On Using Raw MPEG Motion Vectors To Determine Global Camera Motion

Maurizio Pilu
Digital Media Department
HP Laboratories Bristol
HPL-97-102
August, 1997

MPEG, motion estimation, mosaicing compressed domain

This document reports a simple and effective method to determine global camera motion using raw MPEG-1 motion vectors information obtained straight from the real MPEG-1 streams such as that of the new HITACHI MP-EG1A digital camcorder.

The simple approach we have experimented with robustly fits a global affine optic flow model to the motion vectors. Other more robust methods are also proposed. In order to cope with the Group-of-Frames (GOF) discontinuity of the MPEG stream, B frames are used backward to determine the "missing link" to a previous GOF thereby ensuring continuity of the motion estimation across a reasonable number of frames.

As a testbed, I have applied the method to the image mosaicing problem, for which the interesting results obtained are presented.

Although several other methods exists to perform camera motion estimation, the approach presented here is particularly interesting because exploits "free" information present in MPEG streams and bypass the highly expensive correlation process.

Internal Accession Date Only

On using raw MPEG motion vectors to determine global camera motion

Maurizio Pilu
Digital Media Department
HPLABS - Bristol

Abstract

This document reports a simple and effective method to determine global camera motion using raw MPEG-1 motion vectors information obtained straight from the real MPEG-1 streams such as that of the new HITACHI MP-EG1A digital camcorder.
The simple approach we have experimented with robustly fits a global affine optic flow model to the motion vectors. Other more robust methods are also proposed. In order to cope with the Group-of-Frames (GOF) discontinuity of the MPEG stream, B frames are used backward to determine the "missing link" to a previous GOF thereby ensuring continuity of the motion estimation across a reasonable number of frames.
As a testbed, I have applied the method to the image mosaicing problem, for which the interesting results obtained are presented.
Although several other methods exists to perform camera motion estimation, the approach presented here is particularly interesting because exploits "free" information present in MPEG streams and bypass the highly expensive correlation process.

1 Introduction

Recently, the MPEG standard[1] [1][2] has emerged as key technology to the diffusion of digital video and in the last few months we have seen many products using either MPEG or one of its variations (DVD, DVCAM, Motion JPEG) coming out in the consumer arena.

Key to the deployment of the technology in everyday applications is the parallel efforts in creating cheaper and better MPEG encoders. MPEG-1 and MPEG-2 single-chip encoders and decoders have been already available for some time and recently, besides rolling out a series of MPEG-1 encoders, some companies have announced single-chip MPEG-2 encoders.

MPEG-2 encoder technology, albeit sophisticated in implementation, is still far from ideal in performance, as the key motion estimation part is carried out using

[1] This technical report assumes that the reader has a reasonable understanding of the MPEG compression standard. Throughout the paper, there will be scattered description of part of it which are included for the sole purpose of detailing the presented method.

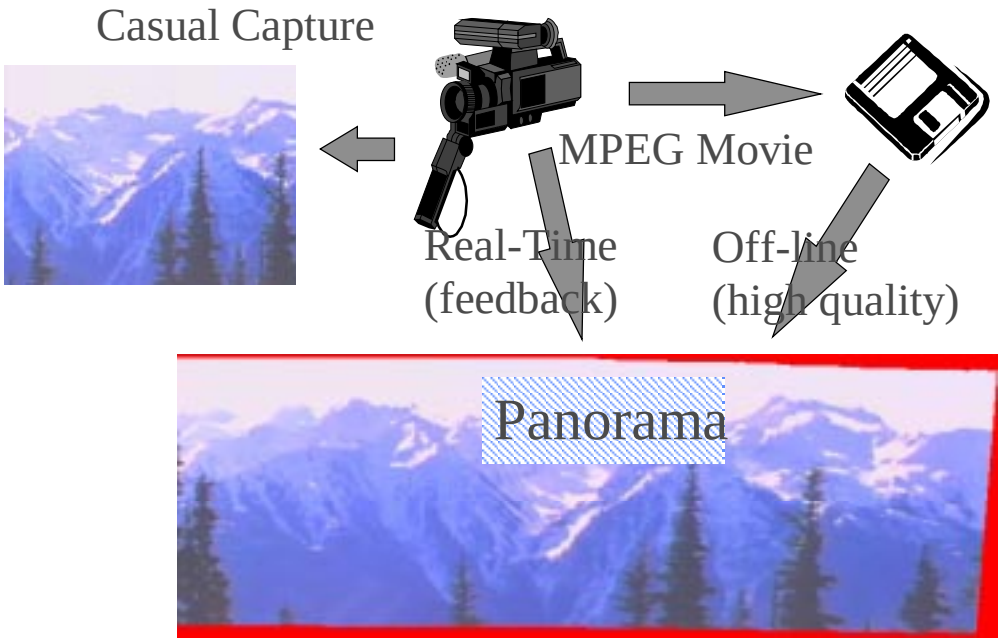

Figure 1 Building panoramas from an MPEG camcorder with the proposed approach

coarse area-correlation method which has been proven inefficient in terms of accuracy. Approximate that they might be, the motion vectors carry a wealth of information about the dynamic of the scene and, in particular, the motion of the camera.

The aim of this work was to investigate how raw, noisy motion vectors can be used to estimate global camera motion and to see whether the results are good enough to find some use in real applications. As a testbed problem, we chose *image mosaicing* (also called *panorama*) which consists of piecing together images taken from different view points (ideally a video sequence where the camera is moved) in order to build a bigger, seamless image (see e.g. [3]). Figure 1 illustrates the kind of application we had in mind.

In order to be in a real context, we have taken as a reference the MPEG-1 streams produced by the HITACHI MP-EG1A (www.hitachi.com) MPEG-1 camera, which features a single chip MPEG-encoder. We deemed this technology representative of what is going to be pervasive in a few years time: all the video capture devices will feature an MPEG single-chip (or on-chip) compressor and video will be, from its source to its consumption, wholly in digital MPEG format. Although we have not yet physically obtained the camera, we have used sequences from the HITACHI WEB site used to advertise the camera.

We believe that following this approach was more significant than using SW-compressed MPEG video streams, where the compression quality - and motion estimation in particular - can me made arbitrarily precise.

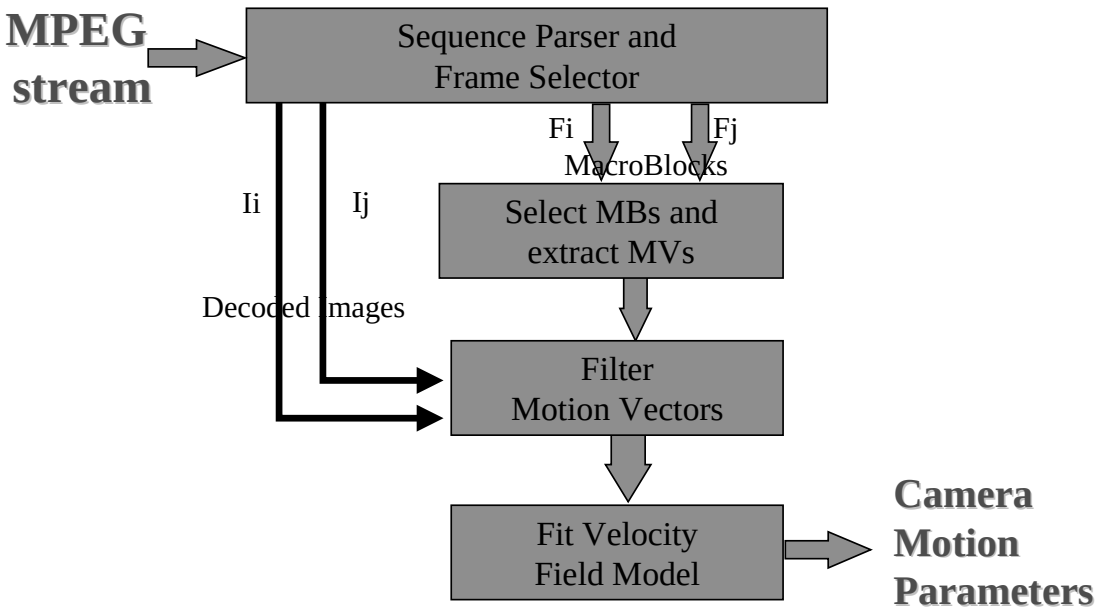

Figure 2 Outline of the proposed motion re-estimation approach

The procedure of estimating camera motion from MPEG raw motion vectors we have called **motion re-estimation**, since motion has already been coarsely estimated during compression and what we try to do is to robustly integrate all the sparse motion estimates into global camera motion.

The structure of the paper is as follows. First we outline the approach and its main components and then describe each of them more in detail. Next we present some experimental results of applying the motion information to the image stitching program developed by Dave Grosvenor [2].
Finally, we highlight problems and possible solutions with the approach and finish with a discussion over applications and intrinsic limitations of the approach.

2 Overview of the approach

The approach we have investigated in this paper to perform motion re-estimation from MPEG-1 video streams is outlined in.

First, the MPEG stream is fed into a sequence parser which “extract” the appropriate frames Fi and Fj that allow to obtain a smooth seamless estimation of the camera motion. This is key phase and will be discussed in detail in Section 3

The frames that have been selected by the parser now bear the displacement in the image plane with respect to a former frame. This information is buried in the macroblocks' motion (displacement) vectors. Not all macroblock hold relevant motion information so only those carrying relevant information are kept; this phase is taken care of by the second stage of Figure 2, which is discussed in Section 4.
Since, depending upon the quality of the image, the motion vectors associated to each motion-informative macroblock could be completely wrong, in the next stage (Section 5) we use the image data Ii, Ij to see whether that information is likely to be reliable or not. In particular, we use a threshold on an "edginess" measure, since the area-correlation matching for edgy image portions is expected to be order of magnitudes better than that of smooth, featureless regions.
In the final stage, discussed in Section 6, we fit a global parametric velocity filed model to the data that survived the previous stages. The parameters, as we will see in the experimental section, can be used to infer the true camera motion. More simply these parameters can be used to warp the images onto a standard reference system, such as a common canvas, to get a seamless panorama.

3 The sequence parser

The sequence parser takes up the role of extracting the proper frames out of the MPEG stream such that a continuos motion estimation can be achieved.

Let us first briefly review the structure of an MPEG stream in order to understand better the requirements and limitations of such a parser. MPEG is a prediction-compensation compression method that tries to exploit spatial and temporal redundancies in image sequences; for our own purposes, what we are more interested is the temporal aspect of the compression, because there is where the information about the camera motion is.
An MPEG stream is composed of a ordered sequence of three types of frames, I, P and B, as shown in Figure 3.
The **I** (intraframe) frames encodes the whole image in a similar way as JPEG does, that is by 8x8 block-wise DCT (Discrete Cosine Transform) and VLC (Variable Length Coding). These kind of frames allow random access to the sequence and are used as "pivots" for the motion prediction. A **P** picture is coded using motion-compensated prediction from a previous **P** or **I** picture, using what is called forward prediction. A **B** picture is coded by using both past and/or future pictures as reference, and thus are called bi-directional. Figure 3 (right) shows the principle of **B** pictures at work. The frames are grouped, at a higher level in Group of Pictures (**GOP**s), which by definition are the frames between two I frames.

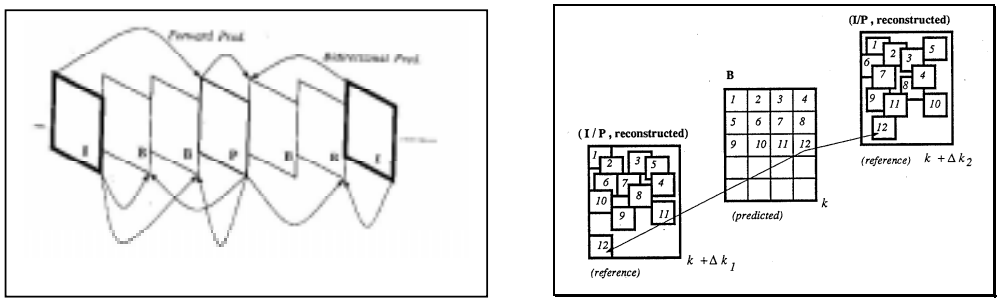

Figure 3 MPEG frames structure and predictive coding of macroblocks

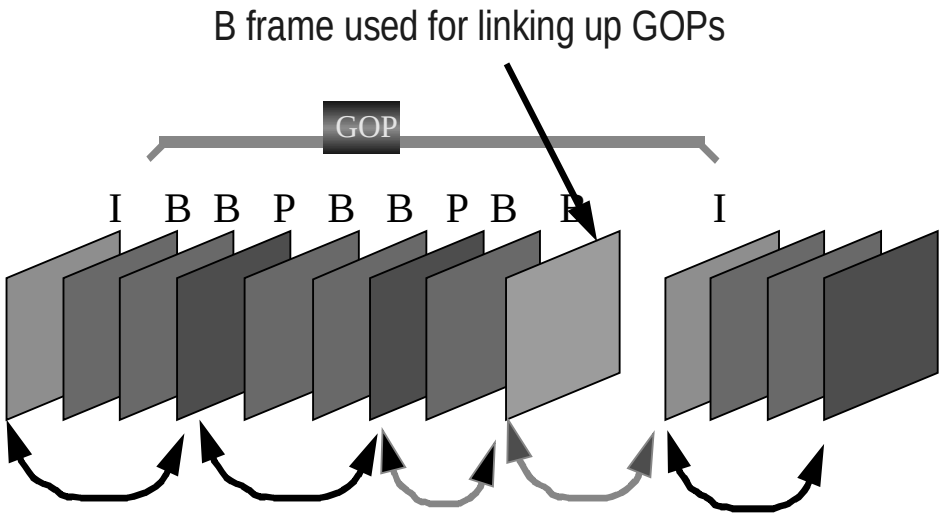

Figure 4 Technique used to chain up motion estimation across GOP boundaries

Thus, in principle, by employing the information available in **P** and **B** blocks, ones should be allowed to chain up frame-to-frame transformations and obtain a continuous motion estimate.

It must be noticed that **B** frames are just "interpolating" frames that hinge on the hard motion information provided in **P** frames and therefore using them for the concatenation of displacements would be redundant.

For our problem of estimating a long chain of incremental displacements, the **GOP** structure poses a problem. At the end of each GOP, the motion estimation is "reset" is some sense, as a new intracoded (**I**) frame is soon to be used and the chain would be inevitably broken.

However, if the MPEG sequence is coded making full use of **B** frames, the last **B** frame of the **GOP** should carry motion information with respect to the previous **P** frame (forward prediction) and to the next **I** frame (backward prediction). In this way, as illustrated in Figure 4, the chain of transformation has become continuous across **GOP**s.

Without this artifice, continuous motion estimation would be impossible. For instance, in a recent work done at MIT [5] which marginally addressed similar

problems, the inherent **GOP** discontinuity problem was avoided by running a traditional feature-based registration method at the end of each **GOP**, which clearly is not convenient. As a matter of fact, the **GOP** continuity is indicated in an MPEG stream header [1], although we have noticed that this bit is often ignored in most SW encoders.

Unfortunately, this approach is not always applicable. Most of the MPEG streams around are in fact wholly intracoded (only **I** frames) or do not have **B** frames. However, since most of the compression efficiency of MPEG is due the use of **P** and, above all, **B** frames, manufacturers of HW encoders have opted for the full deployment of **P** and **B** frames in their latest encoders. As we found out, this is certainly the case of the MPEG encoder used in the HITACHI MP-EG1A camcorder that we have experimented with and more surely still, it is going to be the same for future encoders too.

The parser of the MPEG stream of Figure 2, implements a finite state automata that starting from the first **I** frame activates the successive motion estimation stages each time one of the conditions described above happens: *I)* a new **P** frame *II)* the last **B** frame of a **GOP**, and *III)* the **I** frame marking the start of a **GOP**.

4 Extraction of macroblocks and motion vectors

Once the parser has identified a frame that carries the relevant motion information, the motion vectors have to be extracted from it.
Motion vectors carry the displacement of the current macroblock with respect to a previous reference frame. In the case of **P** frames the reference frame will be a previous **P** or **I** frame, whereas in the case of the last **B** frame of a **GOP** it is the previous **P** frame for the forward motion and the next **I** frame for the backward motion (the motion re-estimation will be carried out twice in this case).
Although **P** and **B** frames are supposed to carry motion information, not all of their blocks do. In fact a macroblock can "downgrade" itself to be as **I** block for **P** frames or either an **I** or **P** block in the case of **B** frames. The classification of macroblocks type is done at the encoding side on criteria of reliability of the motion estimation and encoding efficiency.

For our purpose, when we analyse a **P** frame, we discard **I** block that carry no motion information, and when analysing **B** frames, we not only discard **I** macroblocks but also **P** macroblocks, which do not hold backward motion information to link up with the next **I** frame[2]

[2] To tell the truth, **P** blocks could be used during the first re-estimation with respect to the previous **P** frame, for which these macroblock do carry information. This has however not been implemented.

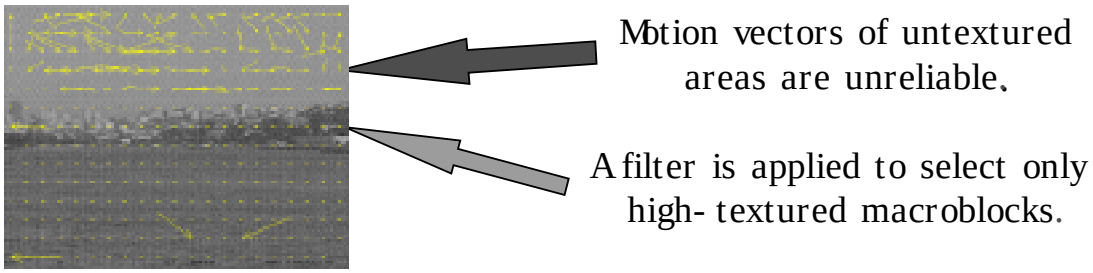

Figure 6 The problem of motion estimation in low-textured areas

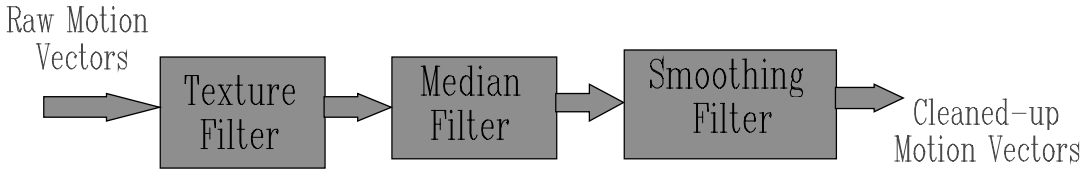

Figure 5 The motion vector filtering pipeline

To come to the point, this stage thus produces the appropriate input data for the next stages, that is a sparse motion vector field, such that shown in Figure 7 (left). This vector field turns out to be extremely noisy and could not be used for the estimation straight away.

5 Motion vector filtering

The motion vector field output by the previous stage leaves lots to be desiderd and therefore some filtering is needed. Although there have been some works on modelling the noise of motion vecotrs for MPEG applications, we felt that those *finesses* are not so important. In fact, in most cases motion vector are not only inaccurate, but is some cases they are completely non-sense data. This would not allow even a sturdy fitting stage to operate reliably.

Figure 5 schematically illustrates the filtering chain applied to the raw motion vectors filters, which is composed of a texture filter, a median filter and an (optional) smoothing filter.

So, to begin, why do we need a texture filter? The origin of the problem can be tracked back to the very nature of prediction-correction coding. Given that the main purpose of MPEG is that of allowing a reasonable rendering quality at high compression rates, the motion estimation can afford to be wrong so long as the errors to correct it are small. In the case of low-textured, uniform areas, the correlation methods used to estimate motion in the first place does not work; however, this does not constitute a problem since it is likely the this will be easily corrected with just a few bits of extra data. This very trade off has allowed encoder to be implemented in HW but clearly from the more *raffine*' image-

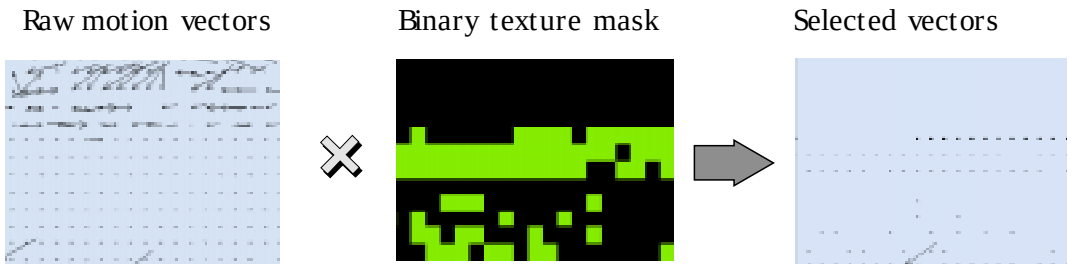

Figure 7 Texture-based filtering of the motion vector field

analysis point of view, the average motion vectors in an MPEG stream are of appalling quality.

Given our purposes, it is better if these low-textured macroblocks are not included in the fitting. For doing so there are several possibilities. The more efficient one would be that of analysing the AC components of the DCT coefficients, thus staying in the compressed domain. The one we have implemented is a cruder one. By relying upon the fact that for the kind of application we have in mind the uncompressed images Ii and Ij (see Figure 2) are also available in parallel, we simply run a convolution mask in each block and count how many pixels have got a gradient above a given threshold. The macroblocks that have too few edgy pixels are deemed low-textured and therefore excluded by further consideration. This method is fairly robust and effectively acts as a binary mask applied to the motion field; Figure 7 shows an example of the initial vector field (left), the mask (centre; black cells represents filtered-out blocks) and the surviving vectors (right). Note how the back cells correspond to low texture regions (such as the sky and part of the sea) in the image of Figure 7 (left).
After having filtered the motion vectors on texture criteria, we run median filtering in order to straighten up single rogue vectors such as that at the bottom of Figure 7 (right) that we found might show up fairly often.
Finally, a simple average smoothing is applied to reduce vector quantization noise correct for vectors that have been median filtered.

6 Velocity field fitting

Once we have sorted out the filtering of the motion vectors in order to eliminate grossly rogue ones and smooth out the others, we have a tidier, even sparser vector field.
Since we have assumed that the camera is moving and the scene is stationary, we can fit a global parametric velocity model the motion vector data.

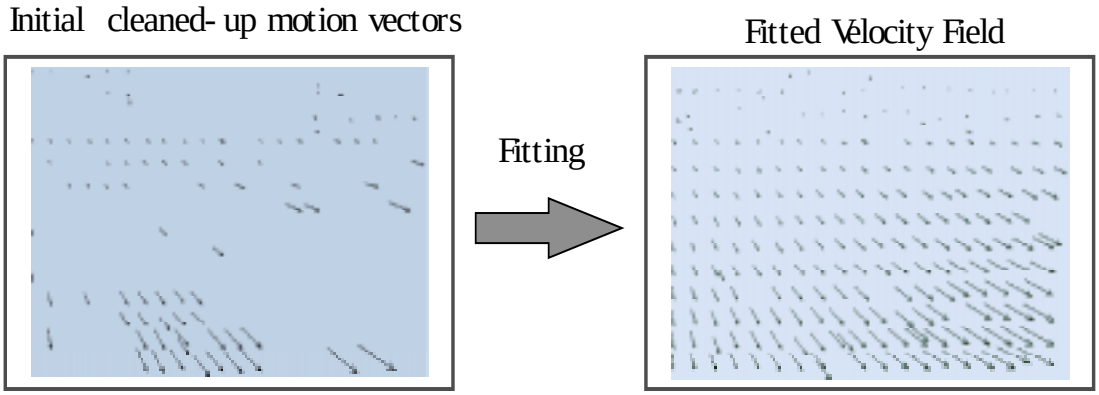

Figure 8 Fitting of the velocity field model

There are several velocity fields that are commonly used in the computer vision literature which were also propose for compression purposes to account for global camera motion at the encoding stage [4].
The most common one is the affine model, given by:

$$V_x = a_1 x + a_2 y + a_3$$
$$V_y = a_4 x + a_5 y + a_6$$

where $a_1, a_2, a_3, a_4, a_5, a_6$ are the model parameters, x, y are the position in the image plane, and V_x, V_y are the velocity components at x, y. The affine model corresponds the to the projection of a planar surface under the motion of an affine camera.
Another common model is the quadratic model, which is given by:

$$V_x = a_1 + a_2 x + a_3 y + a_7 x^2 + a_8 xy$$
$$V_y = a_4 + a_5 x + a_6 y + a_7 xy + a_8 y^2$$

where x, y and V_x, V_y are defined as above and a_7, a_8 are two additional parameters of the vector field model. The quadratic flow is one caused by a plane imaged by a perspective camera.
Although the quadratic model is more accurate in following the true camera motion for close surfaces, the affine model works better in term of noise resilience since it is a lower order model and is less sensitive to noisy data. For this reason, the affine model is the one we have employed in our experiments.
One might argue the affine model (as well as the quadratic one) are not accurate representation of the imaged environment, as it is correct only when the world is flat. However, since we have no information on the geometry of the world and we also assume that the relative distance between the camera and the dimension of the object is high, the affine model has been found the best approximation of global camera motion (see e.g. [5]).

The fitting of the affine velocity model equates to the solution of an overdetermined linear system. In order to solve for the coefficients, we have used the ordinary least square method (OLSQ) by pseudo-inverse [6]. Figure 8 shows an example of filtered and sparse motion vector on the left and the fitted model on the right. More examples will be shown later in terms of mosaiced images.
We have also experimented with a slightly more sophisticated method, namely the iteratively reweighted least squares (*IRWLSQ*), where after the first fitting each datum is weighted with a robust distance norm and then the fitting procedure is run again until convergence is reached. The IRWLSQ method is more robust to outliers in general, but we have not seen dramatic improvements in the results for difficult cases such as those that we will see in Figure 15 and Figure 14.

7 An application with experimental results

As anticipated before, for testing the validity of the method we have used the image mosaicing problem as a testbed.
Image mosaicing has been chosen for two simple reasons. First above all, the approach proposes a method for finding camera motion in the image plane and therefore image mosaicing is the ideal test for it since a good, seamless mosaic is achieved only when motion in the image plane is correctly estimated. Secondly, we had already developed [2] image mosaicing SW and the only thing needed was to bypass the original feature-based method in order to input directly the motion field estimated with the proposed.

Given our target application, the MPEG sequences[3] chosen were those suitable for mosaicing, that is either of panoramas or pan of scenes. Note that it is reasonable to assume that all the sequences were taken without having in mind mosaicing and so can be rightly classified as *casually* captured.

It is important to stress that the motion re-estimation runs near-real time on a workstation and the final mosaic has been built by gradually adding stripes of image to a canvas constituted by the first frame of the sequence.

Figure 9 through Figure 15 show the final mosaic build by using the proposed method to estimate the affine transforms between frames. They have different level of difficulties indeed and the length of the MPEG sequences ranged from 100 to 600 frames.

Figure 9 and Figure 10 are two classic panorama sequences where relevant objects are far away and the camera is slowly panned from right to left and from left to right, respectively. It can be seen that the results are very good. The artefact at their ends, that is the diagonal cut, has been identified as an

[3] Taken off the HITACHI web site

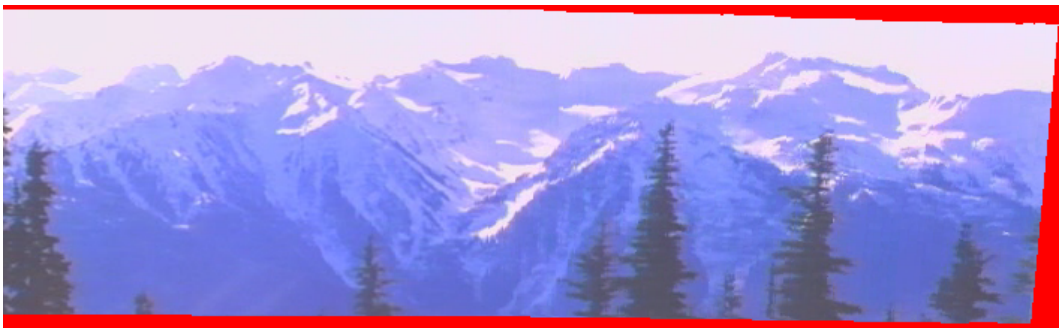

Figure 9 Mosaic of the Mountain Pan sequence

Figure 10 Mosaic of the San Francisco sequence

implementation problem in the image blending SW, so is not due to the motion re-estimation.

Figure 11 and Figure 12 are quite similar to those just described except that are taken from near and, especially for Figure 12, the perspective distortion is evident. However, here too the motion estimation shows to work well throughout the long sequences (hundreds of frames) and, most importantly across the critical **GOP** boundaries (see Section 3).

The example of Figure 13 and Figure 14 are of another class of difficulty. Rather than being just quiet panning sequences, they are jerky untidy aerial sequences. In the first one, the viewpoint goes back to previous positions but our dead-reckoning motion re-estimation is nonetheless immaculate. The second one is a bit trickier since a large part of it is composed by the relatively featureless sea surface and the aim of the camera wandered about crazily and even rotated by 15-20 degrees (it was an helicopter scene). As a consequence the resulting images is slightly striped. Nonetheless, these results would be sufficient to get a feedback on the panorama being built on a hand held device.

The last example in Figure 15 shows another sequence for which the motion re-estimation has worked badly to some degree. The problem here was that the bridge cables were detected as textured areas but constitute a regular pattern

Figure 11 Mosaic of the Root Pan sequence

Figure 12 Mosaic of the Barrel sequence

and, to make things worse, cables are linear features that are notably hard to area-match. As a consequence some wrong motion vectors have not been discarded by the texture filter and, albeit the median filtering helped straighten some isolated outliers, some went through, causing a slightly wrong local estimates that built up frame by frame. Notice that the shrinking of the bridge when going to the right is normal as in reality the bridge is heavily slanted with respect to the camera panning trajectory.

8 Problems with the proposed approach and solutions

The results presented have by no means to be considered definitive. The aim of the work was to investigate if the noisy motion vector could be used for some applications and the answer is clearly yes, but with some reservations.

The texture-based filtering is currently performed by going back into the image domain. Although effective, the solution is not in the spirit of the approach, which is having the whole pipeline in the compressed domain. By following approaches such as that of [9], it would be possible to select highly textured areas just by looking at the AC coefficients of the DCT.

One of the biggest problems is that the linear fitting method described in Section 6 is, by its own nature, very sensitive to outliers, especially if they happen to be leverage data. During the investigation, we have contemplated the use of one of the robust methods that can be found in the literature. The best candidate is in our opinion a method called RANSAC [10], which is classified as a re-sampling method. The RANSAC method would repeatedly draw random sample of 4[4] motion vectors and then chose the subset the yields the best estimate in terms of a distance metric to the other data. We have used this method also to determine the epipolar geometry in another application and it is of proven robustness. We have not tried to apply the method to our particular case but we believe that such and approach might also allow to do without the texture filtering altogether, since RANSAC performs well up to 60-70% of outliers while still keeping a manageable number of random samplings

The last example of Section 7 showed how small local error can build up by using differential estimates of the camera motion frame by frame; all similar methods are plagued by this problem. However, in these cases Kalman filters [11] with a proper error model would help reduce the drifts drastically. The use of Kalman filters is widespread in tracking applications and recursive estimation, which is right the problem we have: a first or second order model could give the estimation some inertia to small isolated error that sometimes so affect the final result.

It is reasonable to assume that for a real application, the method proposed here would be ideal for providing a good feedback but not for producing quality results. As it will be explained in the next section, the most sensible approach would be that of using a two pass approach where the coarsely estimated position of all frames are used to initialise an optimisation based registration method such as that of Peleg [12].

[4] A minimum of four motion vectors is needed to fit the affine optic flow model.

Figure 13 Mosaic of the Sierra Nevada sequence

Figure 14 Mosaic of the San Francisco Aerial sequence

Figure 15 Mosaic of the Golden Gate sequence

9 Discussion and Applications

Apart from some difficulties contingent with the current implementation described in this paper, there are some fundamental limitations with the philosophy of the method, in particular the constraints placed on the MPEG encoder to be **GOP** continuous and with good motion estimation. This should not be a serious problem, though, since if one wants to achieve high compression

rates the motion estimation has to be good. However, even if an encoder is good, there is no guarantee that the conditions are always verified.
This situation typical of a growing research area aiming at adding value to compression methods that are designed to only carry more and more data in fewer and fewer bits. The motto is that if coding is done in a certain way, it is possible to operate in compressed domain and perhaps do 3D reconstruction, object tracking or, to come back to our back yard, do mosaicing.
One could, for instance, envision that there will be a "mosaic-to-come" bit in future MPEG streams that will allow the client to know when a panorama is coming and build it in memory; if the user likes it, he/she could print it out on a photo-quality printer. For doing so, of course, there has to be a co-operating encoder, that would set that bit and (may be) temporally improve the motion estimation quality: this is what we mean by *coding for content*. However, good intentions are not always enough: the rigid MPEG committee rejected a proposal of including parameters such as global camera zoom, horizontal and vertical pan in the standard [2]!

There are alternatives to the method that would not pose limitation on the MPEG stream and work fully in image domain. As far as mosaicing is concerned, recently there has been lots of excitement about a product called VideoBrush (www.sarnoff.com) that does mosaicing from video on a Pentium machine with no HW support. The method uses simple selective correlation to coarsely allign and display frames (for real-time visual feedback) and then a global registration method that takes several seconds to complete in order to obtain the final mosaic. The method's performance is stunning, both in terms of quality and robustness, and does not rely upon MPEG data do operate.

Although the scope of the present work is of a different, exploratory nature, unfavourable comparisons cannot be spared if the application platform that one has in mind is mosaicing on a $3000 Pentium.
But if we shift the focus on adding value to a $500 portable capture appliance such a digital video camera with an MPEG compressor on board, the approach proposed herein is certainly worth considering, least investigating. The approach leverages on the massive industrial effort at producing cut-price MPEG encoders, which no other specific HW used for such application could ever hope for. This hypothetical video camera could have a panorama mode and this method (once properly engineered) would allow a reasonable, though vital, feedback to be produced virtually at no additional HW cost; the high-quality panorama could be then compiled off line, right in the case of the VideoBrush. Note that current digital cameras feed the video data from the CCD/CMOS sensor straight away to the compressor and it is only at rendering stage that the compressed pipeline is broken again. Therefore a image-based method such as that of the VideoBrush would be complicated to integrate.

The method, as many others, has been found wholly unsuitable to mosaic text images. MPEG itself perform bad on text, the reason being that the motion

estimation does not work well. However, although we have experimented with our own sequences, we have not found any MPEG stream captured with the HITACHI MPEG camera we could test the method on.

Besides mosaicing, an interesting potential application is real-time resolution enhancement; however, it is not at all clear at the moment whether the level of accuracy in the motion estimation needed could be ever reached with this method.

10 Conclusions

The method presented in this paper allows to estimate camera motion in the image plane directly from the raw and noisy MPEG motion vectors. The key components of the algorithm are the parsing strategy that allow to overcome GOP discontinuity in the motion estimation, the filtering of motion vectors based on textural information and median filtering, and the fitting of a global velocity model. The method has been applied to image mosaicing and some examples have been presented that show the validity of the method. Extensive discussion sections have given some further insights into the advantages of the approach and its limitation, applications and possible future work.

Acknowledgements

Thanks to David Grosvenor for providing the image mosaicing SW and helping with the integration of the motion re-estimation prototype, and to Adele Lorusso for proof reading this paper.

References

1. ISO-11712: Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to 1.5 Mbits/sec (MPEG1).
2. ISO-13818 Generic Coding of Moving Pictures and Associated Audio (MPEG2).
3. R. Szeliski. Video mosaics for virtual environments. IEEE Computer Graphics and Applications, March 1996.
4. D. Grosvenor Image mosaicing, working paper, HP Laboratories, Bristol, In preparation.
5. S. Mann, R. Picard. Video Orbits of the Projective Group: A New Perspective on Image Mosaicing. MIT Media Lab TR #338, 1995.
6. R.G. Kermode, Andrew B. Lipman. Coding for Content: Enhanced resolution for Coding. IEEE International Conference on Image Processing, Washington, USA, Vol III, pp 460-463, 1995.
7. L. S. Shapiro. Affine Analysis of Image Sequences. University Press, 1996.
8. W.H. Press, B.P. Flannery, S.A.Teukolsky, W.T. Vetterling. Numerical Recipes in C. Cambridge University Press, 1988.
9. J.R. Smith, S.F. Chang. Transform features for textures classification and discrimination in large image databases. IEEE International Conference on Image Processing, pp 407-411, Nov 1994.
10. M.A. Fischler, R.C. Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analsysi and Automated Cartography. Communication of the ACM, Volume 24, Number 6, June 1981.
11. H.F. Durrant-White. Integration, Co-ordination and Control of Multisensor Robot Systems", Kluwer Academic Publishers, 1988.
12. S. Peleg, J. Herman. Panoramic Mosaics by Manifold Projection. IEEE International Conference on Computer Vision, Puerto Rico, pp 338-343, June 1997.