to the regular GM upto 10%.

7 Conclusions and Future Work

In this paper we proposed a new motion estimation algorithm based on gradient methods, which significantly reduces the complexity of state-of-the-art Global motion estimation (GME) algorithms, while achieving similar accuracy. This property makes it especially suit-

Figure 5: Video sequences used for the registration accuracy tests. (a) *Mobile*, (b) *Coastguard*, (c) *Stefan*.

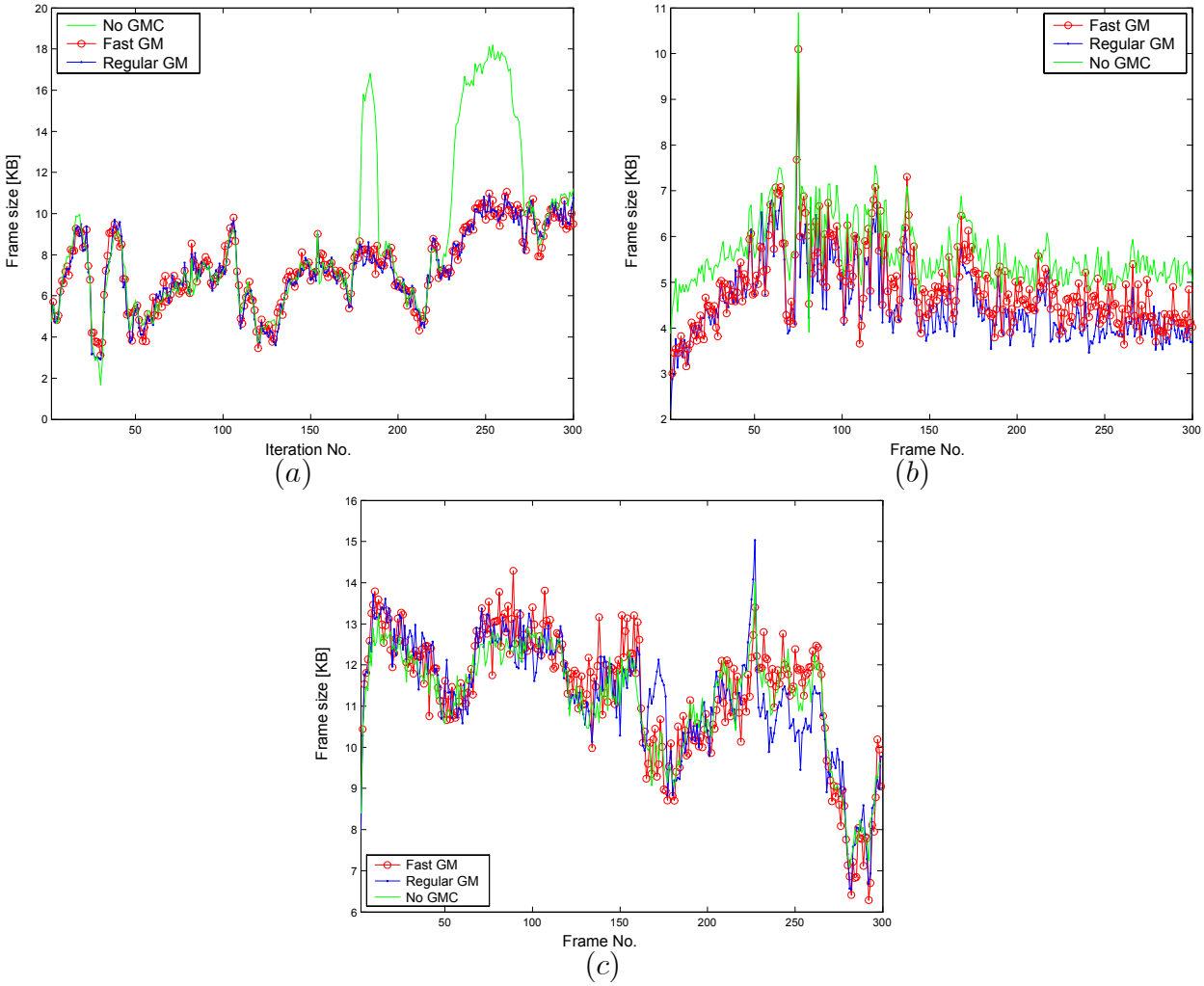

Figure 6: MPEG4 Compression results using the Fast GM, regular GM and non-GMC compression modes for (a) Stefan, (b) Coastguard and (b) Mobile sequences. The sequences were encoded using $Q = 31$, thus the compression results are directly related to the GMC efficiency. The Fast GM provides compression results very similar to the regular GM, while utilizing 20 times less computational complexity.

	Mean frame size[KByte]					
	Q=31			Q=10		
	Stefan	Coastguard	Mobile	Stefan	Coastguard	Mobile
GM	7.3732	4.6155	10.13	37.35	21.84	66.22
Fast GM	7.4322	4.9330	10.3	37.42	22.12	66.15
No GM	8.6437	5.7127	11.3	42.18	21.73	65.76
Complexity gain	20	19.81	33	20	19.81	33

Table 4: Average sizes of the compressed video frames. The Fast GM achieves the same coding efficiency as the regular GM. The bottom line shows the expected computational complexity gain achieved by using the Fast GM instead of the regular GM.

	Stefan 352×240		Stefan 176×144		Coastguard 352×288		Mobile 352×288	
	time $[ms]$	fps	time $[ms]$	fps	time $[ms]$	fps	time $[ms]$	fps
GM	617	1.5	194	4.41	612.14	1.36	1046	1.20
Fast GM	31.82	6.67	11.7	13	32	5.45	32	5.35
Gain	19.4	4.44	16.6	2.94	19.1	4.0	32.66	4.45

Table 5: Global motion estimation (GME) timing data: the "time" column indicates the average time spend in the global estimation function per frame. The second column presents the number of frame per second archived using each GME mode.

able for low-power GME applications such as video compression. The theoretical complexity analysis is back up by experimental results, which were obtained using the MPEG-4 Verification Model. The algorithm can also be used to accelerate image mosaics production and virtual reality applications [3]. In order to avoid compression ratio reductions due to sequences with no inherent global motion, we intend to develop a better algorithm for choosing when to use the GME mode.

References

[1] J. L. Barron, D. J. Fleet and S. S. Beauchemin, "Performance of Optical flow Techniques", Int. J. Computational Vision, Vol. 12, pp. 43-77, 1994.

[2] B. K. P. Horn and B. G. Schunck, "Determining optical Flow," Artificial Intelligence,vol. 17, pp. 185-203, 1981.

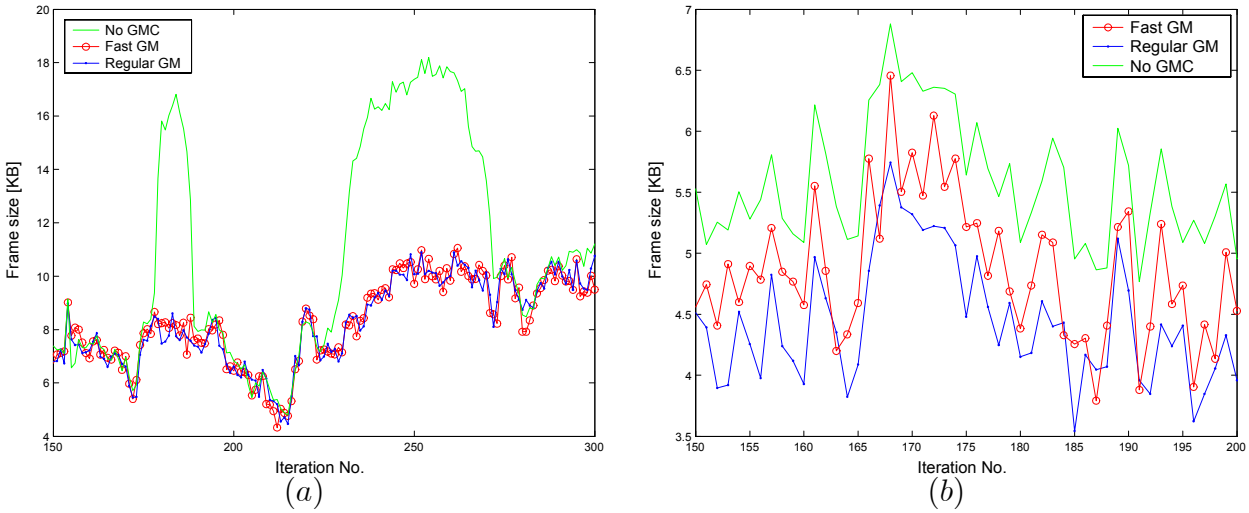

Figure 7: MPEG4 Compression results for sub-sequences within the sequences in Fig. 6. (a) In this subsequence of Stefan, a global motion occurs as the player rushes to the net. Using GME results in a significant reduction in the encoded frame size. (b) A subsequence of Fig. 6.b,due to significant occlusion in the scene the improvement caused of the Fast GM is 10% less than the regular GM, but the computational complexity is less by 20 fold.

[3] M. Irani and S. Peleg, "Motion Analysis for Image Enhancement: Resolution, Occlusion and Transparency". Journal of Visual Communication and Image Representation, Vol. 4, No. 4, pp. 324-335, December 1993.

[4] M. Irani, P. Anandan, and S. Hsu. "Mosaic based representations of video sequences and their applications", International Conference on Computer Vision, pp. 605-611, Boston, USA, June 1995.

[5] M. Irani, B. Rousso, and S. Peleg. "Computing occluding and transparent motions". International Journal of Computer Vision, 12(1): pp. 5–16, 1994.

[6] R. Szeliski "Video mosaics for virtual environments". IEEE Computer Graphics and Applications, 16(2): pp. 22–30, 1996.

[7] A. Tekalp, "Digital Video Processing", Prentice Hall, 1995.

[8] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in Proc. DARPA, Image Understanding Workshop, pp. 121-130, April 1981.

[9] F. Dufaux and J. Konrad, "Efficient, Robust, and Fast Global Motion Estimation for Video Coding", IEEE Transactions on Image Processing, vol. 9, no. 3, pp. 497-501, March 2000.

[10] M. Irani and P. Anandan. "All About Direct Methods", International Workshop on Vision Algorithms, Corfu, Greece, September 1999.

[11] Shigeru Fukunaga et al., "MPEG-4 Video Verification Model Version 16.0", ISO/IEC document JTC1/SC29/WG11 N3312, March 2000/Noordwijkerhout

[12] F. Dellaert and R. Collins, "Fast Image-Based Tracking by Selective Pixel Integration", ICCV 99 Workshop on Frame-RateVision, September, 1999.

[13] A. Averbuch, Y. Keller, "Warp free Gradient methods based image registration", Technical report TBD, Tel-Aviv University, November 2001.

[14] A. Smolić and T. Wiegand, "High-Resolution Video Mosaicing", in Proc. IEEE International Conference on Image Processing (ICIP), Thessaloniki, Greece, Sep. 2001.

[15] E. Steinbach, T. Wiegand, B. Girod, "Using Multiple Global Motion Models for Improved Block-Based Video Coding", IEEE International Conference on Image Processing, ICIP-99, vol. 2, pp. 56-60, Kobe, Japan, October 1999.

[16] H. Sawhney and S. Ayer, "Compact representation of videos through dominant and multiple motion estimation", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 18, No. 8, pp. 814-830, August 1996.

[17] A. Smolic, T. Sikora and J-R. Ohm, "Long-term Global Motion Estimation and Its Application for Sprite Coding, Content Description, and Segmentation". IEEE Trans. Circuits Syst. Video Technol., Vol. 9 (Dec. 1999), 1227-1242

[18] N. Grammalidis, D. Beletsiotis, M. Strintzis,: "Sprite Generation and Coding in Multiview Image Sequences". IEEE Trans. Circuits Syst. Video Technol., Vol. 10 (Mar. 2000), 302-311

[19] F. Dufaux, "Results for Video Coding Using Dynamic Sprite (Core Experiment N3)", Technical Report M1458, ISO/IEC JTC1/SC29/WG11 MPEG-4 meeting, Maceio, Brazil, November 1996.

[20] T. Sikora. "The MPEG-4 Video Standard Verification Model". IEEE Trans. on Circuits and Systems for Video Technology, 7:19–31, Feb. 1997

[21] F. Dufaux, "Background mosaicking", ISO/IECJTC1/SC29/WG11, MPEG96/M0653, Munich, Germany, Jan. 1996.

[22] P. Gill , "Practical Optimization", Academic Press, 1982.

[23] Q. Wei, H.J. Zhang and Y. Zhong, , "A Pre-Analysis Method for Robust Global Motion Estimation".IEEE International Conference on Image Processing, ICIP-99, vol. 2, pp. 26-30, Kobe, Japan, October 1999.

[24] R. Cutler and L. Davis. "Developing Real-Time Computer Vision Applications for Intel Pentium III based Windows NT Workstations," FRAME-RATE: Frame-rate Applications, Methods and Experiences with Regularly Available Technology and Equipment, in conjunction with IEEE International Conference on Computer Vision (ICCV), September 1999, Kerkyra, Greece.

[25] Microsoft Developer Network. 2002. http://msdn.microsoft.com.