VERY FAST GLOBAL MOTION ESTIMATION USING PARTIAL DATA

Hussein Alzoubi, W. David Pan

Dept. of Electrical and Computer Engineering
University of Alabama in Huntsville
Huntsville, AL 35899
Email: <alzoubh, dwpan>@ece.uah.edu

ABSTRACT

The minimization process of the Levenberg-Marquardt algorithm (LMA) used in estimating the global motion parameters tends to be very expensive computationally due to the involvement of all the pixels of an image frame. We propose to reduce the computational complexity of the LMA by using only a small portion of the image data in two stages. In the first stage, we seek to reduce the complexity of the initial guess of the transformation parameters, which is critical to the final convergence of the algorithm. The complexity of computing the initial guess can be lowered by using just a small subset of the pixels in the calculation of the translational components. The second stage of the LMA algorithm is to find the final motion parameters in an iterative fashion, based on the coarse estimate of the motion parameters obtained in the previous stage. The LMA in this stage again operates on a subset of the pixels to further reduce the overall computational complexity. Both analytical and simulation results showed that the proposed partial-data algorithm could achieve a speedup factor of over 25 for global motion estimation (GME) with an eight-parameter perspective motion model on several video sequences, without significant loss in the estimation accuracy compared with the conventional LMA on the full image data.

Index Terms— Global motion estimation (GME), Levenberg-Marquardt algorithm, subset selection, perspective model, computational complexity

1. INTRODUCTION

The following eight-parameter perspective motion model was employed by MPEG-4 [1] to describe global motion by using the mapping functions $x(i,j)$ and $y(i,j)$.

$$x(i,j) = \frac{m_1 i + m_2 j + m_3}{m_7 i + m_8 j + 1}, \quad (1)$$

$$y(i,j) = \frac{m_4 i + m_5 j + m_6}{m_7 i + m_8 j + 1}. \quad (2)$$

In estimating the global motion parameters $(m_1, m_2, \ldots, m_8)$, we seek to minimize the following sum of squared errors.

$$E = \sum_{all\ i} \sum_{all\ j} e^2(i,j), \quad (3)$$

where $e(i,j)$ denotes the error of predicting a pixel located at (i,j) of frame I_{k+1}, by using a pixel at location $[x(i,j), y(i,j)]$ of the previous frame I_k.

$$e(i,j) = I_{k+1}[i,j] - I_k[x(i,j), y(i,j)]. \quad (4)$$

The Levenberg-Marquardt algorithm (LMA) [2][3] can be used to find the vector $\boldsymbol{m} = [m_1, m_2, ..., m_8]$ that minimizes E in (3). In LMA, the motion-parameter vector $\boldsymbol{m}$ is updated iteratively.

$$\boldsymbol{m}^{(n+1)} = \boldsymbol{m}^{(n)} + \boldsymbol{s}^{(n)}, \quad (5)$$

where $\boldsymbol{s}^{(n)}$ is an update (at iteration n), which can be found by solving the following equation.

$$\left[\boldsymbol{J}^T(\boldsymbol{m}^{(n)})\boldsymbol{J}(\boldsymbol{m}^{(n)}) + \mu^{(n)}\boldsymbol{I}\right]\boldsymbol{s}^{(n)} = -\boldsymbol{J}^T(\boldsymbol{m}^{(n)})\boldsymbol{r}(\boldsymbol{m}^{(n)}), \quad (6)$$

where $\boldsymbol{I}$ is the identity matrix and μ is a nonnegative scalar parameter. $\boldsymbol{r}(\boldsymbol{m})$ is a column vector defined in (7).

$$\boldsymbol{r}(\boldsymbol{m}) = \begin{bmatrix} e(1,1) \\ e(1,2) \\ e(1,3) \\ \ldots \\ e(2,1) \\ e(2,2) \\ e(2,3) \\ \ldots \\ all\ i,\ j \end{bmatrix}. \quad (7)$$

And $\boldsymbol{J}(\boldsymbol{m})$ is the Jacobian matrix of $\boldsymbol{r}(\boldsymbol{m})$, as given in (8).

$$\boldsymbol{J}(\boldsymbol{m}) = \begin{bmatrix} \frac{\partial e(1,1)}{\partial m_1} & \frac{\partial e(1,1)}{\partial m_2} & \ldots \\ \frac{\partial e(1,2)}{\partial m_1} & \frac{\partial e(1,2)}{\partial m_2} & \ldots \\ \frac{\partial e(1,3)}{\partial m_1} & \frac{\partial e(1,3)}{\partial m_2} & \ldots \\ \ldots & \ldots & all\ i,\ j \end{bmatrix} \quad (8)$$

Each entry in the above matrix can be evaluated using the following relations.

$$\left[\frac{\partial e(i,j)}{\partial m_k}\right] = \left[\frac{\partial e(i,j)}{\partial x}\right] \cdot \frac{\partial x}{\partial m_k}, \quad (9)$$

$$\left[\frac{\partial e(i,j)}{\partial m_k}\right] = \left[\frac{\partial e(i,j)}{\partial y}\right] \cdot \frac{\partial y}{\partial m_k}. \quad (10)$$

For the eight-parameter perspective model, $\frac{\partial x}{\partial m_k}$ and $\frac{\partial y}{\partial m_k}$ can be evaluated from (1) and (2). Below are some representative equations.

$$\frac{\partial x}{\partial m_1} = \frac{i}{D}, \frac{\partial x}{\partial m_2} = \frac{j}{D}, \frac{\partial x}{\partial m_7} = -\frac{xi}{D}, \frac{\partial x}{\partial m_8} = -\frac{xj}{D}$$
$$\frac{\partial y}{\partial m_5} = \frac{j}{D}, \frac{\partial y}{\partial m_6} = \frac{1}{D}, \frac{\partial y}{\partial m_7} = -\frac{yi}{D}, \frac{\partial y}{\partial m_8} = -\frac{yj}{D}, \quad (11)$$

where $D = m_7 i + m_8 j + 1$. In addition, the partial derivatives in (9) and (10) can be approximated as

$$\left[\frac{\partial e(i,j)}{\partial_x}\right] \approx (1-y_f)\left(I_k[x_i+1, y_i] - I_k[x_i, y_i]\right) + y_f\left(I_k[x_i+1, y_i+1] - I_k[x_i, y_i+1]\right). \quad (12)$$

Similarly,

$$\left[\frac{\partial e(i,j)}{\partial_y}\right] \approx (1-x_f)\left(I_k[x_i, y_i+1] - I_k[x_i, y_i]\right) + x_f\left(I_k[x_i+1, y_i+1] - I_k[x_i+1, y_i]\right), \quad (13)$$

where x_i, y_i are the integer parts and x_f, y_f are the fractional parts of coordinates x and y, respectively.

To calculate $\boldsymbol{r}(\boldsymbol{m})$ and $\boldsymbol{J}(\boldsymbol{m})$ in (7) and (8), the LMA involve all the pixels in an image frame. For each pixel, (9)-(13) must be solved. Therefore, the LMA is very costly computationally. The computational complexity of LMA could be reduced by operating on a small subset of pixels. In this paper, we study how the subset of pixels could be picked wisely with low pixel-selection overhead, so that the resulting LMA based on partial data could still yield reasonably accurate motion parameters. The remainder of the paper is organized as follows. Section 2 provides a brief survey of pixel selecton methods in the literature. In Section 3, a fast LMA algorithm based on partial pixel data is proposed, followed by a detailed computational complexity analysis of the proposed method in Section 4. Simulations results are then presented in Section 5. The paper is concluded in Section 6.

2. CHOOSING A SUBSET OF PIXELS FOR GME

The LMA can be accelerated if only a small subset of pixels is computed. However, searching for a good subset is not free. In [5][6], Keller and Averbuch proposed to use the gradient magnitude as the selection criterion to speed up the GME. The gradient of a pixel (i, j) can be obtained from (4) as follows

$$\nabla e(i,j) = \left[\frac{\partial e(i,j)}{\partial m_1}, \frac{\partial e(i,j)}{\partial m_2}, \ldots, \frac{\partial e(i,j)}{\partial m_8}\right]^T. \quad (14)$$

Before pixel selection, the image is divided into several sub-regions. After the gradient magnitude is calculated for each pixel, the top 10% pixels in a sub-region with the largest magnitudes will be selected. While this selection method can lead to reduced overall complexity, the overhead associated with such extra operations as gradient calculation, sorting and comparisons of magnitudes will offset a lot of gains in complexity reduction allowed by working with fewer pixels. On the other hand, a random subset selection method was proposed by Dellaert and Collins [7] for image-based tracking. This is a rather low-overhead method that can be applied in GME, where a random bitmap can be generated to decide the positions of pixels to be selected. However, since the positions of the selected pixels are random, numerical instabilities might result.

In this paper, we propose to choose pixels based on some fixed patterns. Since neighboring pixels are very likely to experience the same kind of motion, we can choose only one representative pixel from a group of neighboring pixels to participate in calculating the motion parameters. The representatives are chosen according to a certain pattern (see Fig. 1). Obviously, the additional overhead of this selection method is very low. Although similar patterns were used in [8][9] for the purpose of reducing the complexity of block distortion measure (BDM) for block-based local motion estimation, we are not aware of any application of these subsampling patterns in GME.

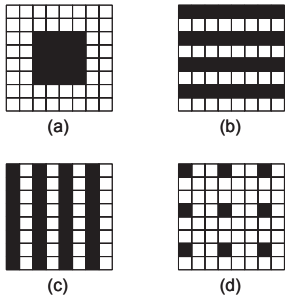

Fig. 1. Patterns of selected pixels (darkened). (a) Subsampling factor is 4. (b) and (c) Subsampling factor is 2. (d) Subsampling factor is $64/9 \approx 7$.

3. THE ALGORITHM

The LMA is a method based on gradient descent. To ensure convergence in the presence of large displacements, a coarse estimate of the translational component of the displacement is often performed in the initial stage, where a 3-step search is typically used [1]. Reducing the complexity of the initial step will certainly reduce the overall complexity of the GME algorithm. To speed up the rough estimation at the initial stage, we can include only part of the pixels in the 3-step search. The selection of pixels is again based on fixed patterns. Since the initial estimate is supposed to be coarse, using partial data in this stage will not affect too much the accuracy of the motion parameters generated by the LMA, as demonstrated by the simulation results in Section 5.

The overall LMA algorithm based on partial data can be summarized as follows.

1. In the initial stage, obtain a coarse estimate of the translational components of the displacement by using the 3-step search operating on a subset of pixels, where a subsampling factor of 36 is adopted by selecting one pixel from a square block of 36 pixels. After this initial stage is finished, the iterations of LMA start.
2. A threshold T is initialized to a large number (e.g., 255), to be used for outlier rejection, similar to that in [4], where 10% of the chosen pixels with the largest errors are excluded.
3. For the entire image, pick one pixel at location (i, j) from every square block of 25 pixels (following pattern (d) in Fig. 1). Compute its corresponding position (x, y), by using (1) and (2).
4. Compute the error e, using (4). If $e < T$, add the pixel's contribution to $\boldsymbol{r}(\boldsymbol{m})$ and $\boldsymbol{J}(\boldsymbol{m})$ in (7) and (8), respectively. However, if this is the first iteration, a histogram of $|e|$ is built.
5. Solve the equation (6) and update the motion parameters by using (5).
6. In the first iteration only, T is computed to exclude the top 10% pixels from the histogram.
7. Steps 3, 4, and 5 are repeated for a maximum of 32 steps. The process stops earlier if the update term $\boldsymbol{s}^{(n)}$ in (5) is smaller

than a threshold of 0.001 for the translational component and 0.00001 for the other motion parameters.

As suggested by the well-known *Amdahl's law* in computing, in order to reduce the overall complexity the GME, we need to reduce the complexity of both the initial stage and the subsequent iterative stages of the algorithm accordingly. A detailed complexity analysis of the algorithm is given below.

4. COMPLEXITY ANALYSIS

Let C_{Full} and $C_{Partial}$ denote the complexities of LMA based on the full set of pixels and based on a subset of pixels, respectively. Then

$$C_{Full} = C_{InitialGuess} + C_{Iterative}, \tag{15}$$

where $C_{InitialGuess}$ and $C_{Iterative}$ denote the complexities of the initial stage and the subsequent stages of the algorithm, respectively. Let N_{Iter} denote the number of iterations, N_{Full} be the total number of pixels in a frame, and C_{Pixel} be the number of arithmetic operations required for each pixel, then (15) becomes

$$C_{Full} = C_{InitialGuess} + N_{Iter}N_{Full}C_{Pixel} \tag{16}$$

Assume that $N_{Iter} = 10$ (our simulations showed that 10 iterations are quite sufficient for LMA to converge.). $N_{Full} = rows \times cols$, where $rows$ and $cols$ refer to the number of rows and columns of a frame, respectively. For the CIF format, $N_{Full} = 352 \times 288 = 101,376$; and for the QCIF format, $N_{Full} = 176 \times 144 = 25,344$. If we denote the desired subsampling factor by SF_{Des}, the actual subsampling factor SF_{Act} can be determined as

$$SF_{Act} = \frac{N_{Full}}{\left\lceil \frac{rows}{\sqrt{SF_{Des}}} \right\rceil \left\lceil \frac{cols}{\sqrt{SF_{Des}}} \right\rceil}. \tag{17}$$

For example, if we desire the subsampling factor to be 25, the actual factor will be 24.62 for CIF and 24.28 for QCIF, approximately. And if the desired subsampling factor is 36, the actual one will be about 35.8 for a CIF image and about 35.2 for a QCIF image.

From (1)-(4) and (9)-(13), C_{Pixel} corresponds to 27 additions and subtractions and 31 multiplications and divisions. The least squares problem encountered in (6) can be solved by the Householder QR factorization [3], which requires about $\left(mn^2 - n^3/3\right)$ additions and the same number of multiplications, where m is the data size (e.g. $m = 101,376$ for a full CIF image) and n is the number of model parameters ($n = 8$ for the perspective model). Our simulations in a typical setting showed that out of a total 7 seconds required by the LMA on a single frame, about 1.1 seconds were spent on performing the initial 3-step search, whereas about 5.9 seconds were on the subsequent iterative stages of LMA. Thus the following relation can be established empirically: $C_{InitialGuess} \approx 1.9N_{Full}C_{Pixel}$. If follows that

$$C_{Full} \approx 1.9N_{Full}C_{Pixel} + 10N_{Full}C_{Pixel} = 11.9N_{Full}C_{Pixel}. \tag{18}$$

For the fast LMA on partial data, we have

$$C_{Partial} = C^P_{InitialGuess} + C^P_{Iterative}, \tag{19}$$

$$\frac{C^P_{Iterative}}{C_{Iterative}} = \frac{N_{Iter}N_{Partial}C_{pixel}}{N_{Iter}N_{Full}C_{pixel}} = \frac{N_{Partial}}{N_{Full}} = \frac{1}{24.62}, \tag{20}$$

in the case of video sequences in CIF format. If the 3-step search in the initial stage is applied on the full set of pixels, then the overall complexity becomes

$$\begin{aligned} C_{Partial} &\approx 1.9N_{Full}C_{Pixel} + \tfrac{10}{24.62}N_{Full}C_{Pixel} \\ &\approx 2.3N_{Full}C_{Pixel}, \end{aligned} \tag{21}$$

and hence the speedup factor is

$$Speedup = \frac{C_{Full}}{C_{Partial}} \approx \frac{11.9N_{Full}C_{Pixel}}{2.3N_{Full}C_{Pixel}} \approx 5.17. \tag{22}$$

But if we want to speed up the 3-step search by using a desired subsampling factor of 36, then $C^P_{InitialGuess} = N_{Full}C_{Pixel}/35.8$ and

$$\begin{aligned} C_{Partial} &\approx \tfrac{1}{35.8}N_{Full}C_{Pixel} + \tfrac{10}{24.62}N_{Full}C_{Pixel} \\ &\approx 0.46N_{Full}C_{Pixel}, \end{aligned} \tag{23}$$

which means a speedup factor of 5.02 over (21) can be achieved. The overall speedup factor now becomes

$$Speedup = \frac{C_{Full}}{C_{Partial}} \approx \frac{11.9N_{Full}C_{pixel}}{0.46N_{Full}C_{pixel}} \approx 25.87. \tag{24}$$

5. RESULTS

To demonstrate the effectiveness of our approach, we choose four video sequences (Carphone, Mobile, Tempete, and Tennis), with 250 frames from each of these sequences being used in the experiments. The Peak Signal-to- Noise Ratio (PSNR), which is a measure of the goodness of the prediction, and the computation times are used in making the comparisons between the proposed algorithm (denoted as *certain pattern*, or CP) and three other methods, including the conventional LMA operating on a full set of pixels (denoted as *full size*, or FS), LMA operating on a subset of pixels chosen according to the gradient criteria (denoted as GR) [5], and LMA operating on a subset of pixels chosen randomly (denoted as RC) [7]. The simulations were conducted on a PC with 3.0 GHz Pentium IV processor, 512 MB RAM and the MS Windows XP OS. The source codes were written in MATLAB.

Simulation results showed that, in the initial stage, applying the 3-step search on a very small subset of pixels (with one pixel being selected from a block of 36 pixels) had very little effect on the accuracy of the motion parameters generated by the LMA. Furthermore, reducing the complexity of the subsequent stages of the LMA by using a subset pattern with subsampling factor of 25 resulted in very negligible losses in the accuracy of motion parameters achievable by using the conventional LMA based on the full set of pixels.

Limited by the space, only the frame-by-frame PSNR values for the Carphone sequence is shown in Figure 2, which indicates that the degradations in PSNR caused by using only a subset of pixels in these fast methods (GR, RC, and CP) are so small, that their curves are almost indistinguishable from that of the FS method. The average PSNR values for the four video sequences are listed in Table 1. Whether the three-step search in the initial stage operates on partial data or not has literally no impact on the final PSNR achieved. However, it has significant impact on the overall computational complexity, as can be seen by comparing the computation times needed and the corresponding speedup factors listed in Table 2 and Table 3. Note that the Carphone sequence is in the QCIF format, whereas Mobile, Tempete, and Tennis sequences are in the CIF format. Thus we expect that the computation time for the CIF sequences will be about four times as long. Only 4% of the pixels were used in the

Table 1. Average PSNR (in dB) of 250 frames from four video sequences. Note that the actual PSNR values are only shown for the FS (full-data) method. For other methods, the degradations of their estimation accuracy (in dB) with respect to the FS method are shown, for ease of comparison.

Sequence	FS	GR	RC	CP
Carphone	32.65	0.09	0.46	0.31
Mobile	26.09	0.01	0.05	0.06
Tempete	27.89	0.01	0.03	0.02
Tennis	27.42	0.31	0.37	0.14

three fast LMA methods (GR, RC, and CP). From figure 2 and Table 1, we see that the proposed CP approach based on fixed patterns comes very close to the full size approach in terms of PSNR (less than 0.1 dB in difference for sequences Mobile and Tempete). The simulation results on the running times agree well with our analysis, which shows that a speedup factor of about 25 on average could be achieved by using the proposed CP method. In addition, the CP method is generally superior to the RC (random choice) method in terms of PSNR, and the CP method can achieve the highest speedup factor among the three fast methods based on partial data.

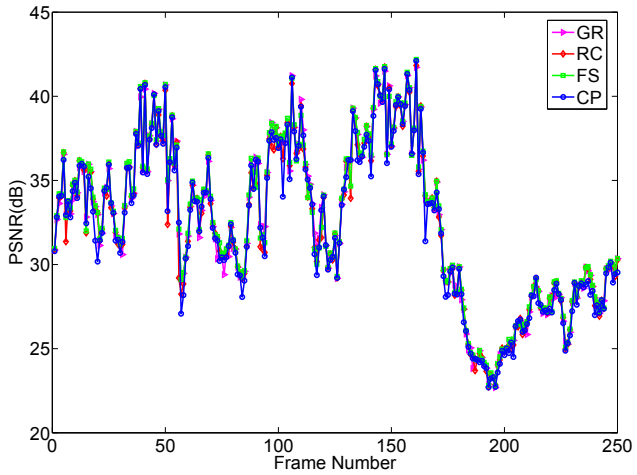

Fig. 2. Accuracy (in terms of PSNR) of four GME methods on Carphone sequence.

6. CONCLUSION

We presented a fast GME method using LMA that operates on a small subset of pixels. Partial data computation based on fixed subsampling patterns was found to be effective in reducing the complexity of the initial 3-step search, and in speeding up the subsequent iterative process in the LMA. We demonstrated that an overall speedup factor of more than 25 was achievable over the conventional LMA using a full set of pixels, without significant loss in the accuracy of global motion estimation.

Table 2. Computation times (in seconds) of the four methods (over 250 frames) and their average speedup factors over the FS method. The 3-step search in the first stage is applied on a full set of pixels.

Sequence	FS	GR	RC	CP
Carphone	428.5	199.7	84.3	77.9
Mobile	1749.9	944.3	366.2	335.2
Tempete	1763.4	946.1	369.4	337.1
Tennis	1720.6	945.9	363.2	332.8
Avg. Speedup	1.00	1.92	4.84	5.28

Table 3. Computation times (in seconds) of the four methods (over 250 frames) and their average speedup factors over the FS method. The 3-step search in the first stage is applied on a small subset of pixels (with a subsampling factor of 36).

Sequence	FS	GR	RC	CP
Carphone	382.6	142.6	21.6	19.5
Mobile	1504.2	715.9	87.4	64.2
Tempete	1527.1	716.5	87.0	68.8
Tennis	1483.6	715.6	86.4	63.7
Avg. Speedup	1.15	2.58	20.01	25.47

7. REFERENCES

[1] "MPEG-4 video verification model version 18.0," in: ISO/IEC JTC1/SC29/WG11, N3908, Pisa, Italy, 2001.

[2] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C: The Art of Scientific Computing,* Second Edition. Cambridge University Press, pp. 656-706, 2002.

[3] M. T. Heath, *Scientific Computing: An Introductory Survey,* Second Edition. McGraw-Hill, New York, 2002.

[4] F. Dufaux and J. Konrad, "Efficient, robust, and fast global motion estimation for video coding," *IEEE Trans. Image Processing,* vol. 9, no. 3, pp. 497-501, March 2000.

[5] Y. Keller and A. Averbuch, "Fast gradient methods based on global motion estimation for video compression," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 13, no. 4, pp. 300-309, April 2003.

[6] Y. Keller and A. Averbuch, "Fast gradient methods based on global motion estimation for video compression," *IEEE ICIP*, pp. 665-668, 2002.

[7] F. Dellaert and R. Collins, "Fast image-based tracking by selective pixel integration," *ICCV Workshop on Frame-Rate Vision,* Corfu, Greece, September 1999.

[8] B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 3, no. 2, pp. 148-157, April 1993.

[9] Y.-L. Chan and W.-C. Siu, "New adaptive pixel decimation for block motion vector estimation," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 6, no. 1, pp. 113-118, February 1996.