

## MOTION ESTIMATION WITH THE REDUNDANT WAVELET TRANSFORM.\*

R. DeVore    A. Petukhov    R. Sharpley

Department of Mathematics  
University of South Carolina  
Columbia, SC 29208

### *Abstract*

*We present a fast method for computation of the motion flow based on the redundant Haar transform of reference frames in a video sequence. This algorithm gives a multiscale representation of the motion flow which allows progressive encoding. The high efficiency of the algorithm is achieved by a multiscale correlation between the current and reference frames of the coefficients on each of the Haar blocks. The algorithm is designed to adapt dynamically to make optimal use of channel capacity.*

The weakness of the state of the art encoders lies at two levels. The first is the inability of these encoders to capture motion occurring within their decomposition blocks. The second is their fixed allocation of bits to describe the motion field. More preferable algorithms should dynamically allocate bits to the motion and the residual in a progressive streaming environment. Addressing these deficiencies would have two benefits. Firstly, it would make full and optimal use of channel capacity. Secondly, it would allow users who have access to more bandwidth resources to select bit rates in order to receive higher quality video.

Most video encoders are based on motion compensation. In this case, the bitstream consists of two substreams. One of these contains information about the flow field which is used for predicting the current frame. The second bitstream contains information on the residual image, i.e., on the difference between the actual and predicted frames.

We propose new and fast algorithms (i.e. *Waveflow*) to compute flow fields by using redundant wavelet decompositions. These flow fields can then be progressively encoded and coupled with progressive encoding of the residual in video compression. Waveflow detects motion at all scales and can be used for pixel-wise flow field estimation.

A video is composed of a sequence of frames which we label as  $F_0, F_1, F_2, \dots$ . Each frame is a digitized image and is therefore initially represented as an array of pixel values. The goal of compression is to reduce the number of bits so as to meet channel capacity. The sender (sometimes referred to as the server) has the original video at their

---

\* This research was supported by ONR contract N00014-91-J-1076, DARPA/NSF Grant DMS-0221642, and ARO contract DAAD19-02-1-0028.

disposal. The receiver receives information to reconstruct the compressed frames  $\overline{F_0}, \overline{F_1}, \dots$ . At the point of sending (a compressed version of) frame  $F_k$ , the receiver already has the previous frames  $\overline{F_0}, \overline{F_1}, \dots, \overline{F_{k-1}}$ . This can be used to advantage by the sender in reducing the number of bits to transmit. Namely, the sender transmits only bits that describe how to compute  $\overline{F_k}$  given that  $\overline{F_0}, \overline{F_1}, \dots, \overline{F_{k-1}}$  are known.

Our main question then, is to determine how to capture the detail in  $F_1$ , viewing it as a rearrangement of detail in  $F_0$ , which is already known to us via  $\overline{F_0}$ . Consider a pixel  $P$  in  $F_1$  which corresponds to a region  $I(P)$  in  $F_0$  which has been moved to its new position in  $F_1$ . If this model is correct, we can describe the pixel value  $P$  by simply identifying  $I(P)$ . With this assumption, one can associate to each  $P$  in  $F_1$  a vector  $v_P$  which identifies  $I(P)$ . This collection of vectors  $v_P, P \in F_1$ , is called *the flow field* for the pair  $(F_0, F_1)$ . The vectors  $v_P$  need to be computed for each new frame, which is computationally intensive. Our goal is to design a fast algorithm for computing the flow field.

The two known ideas for computing flow fields are *block motion compensation* and *optic flow*, which we now describe.

Block Motion Compensation is the most common method for computing and encoding the flow field. In this method, the image  $F_1$  is decomposed into blocks of  $8 \times 8$  or  $16 \times 16$  pixels. In its coarsest implementation, motion compensation would assign to each such block one flow vector which is used to describe the motion of every pixel in the block. In other words, the flow vectors are constant on the block. This approach has high compression since one has to encode very few flow vectors, but it suffers from several disadvantages:

1. Visible block artifacts are introduced.
2. Only linear motion of the whole block defined by the motion vector is captured. That is, if an object is moving within a given block, or if a linear deformation of the entire block takes place, this will not be observed in the block motion compensation encoding of the flow vectors.
3. There is no adaptivity in the bit budget assigned for the motion compensation and therefore neither for the residual.

A method to diminish blocking artifacts was used in [2] and is based on the idea of piecewise bilinear approximation of the flow field, instead of piecewise constant approximation used in block motion compensation. For each vertex (corner) of a block, one determines a flow vector. The flow vectors associated to each of the pixels in a given block is then given by a bilinear interpolation to the corner flow vectors. This method provides the desired continuity across blocks and allows one to partially overcome the

disadvantages 2 and 3. However, the method works with fixed bit budget and cannot be used for progressive transmission. At the same time, the motion estimation part of the algorithm works on the block motion compensation principles, having very high computational cost.

Point-wise estimation of the flow field is an important problem in video encoding. One method (so-called, *Optic Flow*) for computing the flow field is based on differential equations [1]. The sender determines a (nonlinear) transport equation which models the flow from  $F_0$  to  $F_1$  and uses this equation to encode the flow. It appears that this method for computing the optic flow field may be numerically intensive and unstable. Moreover, it will most likely encounter difficulties from lack of uniqueness and non-smoothness in the flow field. There remains also the question of whether the formulation of video by partial differential equations with prescribed forms is faithful to the underlying visual process. Nevertheless, this method needs another serious look.

### Redundant Haar Transform

We give a definition of a one dimensional redundant Haar transform which has two main features: 1) invariance with respect to integer shifts; 2) high computational efficiency.

Let  $(x_i)$ ,  $i \in N$ , be a signal (a sequence of real numbers). We define the recursive procedure for the  $L$ -level redundant Haar transform as follows:

- a)  $x_i^0 = x_i$ ;
- b)  $x_i^{k+1} = x_i^k + x_{i+2^k}^k$ ,  $y_i^{k+1} = x_i^k - x_{i+2^k}^k$ ,  $k = 0, 1, \dots, L-1$ .

Given a finite signal ( $x_i = 0$  for  $i < 0$  and  $i \geq M$ ), the implementation of the transform requires the extension of the signal to the exterior of its support. While most wavelet applications use symmetric extensions, these look less attractive for computation of flow fields. Our numerical experiments have shown that the simple extension  $x_i = x_0$  for  $i < 0$  and  $x_i = x_{M-1}$  for  $i \geq M$  leads to much better quality in flow field computations.

We note that Haar coefficients of the simple signal extension can be computed by a slightly different formulae. For example, for the left extension we have  $x_i^k = x_{-2^{k-1}}^k$ , where  $i < -2^{k-1}$ .

To compute a two dimensional redundant Haar transform for the  $(k+1)^{\text{st}}$  level of the decomposition, we must just apply the one dimensional transform to the rows of coefficients at the  $k^{\text{th}}$  level and then to repeat this operation on the resulting columns. For a more precise estimation of the flow field (up to one quarter of a pixel), we use bilinear interpolation of the whole pixel values.

**The computation of the flow field utilizing the redundant Haar transform.**

Our method for the computation of the flow field, based on multilevel redundant Haar transform, is able to estimate inter-frame motion. The main feature of this method is that it works with wavelet coefficients rather than with pixels. This results in a significant increase in computational efficiency when compared with other methods. In combination with a certain smoothing technique incorporated in the algorithm, it leads to a smooth well-compressible flow field.

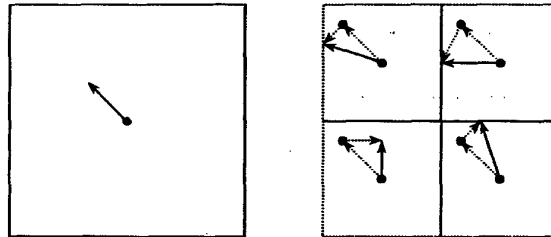
Our present implementations separate the problem of computation from that of compressing (encoding) the flow field. Therefore, our method does not compete in efficiency with the algorithms that exploit a joint computation-compression technique as in [2]. These latter techniques maximize bit allocations by computing an approximation of the motion which can be maximally compressed. The advantage of our method is that it allows fast adaptation to different bit rates and to different channel conditions. Having a good estimation of the flow field, the transmitter can make decisions how to encode the video sequence depending on the required quality and current channel conditions.

The four Haar coefficients corresponding to the same square have very simple geometric meaning. The LL coefficient (L - scale, H - wavelet) is just a mean value of the image intensity inside the square. The LH and HL coefficients give an average value of the gradient. HH characterizes a twist of the surface corresponding to the given square. These reflect the well-known fact that Haar coefficients are a very simple, but efficient, tool for feature detection, in particular, for edge detection. One could imagine that there could be other characteristics (different from Haar coefficients) associated to a square which could extract the essential features of the image on that square necessary for describing the flow field. However, the computational cost for the Haar transform is merely 1 arithmetic operation per coefficient which would certainly be hard to match for alternate methods

To estimate the flow field between frames  $F_0$  and  $F_1$  we compute the regular Haar transform of the frame  $F_1$  and the redundant Haar transform of the frame  $F_0$ . Each dyadic block  $Q$  generated by the wavelet decomposition of  $F_1$  has 4 associated coefficients (1 scaling and 3 wavelet coefficients) at the given level. We start computing the flow field at the coarsest level (the lowest frequency level of the decomposition). For each  $Q$ , our goal is to find the block in  $F_0$  whose associated Haar coefficients has the least deviation from the corresponding coefficients of  $Q$ . Then we process the next level of decomposition and so on (down to pixel-wise estimation), using the motion vectors obtained during the previous step as a prediction for the vectors of the current step. For the lowest frequency level the prediction vectors are taken to be zero.

Generally speaking, the computation of the flow field is an ill-posed problem. In most real video sequences there exist smooth, almost flat, areas of image intensity such that we

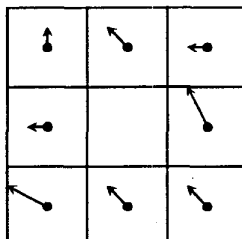
have little chance to assign a reliable motion vector. In this case many vectors can be good candidates and some regularization method must be used. We follow the rule that if there is a choice of which vector to assign to a block, we choose the vector corresponding to a smooth solution. We use two levels of the regularization. The first one provides a smooth transfer between levels (from a coarse level of decomposition to a finer level) while the second provides us with smoothing within the current level.



Now we discuss our procedure for estimating the flow field in more detail. Let us assume that we have already estimated the motion vectors for the  $(k+1)^{\text{st}}$  level of the decomposition. We halve the range of search of that used for the previous level of decomposition. The motion vector  $v^{k+1}$  from the previous level defines the starting value of the search. Given a square  $Q$  at level  $k$ , we determine a vector  $v_0^k$  (associated to this square) which minimizes the functional

$$\| \Delta I(v_0^k) \| + \lambda_L \| v_0^k - v^{k+1} \|. \quad (1)$$

The first term of the functional uses the Euclidean norm of the 4-component vector which is composed of the differences between Haar coefficients of the current dyadic square and those of the redundant transform of the reference frame corresponding to the shift of the dyadic square by the vector  $v_0^k$ . The second term is a penalty for too large a deviation from the predicted vector  $v^{k+1}$ . We note that the above procedure provides a smooth transition from the coarse scale to the fine, however, it does not give a smooth field at the current level of the decomposition. To increase the smoothness inside the current level we use the following additional processing step (iterated 2 times). We start with the vectors consisting of  $v_0^k$  and  $v_{0,a}^k$ , for the eight neighbors of  $Q$ , and introduce an additional term in the functional (1) to penalize the deviation of the desired vector from the weighted average value of the  $v_{0,a}^k$ .



For squares having a common edge with  $Q$  we introduce weights  $\sqrt{2}$  times larger than for squares at the corners. Thus, for the current iteration we are looking for the vector which minimizes the functional

$$\| \Delta I(v_0^k) \| + \lambda_L \| v_{0,a}^k - v^{k+1} \| + \lambda_H \| v_0^k - v_{0,a}^k \|.$$

The weights  $\lambda_L$ , which penalizes deviation from the previous (low frequency) level of decomposition, and  $\lambda_H$ , which penalizes for the violation of the smoothness on the current (high frequency) level, were chosen experimentally. They decrease as the scales become finer.

### Numerical Modeling for Video Coding

While the development of an efficient video codec is not the main goal of this paper, we describe some results on the performance of our method. We incorporated our motion estimation algorithm (WaveFlow) into the codec described in [2]. This wavelet based codec consists of 5 highly optimized modules:

- 1) Discrete wavelet transform with optimized bases generated by rational symbols;
- 2) Encoder of intra-frames (still image encoder);
- 3) Encoder of inter-frames (residual image encoder);
- 4) Motion estimator;
- 5) Motion encoder.

In our experiments we used blocks 1)-3) and replaced the estimator of the flow field in step 4 by our estimator. For encoding the flow field (step5), we used the encoder of step 2). Our scheme outperforms popular formats like MPEG4 or Media Player, however, it gives 20-30% less compression than the codec from [2]. We speculate that the reason for these losses lies in non-optimality of the flow field encoder in step 5. The encoder of intra-frames used in step 2 is adapted to still images whose nature differs from the flow field. The flow field consists of much smoother regions than a regular still image. We believe that customizing the encoder in 5) will improve this component of the compression.

**References**

- [1] C.P. Bernard, "Discrete wavelet analysis for fast optic flow computation," *ACHA* vol.11, pp. 32-63, 2001.
- [2] G. Heising, D. Marpe, H.L. Cycon, and A.P. Petukhov, "Wavelet-based very low bit-rate video coding using image warping and overlapped block motion compensation", *IEEE Proc.- Vision, Image and Signal Processing*, vol. 148, 93 - 101, 2001.