Motion Estimation for Noise Reduction in Historical Films: MPEG Encoding Effects

Fang Chen and David Suter
Electrical and Computer Systems Engineering
Monash University, Melbourne Vic. Australia
Email: fiona@lochard.com.au
d.suter@eng.monash.edu.au

Abstract

Encoding historical films often leads to very poor results as the encoder essentially uses large amounts of the "bits budget" to encode the artefacts as these are abrupt changes in the film. This leaves less bits available to encode the "real parts" of the scene and thus poor reproduction. This paper examines some motion estimation strategies, for motion compensated temporal filtering, and compares the performance in terms of visual quality and in terms of RMS error after MPEG encoding.

1. Introduction

Noise reduction is a major task in processing old historic film. The noise can be defined as a set of pixels whose brightness changes abruptly in temporally and which are spatially distributed randomly (they do not form a set of meaningful geometrical shapes). If we plot the intensity curve in temporal space, of the image brightness at a given spatial position, a noise pixel can be observed as an abrupt transition in the intensity curve. A straightforward approach to reduce noise is to use some kind of temporal averaging filtering techniques to remove these abrupt transitions. However, a fast moving pixel also shows a similar behavior to noise, i.e. the intensities of a fast moving pixel change very sharply during a short time period. The simple averaging techniques will result in a blur or even lost of the motion objects in the restored scene. Thus the question of how to separate the motion effects from noise effects becomes a challenge problem in historical film restoration.

In motion film restoration literature, various attempts have been made to estimate the motion field in the restoration of video sequences. A common one is the block matching approach [1, 2, 3]. The underlying principal of the block matching methods is to make use of the redundant information in motion video to find the similarity between consequence frames. Basically the method divides the image into small blocks and looks for the *best* matching block in the succeeding and preceding frames. The "best match" usually means that the matched pair has a minimum mean absolute or mean square intensity difference.

Under this paradigm, many block-matching based methods have been developed with different searching schemes to estimate motion vector. Rather than using rigid displacements of rectangular blocks, one can use affine distortions of the block in searching succeeding and preceding frames. An alternative set of approaches are based upon a form of Wiener pel-recursive methods [3, 4] that estimate the motion vector by a Taylor series.

1.1. Dual window-searching

The basic block matching approach will always find a match – regardless of how good that match is. In an attempted to reduce the risk of mis-matching one can use a reverse searching validation [5]. We refer this method as a dual window-searching method. In this method, for each block in the current frame, a matched block is found in the target frame. Then a reverse direction searching is applied to the matched target block for its reverse matching. The method further examines whether the reverse matching agrees with the forward matching (within some error limits). If not, the motion vector is assigned to zero (although schemes that try to "fill in" by interpolating neighboring motions are also possible to envisage).

The dual window-searching method, as implemented here, finds matches by two temporal frames in both directions: two succeeding and two preceding frames. Therefore, each block in the current frame has four vectors

– two in forward directions and two in backward directions. For each pixel in the block, a 5 point temporal sequence can be filtered for noise reduction – we use a simple trimmed mean filter (sort the 5 pixels, remove the two ends, average the remaining three).

The dual window-searching method is quite computationally expensive. The method also has limitations in dealing with large motions. (This can be partially improved by using a larger search range, however at greater computational expense, also at greater risk of false matches. Multi-resolution schemes can also help.) In addition the method also fails if there is significant distortion in the moving objects as the blocks are assumed rigid.

1.2. Modified Weiner pel-recursive Method

We extend a conventional gradient based Wiener pel-recursive approach [6] to estimate the four motion vectors. The gradient approach is to estimate the displacement vector by solving a Taylor series expansion involved equation. Consider a basic motion model:

$$S(X,n) = S(X+d, n-1) \qquad (1)$$

where $S(X,n)$ represents the intensity of the pixel *X* in frame *n* and the right term represents the intensity of a pixel located at *X+d* in frame *n-1*. The right term can be expressed by a Taylor's series with respect to a small vector variable *d* and high order error in the Taylor series will converge to zero as d becomes smaller. Solving above equation obtains an estimator of motion vector. This model is very efficient in implement but fails to deal with fast motion objects, i.e. a large *d*, due to limitation of the Taylor series [3].

A pel-recursive solution is often employed to estimate the motion vector in an iterative way. Each iteration produces a new update of the current estimator of the motion vector. The iteration is terminated if the length of the updated vector is less than a predefined size. This recursive modification extends the range of the motion vector to be estimated. A further improvement, Wiener pel-recursive estimator, was made to improve the robustness of the pel-recursive method to noisy signal [4]. The method took account into the influence of higher order terms of Taylor series expansion on the motion vector and attempted to minimize the expected value of the square error between the true update and the estimator.

Compared to the dual window-searching method, the modified Wiener pel-recursive model is a computationally effective approach to the estimation of displacement vector in motion film processing.

1.3. Affine Block-matching Method

An affine transformation can model a greater variety of motion compared to block matching. To some degree, the affine set of deformations can model non-rigid body transformations and perspective distortions. An affine transformation can be defined as:

$$X = AX' + b \qquad (2)$$

where X is the pixel position vector in the current frame and X' is the pixel position in the reference frame, A and b are affine coefficients that define rotation, scaling, skew and translations of the motion object. The affine block-matching method employs equation (2) on each block of the current frame and attempts to find a set of matching pixels in the reference.

However, the affine matching can use its greater flexibility to incorrectly match noise pixels with non-noise pixels: this can be clearly seen in Fig. 1. Another drawback of the affine model is very time consuming as it has 6 degrees of freedom/parameters compared with 2 in the block matching approach.

2. Pixel Classification

In principle, three categories of pixels need to be dealt with: static pixels, moving pixels, and noisy pixels. Our basic difficulty is to avoid treating moving pixels as noisy pixels. If our motion estimation fails for fast motion pixels, such as the fast moving leg shown in Figure 1, then we will incorrectly filter these pixels as if they were noise related.

If we could correctly classify pixels into the three categories then we may be able to avoid some expensive computation (avoid motion estimation for static pixels) as well as improve the filtering. We could accept some mis-classification between moving pixels and noise pixels if the motion compensated filtering of the moving pixels eliminated those that were really noise pixels. It is probably too much to hope for a completely reliable method to a priori classify pixels into these three categories, In this work we search for some statistically motivated heuristics that may be employed to reduce the mistakes and/or reduce the computational cost (with a possible small increase in mistakes).

Statistical analysis tools are employed to determine the threshold levels for pixel classifications. The easiest class to detect would appear to be the static pixels. Their temporal

Fig. 1: Top row: Original frames: Second row: the dual window-searching restorations. Third row: the modified Wiener restorations. Last row: the affine matching restorations. Notice how the dual window and the modified Weiner tend to remove the impulsive noise but also tend to blur the fast moving objects, such as legs in the sequence. The affine block matching is so flexible that even the noise pixels can find matches – the moving objects are not as noticeably blurred though.

trace would be constants in an ideal world. However, frame jitter (frame shake), frame flicker (intensity fluctuations common in historical film) and general digitization and lighting effects, cause the traces deviate from the ideal. Firstly, we try to eliminate much of the

frame shake by calculating frame correlations (in the Fourier domain). This is pretty cheap to compute ad relatively effective. Now, for many of the frames, the majority of the pixels correspond to stationary parts of the scene (in many old movies, due to heavy cameras and unsophisticated camera techniques, there is little zoom, pan or other camera motion). A random sampling procedure is repeated to calculate a set of differences of pixel gray-levels between adjacent frames. The median of these minimum derivations provides a basic level that will be used to classify the static pixels. A scale factor, empirically chosen, is used to scale the deviation level to detect non-static (moving and noisy) pixels. That is, pixels whose inter-frame deviation is less than this factor times the median deviation are classified as static and those above are classified as "moving or noisy".

As stated before, the separation of noisy from moving pixels is the most difficult but most central problem. The heuristic we adopt is that, unless the region is extremely fast moving, then moving pixels will cause a transition that is less impulsive: it will last for more than one frame interval. Thus, what we do, is to temporal filter with a 5 taps and clipped mean is the applied form of the filter. This tends to remove totally impulsive changes but leave a gradual (but smoothed) change where there are non-impulsive changes. We then employ the previously formulated threshold method to define pixels below threshold as noise and above threshold (non-impulsive) as moving. As a further heuristic we take advantage of the fact that motion areas tend to be large and are certainly not isolated pixels – we run morphological open and close operators over the classified motion pixels to remove isolated pixels and remove small holes.

2.1. A Mixed Affine Method

The affine block matching method is ineffective in noise reduction because it is so flexible that noise pixels can find an affine match. Noise effects can be reduced by selecting an error controller, which is used to control some mis-matching, similar in intent to the backward match validation in the dual window-searching method. However, the idea is not really attractive because the affine model is very computationally expensive. Instead of using validation after affine matching, we consider using pixel selection before: we call this a mixed affine method. The mixed affine method is based on above pixel classification. It uses the affine block matching method only on the blocks including moving pixels. For other blocks, the dual window searching method or fast Wiener method, can be used for restoration (remember, the classification does not claim to be perfect – just that we find the majority of the fast moving pixels). This hybrid method greatly improves the visual appearance of individual restored frame, particularly for pictures including large motion figures, such as shown in Figure 2. The computational cost is certainly less than the standard affine method.

2.2. A Mixed Window-searching Method

Another mixed method is proposed by the weighted average of a single window-searching restoration (no backward validation searching) and the simple average restoration. This mixed method uses pixel classification to determine the weight parameters instead of deciding which methods are used in which blocks as in the mixed affine method. The mixed window method can be defined as:

$$S(X,n) = \alpha(X,n)\, S\ (X,n) + (1-\alpha(X,n))\, S_{ave}(X,n) \quad (3)$$

where $\alpha(X,n)$ is a weight parameter, and $S\ (X,n)$ is the restored pixel using the non-validated block matching and $S_{ave}(X,n)$ is the pixel restored by simple averaging. α is set smaller for static and noisy pixels, and larger for moving pixels.

3. Computational Complexity

Among three single models, the affine model is most time consuming one, the modified Wiener model is the cheapest one and the dual window searching model is in between. As for hybrid model, the mixed affine model speeds up the single affine model by applying affine matching only in certain heuristically determined blocks. The increase in speed depends on the number of blocks to be selected, which again depends on the motion nature of the sequence. The mixed window-searching model is cheaper than the single dual window-searching model since only one-way searching is conducted.

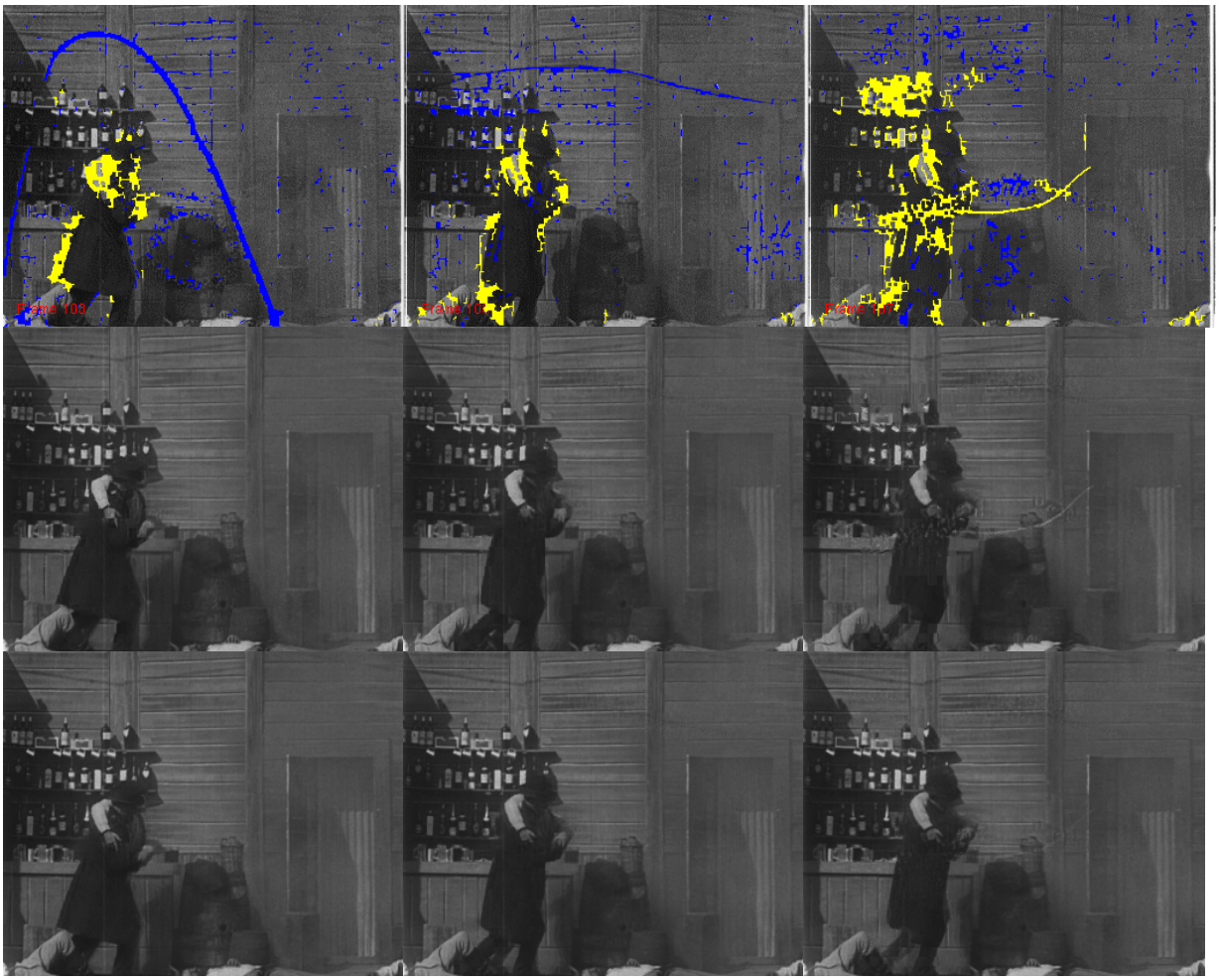

Fig. 2: Top row: motion segmentation, the moving pixels are in yellow (white in monochrome renditions) and noisy pixels are in blue (black in monochrome renditions). Second row: the mixed affine restorations. Last row: the mixed window-searching restorations. Compare row three here with row two of Fig. 1 and row two here with row four of Fig. 1.

Table 1 shows the computational time using different methods and applied to four different sequences. For block affine matching, the scale, shear and shift factors are range from 0.8 to 1.2, -2 to 2 and –20 to 20, respectively. A 8x8 block is chosen in all models and (16x16x h) searching windows are used for the window-searching schemes, where h is 1 for the nearest neighbor frames and 2 for the second nearest neighbors. The run time of the dual window-searching and affine methods depends only on the size of searching window and the ranges of affine coefficients, while other methods depend on the characteristics of individual frames. Of course, one can accelerate an implementation, such as by using Pentium MMX or Pentium III SSE instructions, we have not attempted such hand optimizations.

4. Video coding effects:

Video coding is vital for motion video storage and transmission. This is because motion video contains massive amounts of video and audio signals. There is a subtle interplay between historical film restoration and the effects of subsequent video coding, such as popularly used MPEG coding. To grossly simplify: encoders try to allocate bits to encode changes in a signal. Artefacts in an historical film represent abrupt changes and thus consume valuable bits from the available budget of bits for encoding. This leaves less bits for the rest of the scene and hence reduces the quality. Thus, we find that though our attempts to restore film are not always spectacular when viewed prior to encoding; such attempts can lead to dramatic improvements in

the encoded versions. In this section we examine the video coding effects after pre-processing with our proposed methods of noise reduction. We do this by measuring the storage size after encoding. We also compare the RMS error (average per frame) of the sequence (compared with the frame immediately prior to encoding). This is because a smaller encoded result can be gained simply by "throwing away more information – leading to greater RMS error". Most of our video sequences are in black and white CIF format with 352x288 pixels for luminance and 25 fps of frame rate. We use MPEG-1 compression method in our experiments. Table 2 summarizes the average root-mean square error and the size of the different restorations after MPEG-1 encoding according to set target bitstream of 300 Kbits/s.

5. Conclusion

We have proposed and explored the use of variants of block-matching based methods to estimate motion for historical film restoration. The proposed methods are based on a pixel classification, which segments moving pixels and noisy pixels from static ones. We have presented a statistical filtering approach to classify moving, noisy and stationary pixels. Two hybrid methods have been introduced based on the pixel classifications. The visual performance, the computational costs and the video coding effects of the different restoration methods have been examined.

6. References

[1]. J. Boyce, "Noise reduction of image sequences using adaptive motion compensated frame averaging," IEEE ICASSP, vol. 3, pp 461-464, 1992.

[2]. M. Ghanbari, "The cross-search algorithm for motion estimation," IEEE Transactions on Communications, vol. 38, pp 950-953, 1990.

Table 1. CPU time in seconds per frame for different models.

[3]. A.C. Kokaram, "Motion picture restoration", Springer, 1998.

[4]. J. Biemond, L. Looijenga, D. E. Boekee, and R. H. J. M. Plompen, " A pel-recursive Wiener based displacement estimation algorithm," Signal Processing, vol. 13, pp 399-412, 1987.

[5]. P.Richardson and D. Suter, "Restoration of historical film for digital compression", IEEE Proc. ICIP-95, Vol. II, pp 49-52, 1995.

Table 1: Time comparison for different models

Model	**CPU time (sec / per frame)**
Dual Window	280
Modified Wiener (frame dependent)	Sequence 1: 0.3 Sequence 2: 0.31 Sequence 3: 0.22 Sequence 4: 0.38
Affine matching	8770
Mixed affine (frame dependent)	Sequence 1: 837.64 Sequence 2: 1170.45
Mixed window (frame dependent)	Sequence 1: 27.33 Sequence 2: 28.67 Sequence 3: 20.19 Sequence 4: 28.49

Table 2: MPEG-1 encoding performance comparison. A smaller reconstruction error is achieved for the same size final compressed file. Visually, we confirm that there is a strong correlation – the sequences with lower RMS error tend to look better. (ORI – original, DW – dual window search, MW – modified Wiener, A – affine, MA – mixed affine, MWIN – mixed window search).

Seq.	**Models**	**Ave_rms**	**std**	**Size (Mbyte)**	
				Before encode	**After encode**
1	ORI	10.829	0.783	14.59	0.236392
	DW	8.585	0.839		0.219659
	MW	8.483	0.889		0.219524
	A	10.169	0.778		0.225870
	MA	8.696	0.965		0.219479
	MWIN	8.287	0.919		0.219579
2	ORI	11.727	0.996	13.89	0.233117
	DW	9.594	0.666		0.206695
	MW	9.578	0.691		0.207228
	A	11.374	0.700		0.211998
	MA	11.367	0.707		0.207432
	MWIN	9.499	0.712		0.207080
3	ORI	18.842	0.866	2.58	0.063239
	DW	16.335	0.895		0.059783
	MW	16.308	1.014		0.059642
	MWIN	16.292	1.010		0.059378
4	ORI	16.027	0.611	7.20	0.123477
	DW	14.483	0.711		0.106750
	MW	14.488	0.715		0.107421
	MWIN	14.303	0.748		0.107470

[6]. J. Robbins and A. Netravali, "Image sequence processing and dynamic scene analysis," pp 76-103, Springer-Verlag, 1983.