Motion compensation based on Tangent Distance prediction for video compression

Jonathan Fabrizio[1], Séverine Dubuisson[2], Dominique Béréziat[2]

[1] *LRDE-EPITA, 14-16, rue Voltaire, F-94276 Le Kremlin Bicêtre cedex France*
[2] *Université Pierre et Marie Curie, Laboratoire d'Informatique de Paris 6, 4 place Jussieu, Paris 75252 Cedex*

Abstract

We present a new algorithm for motion compensation that uses a motion estimation method based on tangent distance. The method is compared with a Block-Matching based approach in various common situations. Whereas Block-Matching algorithms usually only predict positions of blocks over time, our method also predicts the evolution of pixels into these blocks. The prediction error is then drastically decreased. The method is implemented into the Theora codec proving that this algorithm improves the video codec performances.

Keywords: video compression, motion compensation, tangent distance, Theora

1. Introduction

Video compression refers to reducing the quantity of data used to represent video images. A video is a sequence of frames (or images) that are related along the temporal and spatial dimensions: two consecutive frames might be similar, and the only observed changes are supposed to be due either to the displacements of objects or the camera, to the changes of illumination, or to the noise. In order to reduce the amount of data in image sequences to be transmitted, it is then necessary to determine the spatio-temporal redundancies and to exploit it by defining predictable properties. Considering the data to encode, these properties are used to make predictions, and only the errors between original and predicted data are sent. This technique by itself does not reduce the amount of data (for video compression, we transmit an image of errors, that contains as many pixels as the original image), but, combined with a statistical entropy coding, reduces the data size. In fact, these errors have smaller dynamics than the original pixel values, giving a smaller entropy, resulting in a diminution of the number of bits necessary to encode data.

We distinguish two main prediction types: the temporal prediction and the spatial one.
Spatial, or intraframe prediction, only uses the current frame information: pixels of the frame buffer, sorted into their raster order, are supposed to be similar.

Signal Processing: Image Communication Volumes 27, Number 2, 2012
DOI: 10.1016/j.image.2011.12.001

By only considering pixels previously examined (and thus already coded) in a specific neighborhood, the coder predicts the value of the current pixel. The main difficulty in such approaches is the choice of weighting coefficients for the pixels in the neighborhood. Usually, spatial prediction is adapted to the image content (edges, area, *etc.*). Among the large number of existing spatial predictors, a well known one is the Median Adaptive Predictor (MAP) [24]. MAP selectively uses three linear predictors based on a simple function of surrounding values and gives a good prediction even in the presence of edge features. This predictor has been embedded into the LOCO-I algorithm [37]. Spatial prediction is used in numerous codecs in the spatial domain (H.264/AVC [38, 30]) or in the frequency domain (Theora Codec [8]).
Temporal prediction, also called interframe prediction, uses earlier or/and later frames in the sequence to predict the current frame. Considering a pixel in the current frame, its neighbor pixels in the next and previous frames are assumed to be similar. Motion estimation can be used to eliminate temporal redundancies between frames in a video sequence [9]. Two kinds of motion estimation approaches are used to perform the prediction: (i) in an unidirectional way (front or back), without taking into account occlusion, appearance or disappearance problems; or (ii) in a bidirectional way, by using both backward and forward information in the sequence. This paper focuses on the issue of temporal prediction.

Block-Matching (BM) algorithms are commonly used for motion estimation in most of MPEG [19] implementations as well as in many other encoders. The idea consists, for a given frame, in partitioning the image domain into non overlapping blocks (generally square blocks), and then, for each block, in searching in a reference frame the most similar region over an area near the position of the block. The more similar the regions, the lower the prediction error. A similarity criterion is usually defined as either mean square difference, or mean absolute difference. There are lots of BM algorithms in the literature: surveys can be found in [13, 5, 39], and an interesting empirical comparative study in [20]. Recent works mainly concern the diminution of BM complexity. This is often done by using a specific search technique, such as the most powerful one, called the diamond search [40], or by using an adaptive modeling of blocks. We can also adopt a coarse-to-fine strategy, such as the multi-level approach proposed in [11], where outliers areas are progressively eliminated. In [27], the authors propose to use patterns for motion vector estimation whose size is adapted depending on the context. In [6] a geometry-adaptive block partitioning is used, and a very recent improved version is proposed in [12] that only seeks for a limited number of partitions.

The mesh-based motion model, also called grid wrapping, provides improved interpolation accuracy compared with block-based motion models when the motion field varies smoothly in the spatial domain. The variation of the mesh topology as well as the strategy for coding the synthesis error are defined by an optimization technique following the rate-distortion criterion. The motion is generally modeled by the displacements of the mesh nodes, so that the amount of motion information to be transmitted remains small [2, 35]. In [18] the use of

a mesh-based motion compensated interpolation is shown to give better results than a simple BM. Active meshes [26] can also efficiently represent and code the various regions of the scene and the motion information are also used in temporal prediction. However, they often fail at solving the problem of motion discontinuities (in particular the cases of small objects or occlusions). Recent works try to overcome this problem by refining, as a post-processing step, the mesh node positions when their surrounding patches contain motion edges [15]. In [28], the authors design specific interpolation kernels for mesh-based motion estimation that permit to integrate BM motion vectors into a mesh. In [4] node-point motion vectors of a triangular mesh-based model are estimated using a hierarchical hexagonal refinement algorithm.

Other kinds of approaches used for motion estimation are gradient-based optical flow methods [14]. As they provide a dense vector field which is inefficient for the video compression issue, it is then necessary to reduce the velocity field size and to consider a parametric velocity model, generally chosen constant [21] or affine [3]. The optical flow estimation is then reduced to an over-constrained linear problem that can easily be computed using Least Square method for instance. Parametric optical flow has been successfully applied to video compression [16, 1]. Optical flow as well as BM methods make the hypothesis of a luminance constancy between two consecutive frames, that is, in general, not true in video sequences. To be robust to luminance changes, an adequate luminance model, i.e. a model describing the luminance evolution between two consecutive frames, must then be defined. For example, supposing a constant variation, or an affine one, significantly improves the prediction accuracy and then the compression bit rate. The luminance model can be global [17] over the image domain or locally defined for each block [36, 31]. If robustness to luminance changes has been earlier introduced for the determination of the optical flow [29], it was coupled to BM methods only for the issue of video compression [36, 17, 31].

We have previously proposed in [7] a new motion estimation technique. The purpose of this article is to prove that this contribution improves the motion compensation step if included into a compression scheme: we offer a new approach for motion compensation using a temporal predictor. We revisit the BM algorithm and change the way blocks are matched: we substitute the classic mean square or absolute difference for the tangent distance. Classical BM algorithms only predict block positions over time. By the use of the tangent distance, we not only estimate block positions over time, but also the affine evolution of pixels into these blocks. We demonstrate that tangent distance is equivalent to an affine parametric optical flow method: our method then takes advantages of both approaches (BM algorithm and optical flow). It is also the opportunity to theoretically introduce a luminance model in tangent distance that is robust to local and constant luminance changes.

The organization of the article is as follows. In Section 2, we present video compression by motion compensation using the tangent distance approach. After introducing the tangent distance concept, we show how it can be embedded into a motion compensation framework, justify our choices and show links with

other approaches. We then expose the encoding and decoding schemes. Qualitative and quantitative results are given in Section 3. First, Sections 3.1 and 3.2 specifically focus on the motion compensation step. The prediction robustness is also evaluated in Section 3.3. Then, a complete compression scheme is presented in Section 3.4: we have embedded our method and BM into Theora codec, and compared the coding performances of these two approaches on eight standard video sequences. Concluding remarks and perspectives are finally given in Section 4.

2. Video compression by motion compensation using tangent distance

Motion compensation is split into two main parts: an encoding and a decoding step. During the encoding step, the motion is roughly estimated between the frame to encode, also called current frame, and a reference frame (or more). A current frame is predicted using the estimated motion and the reference frame. Instead of recording the current frame, the estimated motion and the reference frame are encoded. As this estimation is not perfect, errors between the current frame and its prediction are also encoded. BM algorithms are well-known to estimate this motion. In such approaches, the current frame to encode is partitioned into non overlapping blocks and, for each block b, we seek for its more similar region b' in the reference frame, around the position of the original block b: a displacement vector is then estimated for each block.

During the decoding step, the frame is decoded by predicting blocks b using displacement vectors computed during the encoding step and blocks b' of reference frame. This gives a predicted frame, that is refined using the encoded error.

As we can see, this motion compensation approach is performed by only considering block translations, that are obviously approximations of the real motion in most cases. We propose to enhance predictions by estimating other transformations such as affine transformations of pixel patterns or local changes of brightness of patterns. We substitute the standard BM criterion for the tangent distance one. We show this new criterion is equivalent to an affine optical flow constraint method, robust to illumination changes. Thus, the predictor models both block translations and affine transformations inside these blocks. This provides a better prediction and then decreases errors. This predictor is then built using the tangent distance paradigm as described in the next subsection.

2.1. Tangent Distance

The tangent distance allows two patterns (i.e regions of an image) to be compared according to small transformations. Introduced by Simard [34] in the early 90's, it has been used for multiple classification tasks, mainly for character recognition [33, 32], but also for face detection or recognition [23] and for speech recognition [22].

Let $I : \Omega \to \mathbb{R}$ be an image, and s a mapping on I. Mapping s is supposed to be parametric and differentiable with respect to its parameters that are denoted

$\theta = \begin{pmatrix}\theta_1 & \cdots & \theta_q\end{pmatrix}^T \in \Theta \subset \mathbb{R}^q$. The mapping s is also supposed to be identity if $\theta = \vec{0}$, *i.e.* if we have:

$$s(I, \vec{0}) = I \tag{1}$$

The comparison of two images is equivalent to the search of the best mapping between them, and then of the optimal configuration for θ. Given an image I, the manifold $I_s = \{s(I, \theta) \mid \theta \in \Theta\}$ is built and the comparison between images is achieved by computing the distance between their manifolds. As a mapping between images is a non linear function with respect to θ, the problem is approximated by a first order Taylor expansion around $\theta = \vec{0}$:

$$s(I, \theta) \simeq s(I, \vec{0}) + \frac{\partial s}{\partial \theta}(I, \vec{0})^T \theta \tag{2}$$

with $\frac{\partial s}{\partial \theta} = \begin{pmatrix}\frac{\partial s}{\partial \theta_1} & \cdots & \frac{\partial s}{\partial \theta_q}\end{pmatrix}^T$. Let us denote $L_I = \frac{\partial s}{\partial \theta}(I, \vec{0})$, that represents the *tangent plan* to I_s at point $s(I, \vec{0})$, whose components are called *tangent vectors*. Using Eq. (1), Eq. (2) is rewritten as:

$$s(I, \theta) \simeq I + L_I^T \theta \tag{3}$$

In the context of pattern recognition, a distance invariant with respect to a class of transformations is required to efficiently compare two images or patterns. In [33], the authors propose to use the Euclidean distance between the two manifolds generated by the images to be compared:

$$d(I, J) = \text{dist}(I_s, J_s) = \min_{\theta_I, \theta_J} \|s(I, \theta_I) - s(J, \theta_J)\| \tag{4}$$

It is clear that d is null if $J \in I_s$ or if $I \in J_s$ and then invariant to the transformations modeled by s. However, such a distance is very difficult to compute due to its non linearity, that is why it is approximated using Eq. (3) by:

$$d(I, J) \simeq \min_{\theta_I, \theta_J} \|I + L_I \theta_I - J - L_J \theta_J\| \tag{5}$$

As we compute a distance between two tangent plans, d is called *tangent distance*. The computation of d then corresponds to a linear regression. A unique solution may be computed by solving $\frac{\partial \|I + L_I \theta_I - J - L_J \theta_J\|^2}{\partial \theta_i} = 0$ with $j \in \{I, J\}$, leading to the invertible linear system:

$$(L_{JI} L_{II}^{-1} L_{IJ} - L_{JJ})\theta_J = (L_J - L_{JI} L_{II}^{-1} L_I)(J - I) \tag{6}$$

$$(L_{II} - L_{IJ} L_{JJ}^{-1} L_{JI})\theta_I = (L_I - L_{IJ} L_{JJ}^{-1} L_J)(J - I) \tag{7}$$

with $L_{ij} = L_i L_j^T$, $(i, j) \in \{I, J\}$. All matrices are $q \times q$. In practice, q is lower than 10. So the computation of d is direct and fast.

2.2. Choices for s

The transformation s is chosen as $s(I,\theta)(\mathbf{x}) = I \circ t(\mathbf{x},\theta)$, where $t : \Omega \times \Theta \to \Omega$ is a parametric mapping. To respect condition (1), t has to verify $t(\mathbf{x},\vec{0}) = \mathbf{x}$, $\forall \mathbf{x} \in \Omega$. The differential of s at $\theta = \vec{0}$, or tangent plan, is then:

$$L_I = \frac{\partial t}{\partial \theta}(\mathbf{x},\vec{0})^T \nabla I(\mathbf{x}) \tag{8}$$

where $\nabla I(\mathbf{x}) = \begin{pmatrix} I_x(\mathbf{x}) & I_y(\mathbf{x}) \end{pmatrix}^T$ is the gradient of I and I_x the partial derivative w.r.t. x and $\mathbf{x} = (x, y)$. For example, considering the class of rigid transformations (translation and rotation), we have:

$$t(\mathbf{x},\theta) = \begin{pmatrix} (x-a)\cos\alpha - (y-b)\sin\alpha \\ (x-a)\sin\alpha + (y-b)\cos\alpha \end{pmatrix} \tag{9}$$

with $\theta = (a, b, \alpha)^T$ and we then have:

$$\frac{\partial t}{\partial \theta}(\mathbf{x},\vec{0}) = \begin{pmatrix} -1 & 0 \\ 0 & -1 \\ -y & x \end{pmatrix} \tag{10}$$

We finally get for rigid transformations $L_I = \begin{pmatrix} -I_x & -I_y & -yI_x + xI_y \end{pmatrix}^T$. It is also possible to model all affine transformations by adding the case of scale changes. The choice of s will depend on the experimental context or on the user's choices.

In addition to rigid transformations, we consider the constant changes of brightness which frequently occurs between the acquisition of two successive frames. This transformation is defined by $s(I, g) : \mathbf{x} \mapsto I(\mathbf{x}) + g$ and, in this case, we have $L_I = 1$. If we consider the class of rigid transformations and constant change of brightness, the function s would be: $s(I, \theta, g) : \mathbf{x} \mapsto I \circ t(\mathbf{x}, \theta) + g$ and the tangent plan is given by $L_I = \begin{pmatrix} -I_x & -I_y & -yI_x + xI_y & 1 \end{pmatrix}^T$.

2.3. Link with other works

In fact, the linear system defined by Eq. (6) and (7) is over-constrained as it can be rewritten as:

$$\begin{aligned} L_I(I + L_I^T\theta_I - J - L_J^T\theta_J) &= \vec{0} \\ -L_J(I + L_I^T\theta_I - J - L_J^T\theta_J) &= \vec{0} \end{aligned}$$

We can also exhibit other solutions for this system. For instance, by choosing $\theta_J = \vec{0}$ and considering $L_I(I - J + L_I^T\theta_I) = 0$ or equivalently:

$$\theta_I = L_{II}^{-1} L_I (I - J) \tag{11}$$

This means that we consider the distance between the image J and the manifold I_s. The matrix L_{II} is symmetric, positive-definite and then invertible. It is well known that Eq. (11) is the solution of the linear regression problem:

$$\underset{\theta}{\operatorname{argmin}} \|J - I - L_I^T\theta\|^2 \tag{12}$$

If we consider s as a translation of I, i.e. $s(I,\theta)(x,y) = I(x+a, y+b)$, we have $J - I - L_I^T\theta = J - I + aI_x + bI_y$ and Eq. (12) then leads to the method of Lucas *et al* [21], initially proposed for image registration purposes. If affine transformations are considered, Eq. (12) is equivalent to the work proposed in [3] to compute optical flow. Actually, tangent distance gives a geometric interpretation of these methods that consider the problem from an error minimization point of view.

2.4. Encoding

During the encoding step, we have to find, for all blocks in the current frame, their corresponding region in the reference frame. As we said before, the tangent distant can be used to compare efficiently two patterns I and J. During this step, these two patterns are known: J is the block to predict in the current frame and I is a candidate in the reference frame (the two frames are known during encoding). Tangent vectors L_I are obtained from I using Eq. (8). Eq. (11) is used to determine a distance between the candidate region I and the block J to predict. By iterating on all candidates I, we select the distance that best matches with J and then provide an estimation of θ. As J is known during the encoding step but not during the decoding step, we use Eq. (11) instead of Eq. (5). J is required to compute L_J during the decompression process. The solution of Eq. (11) is trivial, simple and fast to compute, because it only consists in a matrix inversion (pseudo-inverse). The encoding step is given in Algorithm 1.

Algorithm 1 Encoding step

{Compute the prediction}
for all blocks b from current frame **do**
  $S_{\min} \Leftarrow +\infty$
  for all possible translations **t do**
    Consider block b' in the previous frame such as $b'(\mathbf{x}+\mathbf{t}) = b(\mathbf{x})$
    Compute tangent vectors $L_{b'}$ of block b'
    Compute $S_{b'} \Leftarrow \min_\theta d(b' + L_{b'}^T\theta, b)$ using Eq. (11)
    if $S_{b'} < S_{\min}$ **then**
      $b'_{\min} \Leftarrow b'$
      $S_{\min} \Leftarrow S_{b'}$
      $\mathbf{t}_{\min} \Leftarrow \mathbf{t}_{b'}$
    end if
  end for
  Record $\mathbf{t}_{\min}$ and the value of θ that minimizes $d(b'_{\min} + L_{\min}^T\theta, b)$
end for
Compute and record prediction error

2.5. Decoding

During the decoding step, the predicted image is generated by combining each block I from the reference frame with its associated transformation trans-

lated by the displacement vector, both computed during encoding step. This means that for each block J, the corresponding block I is given by its translation estimated during the encoding step. Tangent vectors L_I are computed and then, by using the recorded values of θ, the prediction I' of block J is given by:

$$I' = I + L_I^T \theta \tag{13}$$

The decoding step is described in Algorithm 2. This step is only based on the computation of Eq. (13) and can be performed easily in real time.

Algorithm 2 Decoding step

{Generate the prediction}
for all blocks b from current frame **do**
  Select block $b'_{\min}$ from previous frame (using vector $\mathbf{t}$ computed during the encoding step)
  Compute tangent vectors $L_{\min}$ of block $b'_{\min}$
  Generate block $b''_{\min}$, prediction of b, using θ computed and recorded during the encoding step:
  $b''_{\min} \Leftarrow b'_{\min} + L_{\min}^T \theta$
end for
Correct the prediction error

In the next section we give experimental results to prove the efficiency of our method in various common and complex situations. We compare our approach with a classic BM algorithm and prove that our method improves the compression ratio.

3. Results

The proposed temporal predictor has been designed in order to provide better quality results for motion estimation and lower prediction error rates than a standard BM algorithm. Integrated into a complete compression scheme, our tangent distance based motion compensation algorithm should then decrease the size needed to record the prediction error. Compression ratio should also be improved but, to decode an image, all parameters θ are required and must be recorded in the encoded stream: it is then important to take into account the size of parameters required for the decoding step to argue if our approach gives or not better compression ratio. In Subsections 3.1 and 3.2, a theoretical study, based on entropy and focused on the motion compensation step, is first performed. Robustness is then studied in Subsection 3.3. In Subsection 3.4, the both methods are integrated into a real codec, and an experimental study on resulting bit rate and prediction qualities is then performed.

In this section, BM refers to a standard Block-Matching method using mean absolute difference as similarity criterion and TD refers to our tangent distance method.

3.1. Theoretical compression gain

The theoretical compression gain is measured using entropy, that is given, for an image I quantized into n levels, by:

$$H(I) = -\sum_{i=1}^{n} p_i \log_2 (p_i) \tag{14}$$

where p_i , $i = 1, \ldots, n$, is the probability for the i^{th} quantized level to be used (given by the normalized histogram of the pixel intensities). The entropy evaluates the mean size of an image I composed of s pixels: $s \times H(I)$ bits.

To measure the theoretical compression gain, we then consider two successive frames in a sequence, predict the second frame from the first one and compare the entropies given by TD and by BM.

For these experiments and also for the two next subsections 3.2 and 3.3, the block-matching is performed on 8×8 size blocks embedded into 23×23 size search windows. That means $16 \times 16 = 256$ translations are possible considering a 1-pixel precision. Three transformations are modeled in our approach: horizontal and vertical stretches and a constant change of illumination. Other affine transformations could also be modeled (rotations, translations) but implying more information to store. Note that modeling translation in tangent distance, combined with block translations, permits to reach a sub-pixel precision. For their recording, θ component values are rounded to the nearest 0.1.

Experiments are made on `Robot` sequence. Each frame of this sequence (see frames 0, 1 and 12 in Figure 1) is a 512×480 (245760 bytes) gray-level image (1 byte per pixel). This sequence, displaying a zoom, is challenging for the image prediction issue because some of the objects disappear (see for example the barrel on the left, Figure 1). Considering the first frame (Figure 1.(A)) we predict the next one (Figure 1.(B)). The predicted frames given by BM and by TD are shown in Figure 2. Both approaches provide good quality predictions, but artifacts appear on some boundaries and particularly on the top of the barrel on the image predicted by BM (Figure 2.(A)). Figure 3 shows histogram differences between the current frame and its prediction (*e.g.* histograms of prediction error image) computed by the two methods. We also show the result if the motion compensation step is removed *i.e.* if we only encode the frame difference (this is named "Frame difference prediction" in the following tables and figures). Most values of prediction errors given by our method are close to 0, showing we get a high statistical redundancy. A statistical study of these histograms is given in Table 1. If we compare with the results given by BM, we expect our algorithm could give better compression gains because it reduces the quantity of information to store (this will be proved in Section 3.4).

Table 2 gives estimations of the quality of the prediction. We see TD decreases the Mean Square Error (MSE) computed between the frame and its prediction and entropy values computed on the predicted frame, compared to BM results. The theoretical size needed to store prediction errors is 102319 bytes. θ values must also be stored, adding 4921 bytes. This gives 107240 bytes while BM needs 160160 bytes (all these estimations are theoretical and based on

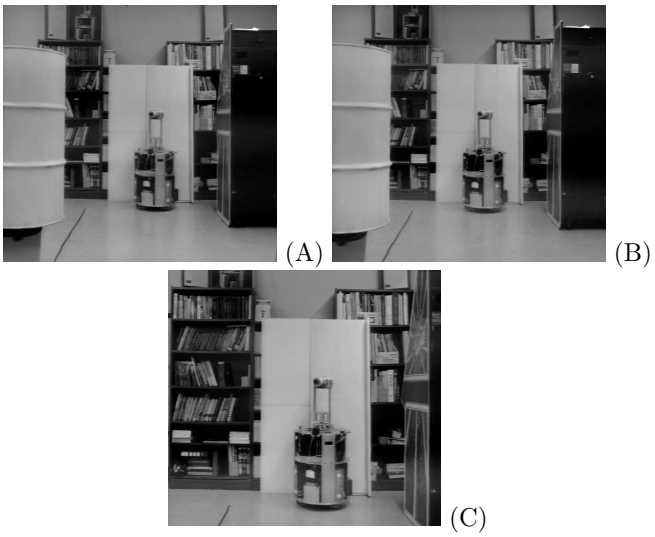


Figure 1: `Robot` sequence. (A) Image 0, (B) Image 1 and (C) Image 12 (note that a part of the image disappears, in particular the barrel on the left).

Table 1: `Robot` sequence. Comparisons of histograms of prediction errors computed between the frame and its prediction: using the previous frame as predictor (Frame difference), a BM-based predictor and a TD-based predictor. μ is the mean value of the histogram, σ its standard deviation

Predictor	Range	number of symbols	μ	σ	number of zeros
Frame difference	$[-254, 188]$	404	0.04	22.97	7866
BM	$[-199, 90]$	250	-0.64	12.05	21689
TD	$[-88, 70]$	125	0	3.75	67292

entropy for both prediction errors and θ values). Note that, as block displacement vectors are equivalent for both algorithms, their size are not taken into account in this computation (having 256 possible positions for a 8×8 block, this will then not add more than 1 byte per block).

Table 2: `Robot` sequence. Comparison between qualities of prediction for three predictors. Our method divides by 10 the MSE.

Predictor	MSE	Entropy	Theoretical size (bits)
Frame difference	527.7	6.15	1511213
BM	145.5	5.21	1281287
TD	14.1	3.33	818557


(A)


(B)

Figure 2: `Robot` sequence. Prediction of image of Figure 1.(B) from image of Figure 1.(A). (A) Using BM, the prediction is correct but some artifacts appear. (B) Using TD, the prediction is better.

The improvement is then significant and the compression gain is important ($\approx 33\%$) with our approach on this sequence.

3.2. Prediction quality

We have compared the quality of prediction obtained with both motion estimation methods on `Foreman` video sequence. The goal of this test is to show the improvement provided by our method in term of motion estimation quality. To this end, each image (except the first one) of the sequence is predicted using the previous predicted one and the prediction error is dropped. The evolution of the Peak Signal-to-Noise Ratio (PSNR) computed between each frame and its prediction can be seen in Figure 4. Note that the PSNR computed for TD is always higher than the BM's one. The reason is simple: in the worst case, values of θ equal 0 and the result of TD and BM are similar. On this sequence, both predictors provide similar performances, except in cases of important variations between frames: BM rapidly decreases its accuracy contrary to TD (see for example between frames 62 and 63, Figures 5.(A-B), when the man suddenly

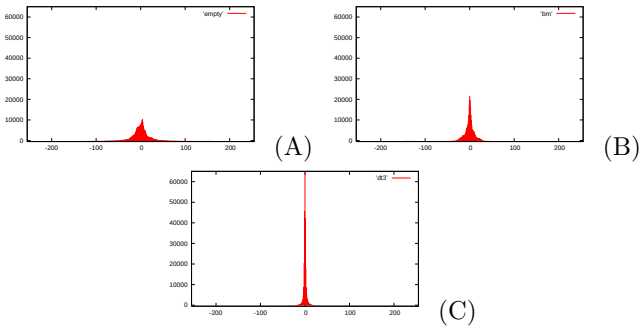


Figure 3: `Robot` sequence. Histograms of prediction errors: (A) frame difference prediction, (B) with a BM-based predictor, and (C) with a TD-based predictor.

opens his mouth). The PSNR of BM decreases from 33 dB to 28 dB while the PSNR of TD stays up to 33 dB. The predicted frame given by both approaches for this particular case can be seen in Figures 5.(C-D).

3.3. Prediction Robustness

In this section, we evaluate the robustness of predictions, by considering various conditions and studying the predictions given by BM and by our approach. We have then generated our own sequences, containing frames of size 704×528, to consider specific conditions and to highlight the abilities of our approach. Assuming a coordinate space with horizontal along the x-axis, vertical along the y-axis and optical axis along the z-axis, we compare BM and TD in following situations: the case of moving objects and the special case of a transparent moving object (Section 3.3.1), the case of a deformable object (Section 3.3.2), the cases of rotation of the camera around the x-axis (Section 3.3.3) and around the y-axis (Section 3.3.4), and finally two cases of large deformations (Section 3.3.5). Note that the case of zoom transformation has already been treated in Section 3.1.

3.3.1. Moving objects and transparent objects

We consider two consecutive frames of a sequence showing objects moving independently (Figure 6.(A-B)) and predict the second frame using the first one. Figure 6.(C-D) shows the prediction obtained with both approaches: we can see the one given by our algorithm are better. BM fails in many ways (see Figure 6.(E) for details): scissors are noisy, pencil is rounded whereas our algorithm gives a visually correct prediction. These results also show that our algorithm deals with transparent objects as it correctly predicts the ruler while BM tries to predict transparent foreground and background, and provides a double blur image (Figure 6.(E)). Table 3 gives quantitative measures of this comparison: the best quality is obtained with our method that significantly reduces the MSE.

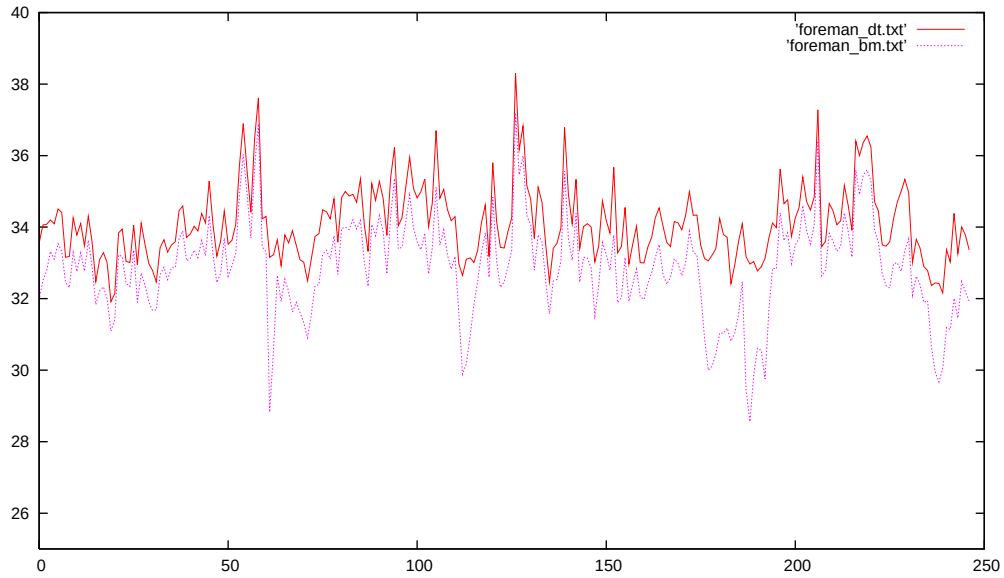


Figure 4: `Foreman` sequence. Evolution of the PSNR values (in dB) for successive predictions. BM (dotted pink line) and TD (plain red line) both give good predictions. However, in case of strong changes between two consecutive frames (like between frames 62 et 63, see Figures 5.(A-B)), our algorithm is more robust.

3.3.2. Deformable objects

The motion of a deformable object is very difficult to predict in a sequence. To evaluate the efficiency of our method on deformable structures, we have tested the case of a moving and deforming hand (Figure 7.(A-B)). The predicted images are given in Table 4. In Figure 7.(C-D), we see TD correctly handles deformable objects while BM can not.

3.3.3. Rotation along the x-axis

This experiment is concerned by the case of a camera rotating along the x-axis. The two considered frames are shown in Figure 8.(A-B) and their pre-

Table 3: Moving objects and a special case of transparent object. Comparison of quality of prediction between Frame difference, BM and TD approaches. The MSE is divided by more than 2 using our method.

Predictor	MSE	entropy	theoretical size (bits)
Frame difference	247.4	5.5	2054323
BM	50.6	4.4	1634224
TD	22.1	4.0	1478496

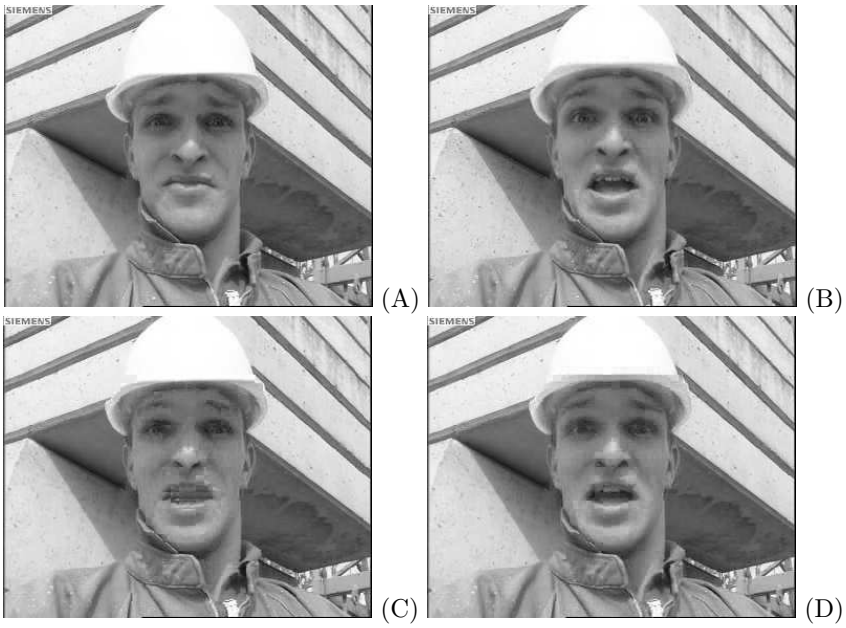


Figure 5: `Foreman` sequence. Frames 62 (A) and 63 (B): the man suddenly opens his mouth. Predictions of frame 63 using BM (C), and using TD (D). Only TD correctly handles the movement: the mouth and the bottom of the hat are very noisy with BM.

dictions in Figure 8.(C-D). We see in Figure 9 that our algorithm gives clearly the best result: BM fails on the checker floor and artifacts appear on the barrier. Qualities of both predictions are given in Table 5. Our method divides MSE by nearly 5 comparing to the one obtained with BM.

3.3.4. Translation along the y-axis

This case is certainly the simplest one and, as expected, the two algorithms succeed. However our algorithm still gives the best result (see Table 6) as the MSE is divided by 2.

Table 4: Deformable object (a hand). Comparison of quality of prediction for a deformable object between Frame difference, BM and TD approaches. The MSE is divided by more than 10 with our method. Note that the resolution of the image is 1350×1072.

Predictor	MSE	entropy	theoretical size (bits)
Frame difference	269.1	5.2	7480553
BM	102.2	4.4	6355108
TD	8.6	3.2	4666871

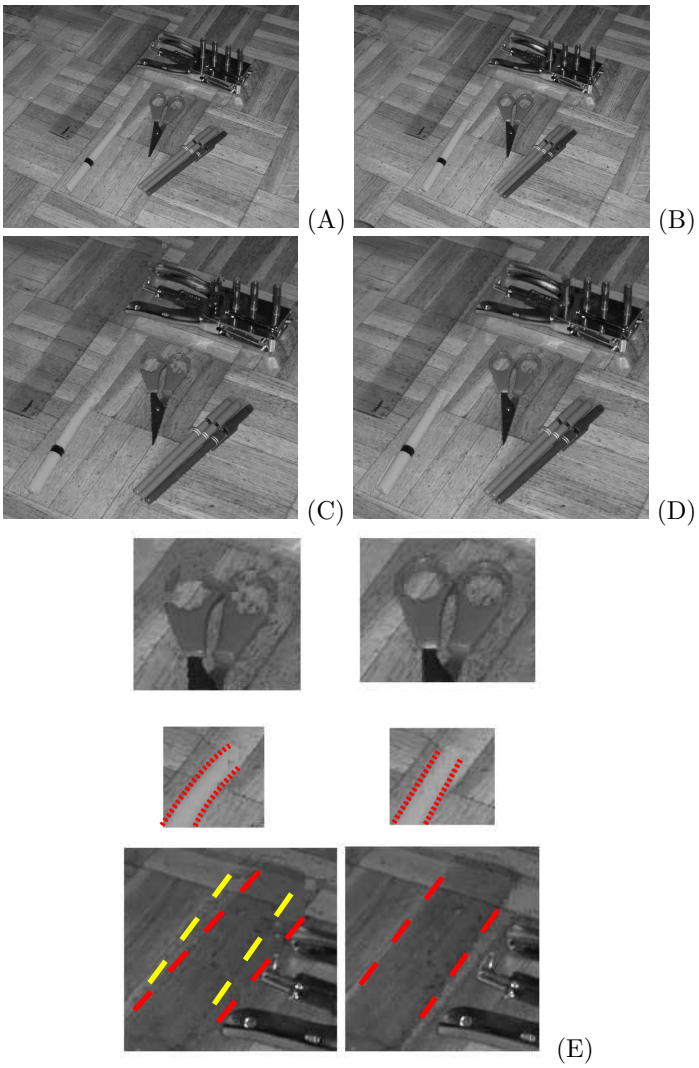


Figure 6: Moving objects and a special case of transparent object (the ruler). (A-B) Two consecutive frames. (C) Prediction using BM. (D) Prediction using TD. (E) More details for comparison between predictions given by BM (left column) and TD (right column): many artifacts appear with BM (scissors are noisy, pencil is rounded) while TD is visually correct. Our algorithm can also deal with transparent objects: the ruler is correctly predicted with our approach while BM predicts two superposed rulers.

3.3.5. Cases of large deformations

The robustness of both approaches have been checked by increasing the time between the two considered frames on `Robot` sequence: we try to predict image 12 (Figure 1.(C)) from image 0 (Figure 1.(A)). This is a difficult situation

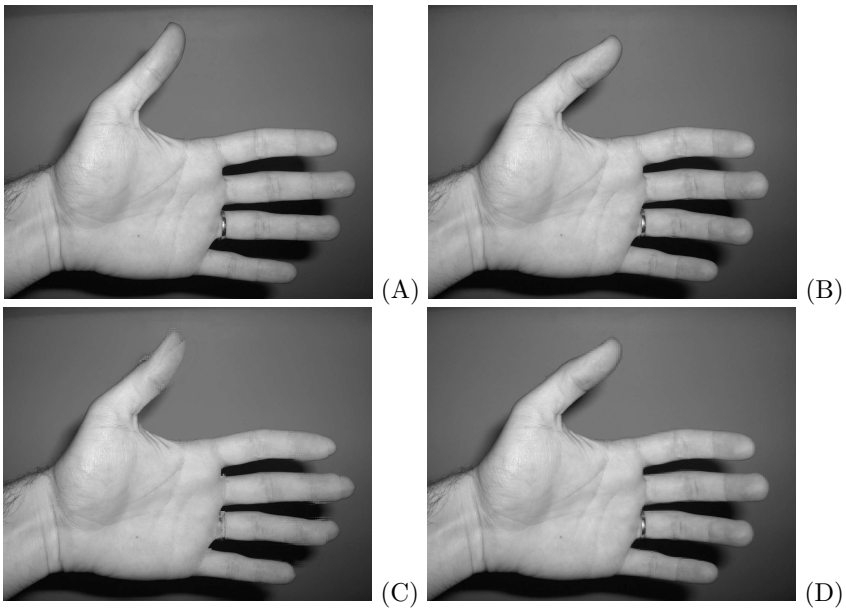


Figure 7: Deformable object (a hand). (A) and (B) Two consecutive frames of the sequence. (C) Prediction using BM. (D) Prediction using TD. Fingers are noisy and deformed with BM approach but not with our method.

Table 5: Rotation along the x-axis of the camera: comparison of prediction quality.

Predictor	MSE	entropy	theoretical size (bits)
Frame difference	779.1	6.4	2377104
BM	190.2	5.4	2003655
TD	42.8	4.6	1697992

because the barrel on the left disappears between these two frames and the size of the robot increases. Figure 10 shows the obtained predictions. While TD provides a correct prediction, BM algorithm completely fails: the barrel is still visible and the image prediction is noisy. Data on Table 7 quantifies the quality of the predictions and shows the robustness of our method.

We have also performed a test in extreme conditions. We have taken two images totally different, and tried to predict one from the other. The initial image is displayed in Figure 9.(A) and as final one in Figure 6.(A). Figure 11 shows prediction results and Table 8 gives quantitative results. Even in such extreme conditions, our algorithm provides good predictions. It is important to understand the meaning of this test. For the motion estimation issue, it is crucial to match the best possible corresponding blocks over time, and bad associations are mistakes. For the compression issue using motion compensation

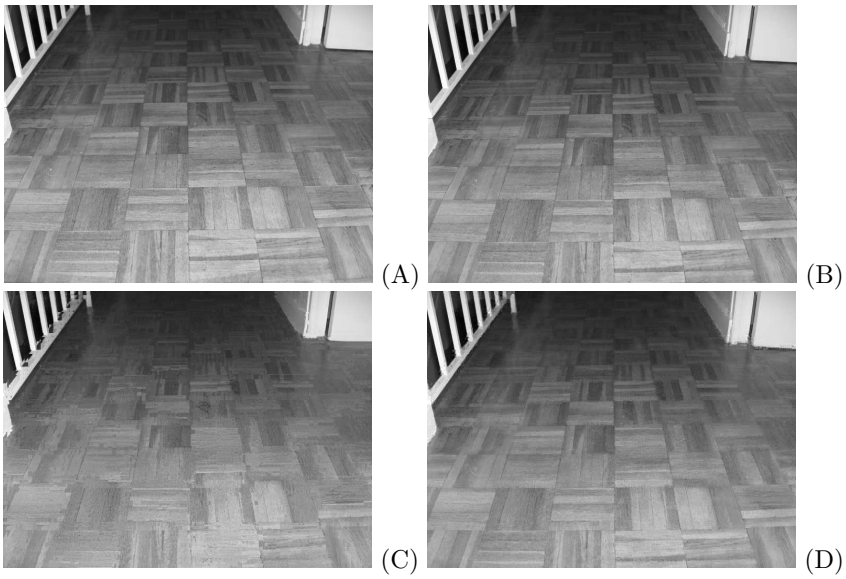


Figure 8: Rotation of the camera of the camera around x-axis. (A-B) Two consecutive frames. Prediction of frame (B): (C) using BM, and (D) using TD.

Table 6: Rotation along the y-axis of the camera: comparison of prediction quality given by Frame difference, BM and our approach.

Predictor	MSE	entropy	theoretical size (bits)
Frame difference	773	5.2	1935827
BM	19	3.3	1225064
TD	9.1	2.8	857932

we just need to find the block associations minimizing the errors (and then the encoding size) but do not need perfect associations. During this complex test, the two images are different and it is not possible to perfectly match blocks between them. The proposed method generates a good prediction as it finds similar blocks and predicts a block with its not so similar matched block. This situation arises, for example, when a sequence is encoded with intra-frames introduced regularly (as in Section 3.4). This means that we could have a major change between images without necessarily needing a new golden frame. It happens, for example, between frames 97 and 98 of `Tennis` sequence because the point of view of the camera changes (see first line of Figure 12). If we compare predictions of frame 98 with BM-based motion compensation and with TD-based motion compensation, the difference is huge (Figure 12, bottom line). With the implementation in Theora codec of both prediction algorithms (see subsection 3.4), the size of the image predicted using BM algorithm is 58111

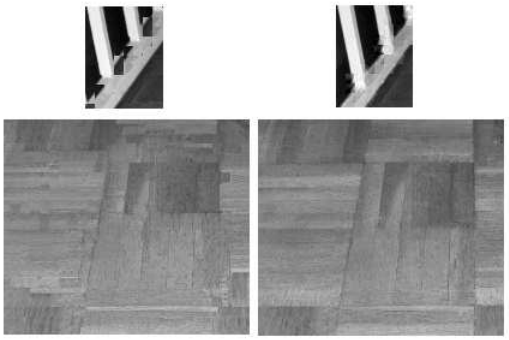
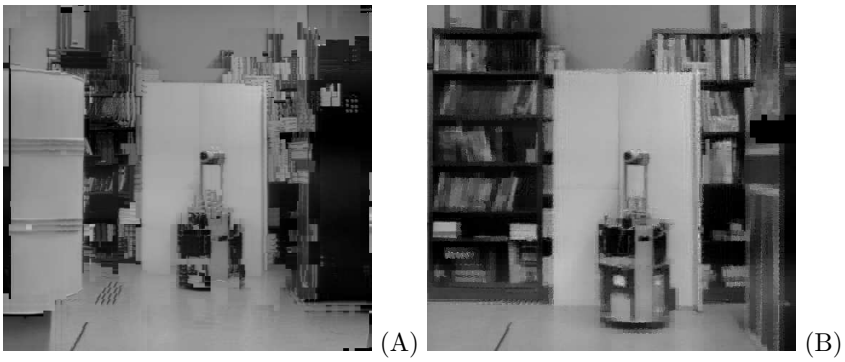


Figure 9: Rotation of the camera around x-axis: many artifacts appear with BM (left) while TD (right) is visually correct.



(A) (B)

Figure 10: Prediction of image Figure 1.(C) from image Figure 1.(A). (A) With BM, the quality of the predicted image is very poor, and very different from the real one. (B) With TD, the quality is acceptable (notice the barrel on left disappears).

Figure 11: Prediction of image in Figure 6.(A) using image of Figure 9.(A). Prediction using BM (left): the image is unusable, and using TD (right): the quality of the prediction is acceptable.

Table 7: Study of robustness: comparison of prediction quality.

Predictor	MSE	entropy	theoretical size (bits)
Frame difference	4919.5	8.0	1966139
BM	3094	7.5	1847711
TD	189.5	5.1	1243822

Table 8: Prediction of an image using a totally different one: comparison of prediction quality.

Predictor	MSE	entropy	theoretical size (bits)
Frame difference	2106	7.4	2751139
BM	1175	7	2599123
TD	82.4	5	1879420

bytes while it is 56955 bytes with TD (more than 1 Kb saved). DCT coefficients of prediction error need 55671 bytes with BM algorithm while they need only 48114 bytes with TD.

Figure 12: `Tennis` sequence. Top: Frames 97 and 98 (the point of view of the camera changes). Bottom: Predictions of frame 98 using Theora codecs with BM based motion compensation (left) and with TD based motion compensation (right).

We conclude our approach well-performs BM and highly improves the quality

of predictions in a lot of situations. This efficiency may lead to reduce the number of reference frames needed in the video sequence. This is another way to improve compression rate.

3.4. Compression gain into an operational codec

In Subsection 3.1, we have shown the proposed approach provides a significant theoretical compression gain. However, common codecs implement more than a simple entropy coding in the spatial domain (use the frequency domain, quantization, RLE, *etc*) and this theoretical gain may not be reachable in practice. To ensure our method provides an effective compression gain, an implementation into a codec is then necessary and results must be quantified. The purpose of this subsection is then to show our method is more robust than a classic BM-based method, and to quantify this robustness in term of compression rate.

3.4.1. The codec: Theora

We have chosen the free Theora codec [8] supported by the Xiph.Org Foundation[1]. This choice is motivated by the following reasons:

- it is free,
- it does not implement too much features (like all H.264/AVC features), that permits to better quantify our motion compensation algorithm benefices,
- it becomes more and more popular. Numerous web browsers have a native Theora decoder (as Theora was suggested for HTML 5 standard). Some major websites (Wikipedia, *etc.*) or online video servers (Dailymotion, *etc.*), provide their video content by the use of Theora.

Classically, Theora codec uses two kinds of frames: intra-frames (or golden frames) and inter-frames. Intra-frames are directly recorded while inter-frames are predicted from a reference frame and only the prediction error is recorded. The color space is YUV and each channel is partitioned into non overlapping 8×8 blocks. A value in a channel has 1-byte size. Inter-frames are predicted from a previous reference frame (forward prediction) which can be the last inter-frame or the last intra-frame: the reference block is retrieved in the previous frame from a block translation ranging up to $(\pm 15, \pm 15)$ with an accuracy of 0.5 pixel. Note that H.264/AVC can perform forward or backward predictions while Theora is restricted to forward predictions. The error between the predicted frame and the current frame is stored and encoded in the same way than intra-frames using a JPEG-like method (see Figure 13): a Discrete Cosine Transform (DCT) is performed on each block and then quantized. The first DCT coefficient (DC) is corrected by the prediction from DC of surrounding blocks. Finally blocks are encoded using RLE and Huffman tables.

[1] `http://www.xiph.org`

To decode a frame, the inverse principle is applied with an additional "deblocking filter" devoted to remove various undesirable effects induced by the block encoding method.

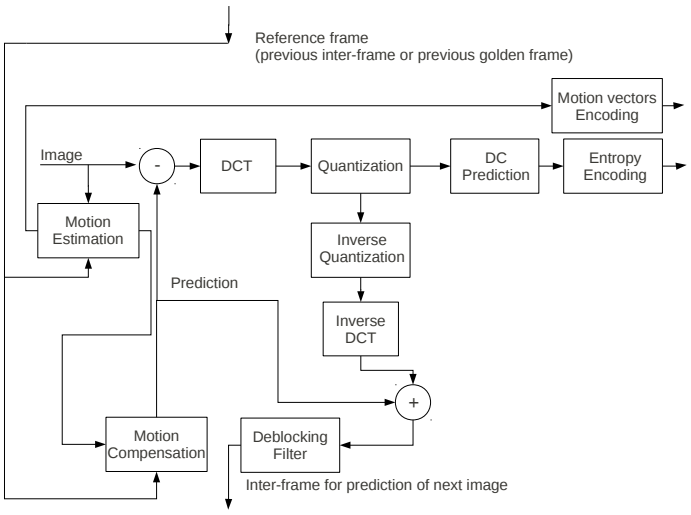


Figure 13: Theora inter-frame encoding.

We have implemented Theora codec (both encoder and decoder) according to its specifications [8]. We have derived this implementation into two versions, each one having its own motion estimation stage: the first one integrates a BM-based predictor and the second one our TD-based predictor. The BM-based predictor uses the mean absolute difference as criterion with an exhaustive search. The codec parameters (Huffman table, block filter parameters, *etc*) are taken from the VP3 codec [25, 8]. As we focus on motion, all blocks are recorded using motion compensation. These two versions allow to make relevant comparisons in the strictly same conditions.

3.4.2. Implementation of the TD-based predictor into the codec

To implement our method into the Theora codec, some strategic choices have been made to store the motion parameters θ. Even if a good encoding strategy is important, this is out of the scope of this article to discuss the optimal way to store this information.

Non-zero values of θ (all θ values are rounded to the nearest 0.1) are recorded as follows: the integer part is given by a static Huffman code, the decimal part by another Huffman code and the sign is one bit coded. Sometimes our predictor can provide a better compression gain than BM's one, but not high enough to allow a competitive storage of θ parameters. In such cases we set $\theta = 0$, that decreases the storage size. To store zero values of θ, we use a RLE encoder (described in Section 7.8.1 of the specification of Theora [8]) that indicates which of the coded blocks have θ values set to zero.

We do not claim we made optimal choices and we think it could be improved (for example, using an arithmetic coding or BZ2 algorithm). However, if we succeed with this very simple code, this is reasonable to hope better results with more sophisticated codes.

To decide if tangent distance parameters must be kept or set to zero, we have to design a relevant criterion. A first idea could be to compare the entropy of prediction errors obtained with the original θ values and with θ values set to zero: if and only if the difference is big enough, we keep θ values. However, this is not a good strategy, because the comparison is performed into the spatial domain, while the encoding is performed into the frequency domain. In fact, a given block might be similar to a block into the spatial domain, but closer to another one in the frequency domain or generate a lower prediction error. It is then impossible to predict the gain in the encoding size of the block by only comparing blocks into the spatial domain because there is a lot of other factors that influence the encoding size such as the quantization or the RLE encoding of successive zeros.

The interest of using (or not) θ values for a specific block is evaluated by comparing predicted blocks into the frequency domain after the quantization step. Two parameters influence the compression gain, and then have to be considered to compare predictions: the similarity between two quantized DCT blocks and their number of zeros. We propose a three-term score associated to the prediction and defined as follows.

The first term is related to the number Z of zeros contained in the 8×8 quantized prediction error block into the frequency domain. This term is then defined by $k_Z(64 - Z)$: a good prediction, with a lot of zeros, will then provide a low score.

To penalize prediction errors with high amplitude values in the frequency domain we add two other terms to the score. Let E_{DC} be the absolute error between the DC coefficient of a block and the one of the prediction. Let E_{AC} be the cumulative absolute error between the other DCT coefficients of a block and the ones of the prediction. The score is:

$$S_1 = k_Z(64 - Z) + k_{\mathrm{DC}}E_{\mathrm{DC}} + k_{\mathrm{AC}}E_{\mathrm{AC}}^2 \tag{15}$$

with k_Z, k_{DC} and k_{AC} weighting coefficients.

This score is not fully satisfactory because the spatial domain is not taken into account. We then add the parameter E_{IMA}, corresponding to the absolute error into the spatial domain. This gives a new score formulation:

$$S_2 = S_1 + k_{\mathrm{IMA}}E_{\mathrm{IMA}}^2 \tag{16}$$

with k_{IMA} a weighting coefficient.

To choose the block predicted either with TD, or with all θ set to 0, we also add a parameter N_θ, corresponding to the number of θ values to encode, and consider that adding one θ value is as penalizing as removing two zeros to the block into the frequency domain. The score function finally becomes:

$$S_3 = S_2 + N_\theta \times 2k_Z \tag{17}$$

If the TD predictor decreases the score S_3, θ parameters are kept, otherwise they are set to zero.

During our experimental tests, we have noticed that taking into account the frequency domain (and not only the spatial domain) during the block comparison greatly improves the choice of the block as the prediction error is significantly decreased. For this reason, S_3 is used as criterion to select the final predicted block using θ parameters computed by the TD motion compensation. S_3 is then used for both selecting the best matching block and deciding if the θ values are kept.

3.4.3. Effective compression gain

In this subsection, we compare the efficiency of the first codec (with a BM-based prediction algorithm) and of the second codec (with a TD-based prediction method).

For the TD approach, we have modeled both horizontal and vertical translations and illumination changes. This means we have, for each block, three θ values to encode. The score function is computed with $k_Z = 2$, $k_{\mathrm{DC}} = 0.1$, $k_{\mathrm{AC}} = 4.44 \times 10^{-5}$ and $k_{\mathrm{IMA}} = 4.26 \times 10^{-5}$ (empirically fixed). Experiments are done on height well-known video sequences often used to test codecs: `Coastguard` (352×288), `Container` (352×288), `Football` (352×240), `Foreman` (352×288), `Garden` (352×240), `Hall monitor` (352×288), `Mobile` (352×240) and `Tennis` (352×240).

To evaluate the performances of TD and BM for several given bit rate values, we compute the rate-distortion (RD) curves of each method and compare them using the Bjontegaard's metric [10]. The principle consists in computing the PSNR and bit rate values for several qualities (Quantification Points - QP) from which two RD-curves are interpolated: one for each approach to compare. The difference of area Δ between these curves corresponds to the Bjontegaard's metric. We then have computed PSNR and bit rate values for seven QP values: $QP = \{0, 10, 20, 30, 40, 50, 60\}$, and the RD-curves for the eight video sequences are shown in Figure 14. We define $\Delta\mathrm{SNR} = \mathcal{A}^{\mathrm{TD}}_{\mathrm{SNR}} - \mathcal{A}^{\mathrm{BM}}_{\mathrm{SNR}}$ where $\mathcal{A}_{\mathrm{SNR}}$ is the RD-curve area of SNR for a given method. Similarly, we define $\Delta\mathrm{BR} = \mathcal{A}^{\mathrm{TD}}_{\mathrm{BR}} - \mathcal{A}^{\mathrm{BM}}_{\mathrm{BR}}$. This differences of PSNR values (ΔSNR) and percentages of bit rate reduction (ΔBR) are presented in Table 9 for four intervals of QP: the full interval $[0 - 60]$, the lowest quantification interval $[0 - 20]$, the medium one $[20 - 40]$ and the highest one $[40 - 60]$.

As it can be computed from Table 9, the average performance of TD is better than the one provided by BM: we get an average bit rate saving of 4.92 % and an average increasing of the PSNR of 0.43 dB. In particular, our scheme performs much better for `Hall Monitor` sequence, for which we get an average bit rate saving of 11.29 % and an average increasing of the PSNR of 1.2 dB, or `Container` sequence, for which we get an average bit rate saving of 11.76 % and an average increasing of the PSNR of 1.39 dB.

Globally, our approach outperforms BM for lower quantification rates, reaching an average rate saving of almost 10 % and an average increasing of the PSNR of 1.3 dB. This is particularly significant for the complex sequences `Foreman` and

`Tennis` that present high spatial details and large amount of movements. We compute an average bit rate saving and a PSNR average increasing of respectively 12.24 % and 1.23 dB for `Foreman` sequence and respectively 11.35 % and 0.86 dB for `Tennis` sequence.

However, we remark TD has similar performances on sequence `Garden` than BM. This is probably due to the simple translation motion present in this sequence, that is still well handled by the default sub-pixelic predictor. Ou approach also gives just a little better results on `Football` and `Mobile` sequences.

Table 9: Comparison in terms of differences of PSNR (ΔSNR, expressed in dB) and of bit rate saving (ΔBR, expressed in %) between TD and BM approaches, using Bjontegaard's metric [10], for eight video sequences. The computations were made for four intervals of quantification points.

QP interval	$[0-60]$		$[0-20]$		$[20-40]$		$[40-60]$	
	ΔSNR	ΔBR	ΔSNR	ΔBR	ΔSNR	ΔBR	ΔSNR	ΔBR
`Coastguard`	+0.11	-3.3	+0.61	-11.96	-0.06	+0.83	-0.07	+0.31
`Container`	+1.39	-11.76	+3.89	-18.54	+1.48	-14.72	+0.36	-3.57
`Football`	+0.03	-1.52	+0.14	-5.56	-0.08	+2.58	+0.05	-1.50
`Foreman`	+0.30	-4.02	+1.23	-12.24	+0.03	-1.43	+0.07	-0.50
`Garden`	+0.02	-0.57	+0.01	-0.89	+0.01	-0.05	+0.05	-0.70
`Hall Monitor`	+1.20	-11.29	+3.64	-15.35	+0.74	-13.07	+0.45	-5.24
`Mobile`	+0.11	-2.43	+0.10	-4.09	+0.08	-0.87	+0.17	-2.42
`Tennis`	+0.28	-4.53	+0.86	-11.35	+0.20	-4.85	+0.06	-0.10
Mean	+0.43	-4.92	+1.30	-9.99	+0.30	-3.94	+0.14	-1.71

3.4.4. Type and amount of data to transmit

We have seen our algorithm improves bit rate performances of Theora. In this subsection, we now compare the amount of data necessary to be transmitted to the decoder for both approaches using the experimental conditions described in Subsection 3.4.3. In particular, we measure, for some video sequences, the bit rates of the prediction error and of the motion vectors (for both approaches), and also of the θ parameters (for our approach). Results are shown in Figures 15 and 16.

For `Container` and `Hall Monitor` sequences, our predictor greatly improves the encoding of the sequence. Figure 15 clearly shows the amount of data necessary to be transmitted is lower with TD than with BM (see "Whole image" lines for BM, in black, versus TD, in red). On the contrary Figure 16 shows, for `Coastguard` sequence, our predictor fails to improve prediction for the high quality values. For `Garden` sequence, the amount of data necessary to transmit is approximately the same for both approaches. Note that for very low qualities the major information are much more carried by motion vectors and tangent distance parameters than by the prediction error. Finally we remark the amount of data needed to store tangent distance's parameters plus the prediction error is systematically lower (except for `Coastguard` sequence) than the amount of data needed for BM.

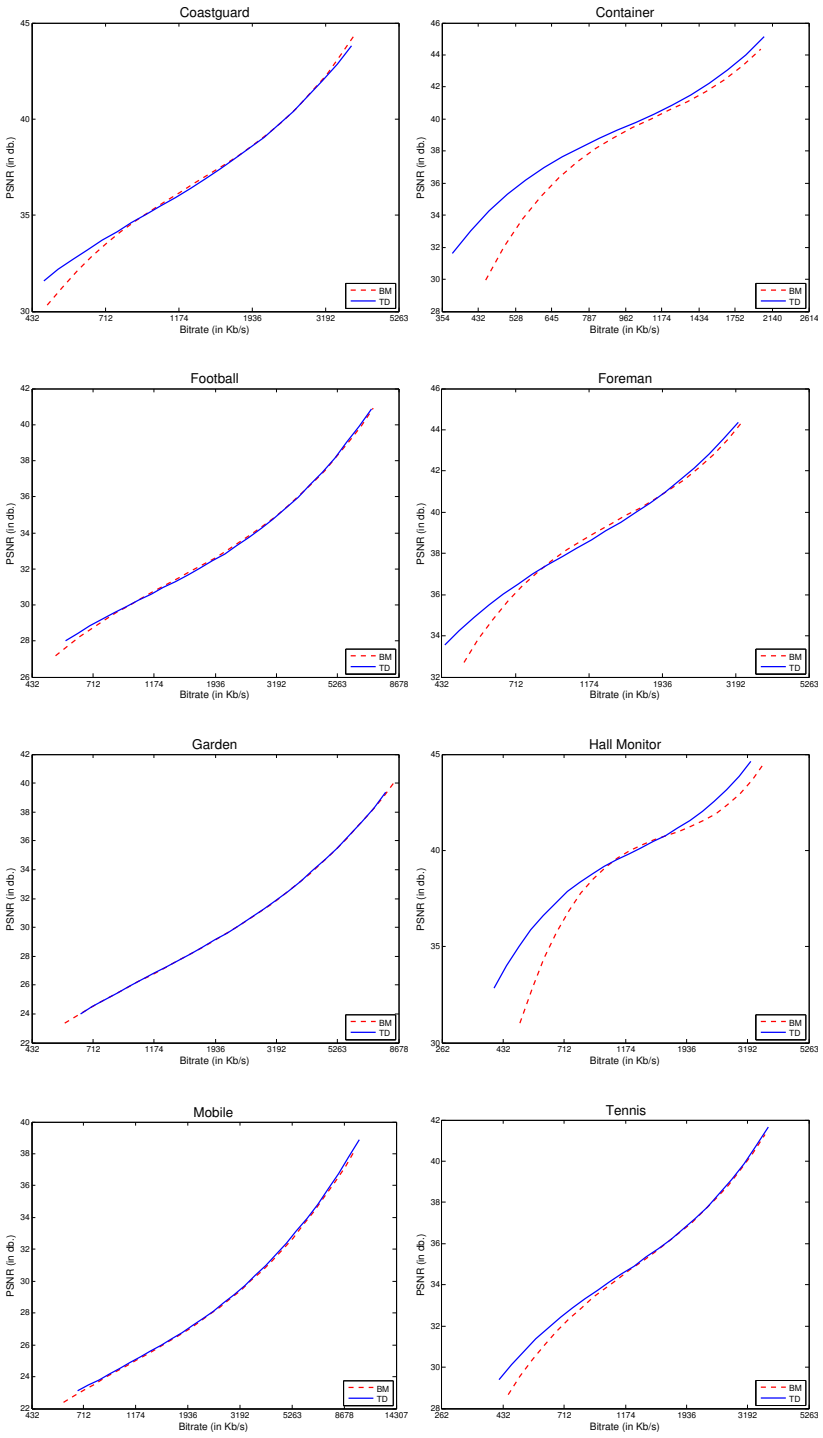


Figure 14: Rate-distortion curves for BM (red dashed lines) and TD (blue plain lines) algorithms for eight standard video sequences.

The average number of blocks per image predicted by TD with $\theta \neq 0$ is less than 3% over the eight tested sequences. This percentage depends on the tested sequence but also on the quality. It does not exceed 10% and decreases for medium qualities. However, on some frames, it can achieve 30 to 40% and even 50%. The interpretation is rather simple: most of time, BM prediction gives an acceptable result and DT prediction is only used if the prediction becomes difficult. This confirms our interpretation of result obtained on `Garden` sequence in Section 3.4.3. Obviously, a better score function and an adaptation of modeled transformations to the image motion would probably increase these percentages and the compression gain.

3.4.5. Conclusion about our codec implementation

Tests of Sections 3.4.3 and 3.4.4 show that our approach outperforms BM both in compression gain and amount of data to transmit. In particular, we note that TD highly increases compression gains in the lowest quality interval (i.e. $[0 - 20]$) while significantly reducing the amount of data necessary to be transmitted to the decoder (see for example results for `Container` and `Hall Monitor` sequences).

Even if this bit rate is lower than the one evaluated theoretically (compared to the simple framework of the previous section, all advanced treatments of the codec shrink the difference between the size of the error prediction generated by our method and the one generated by BM), the proposed method efficiently enhances the Theora codec in real conditions. With such a coarse and naive approach, these results are very encouraging: for all the sequences we have tested, we used the same set of parameters. As we said before, the scope of this paper is not to find an optimal set of parameters but only to prove this algorithm is efficient and works well. Moreover tuning our algorithm, for each tested sequence, until finding the perfect set of parameters would not have permitted to honestly compare both methods.

However this is clear that these results could be greatly improved by considering the following points.

- For simplicity we model the same transformations in tangent distance for all the sequences. A better selection (for instance depending on the context) should improve results by adapting them to the situations that occur (type of motions, *etc*).
- θ values are naively recorded: this encoding could be improved.
- The score function could take into account more parameters (number of successive zeros, amplitude of motion vector, *etc*). Coefficients of this score function should be adapted according to the expected quality.

Experiments presented in this subsection confirm that our approach significantly improves the compression rate.

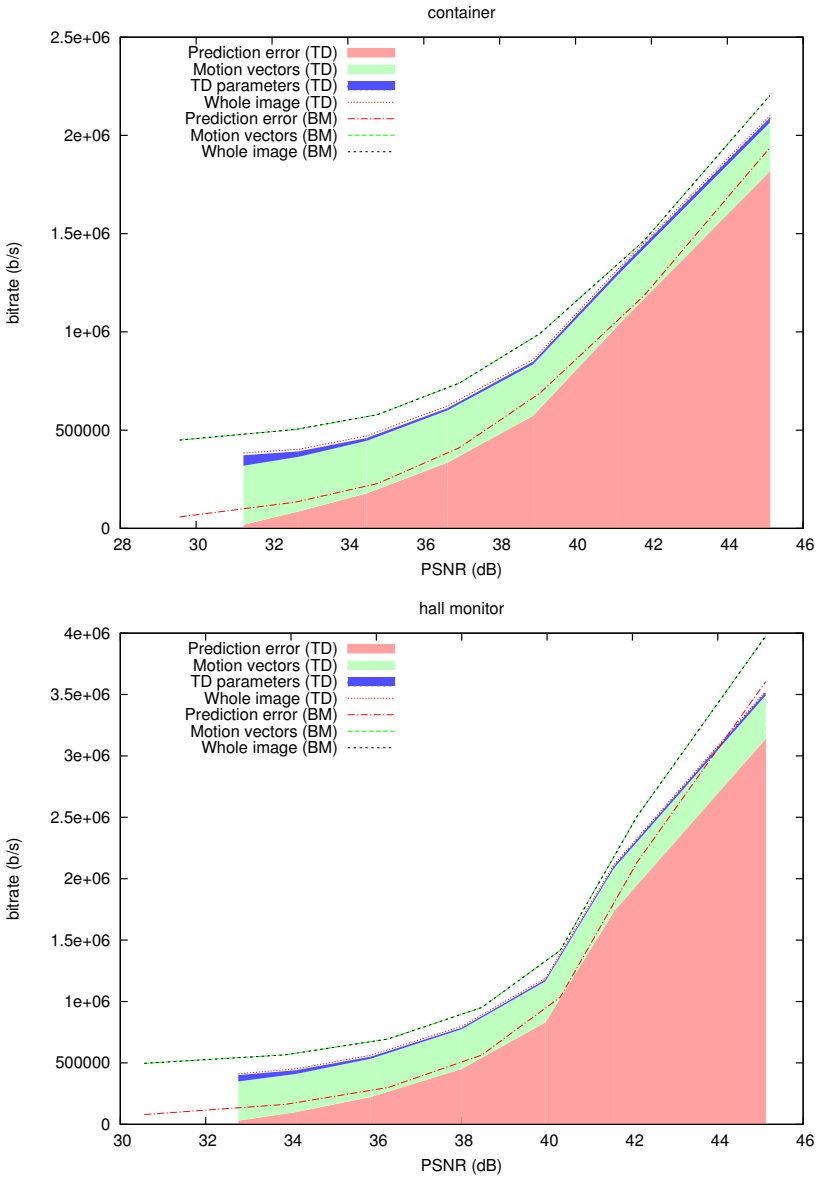


Figure 15: Distribution of bit rate to transmit for `Container` and `Hall Monitor` sequences and comparison between TD and BM. The figure gives, for a given PSNR value, the repartition of encoded data between prediction error, motion vectors and TD parameters.

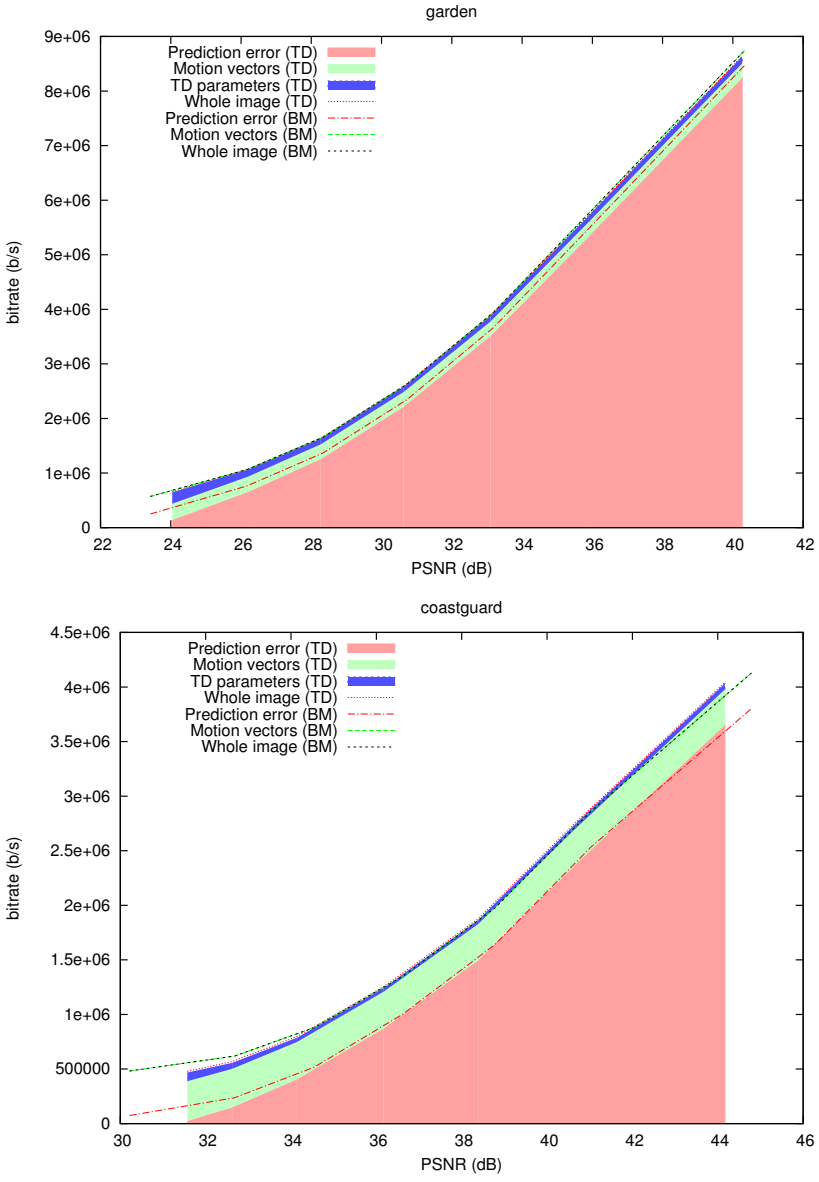


Figure 16: Distribution of bit rate to transmit for `Garden` and `Coastguard` sequences and comparison between TD and BM.

4. Conclusion

We have presented a new motion compensation algorithm based on the use of tangent distance. Unlike many Block-Matching methods, the proposed method not only handles the evolution of positions of blocks, but also the evolution of pixels inside these blocks. The method is simple and very robust. We prove its robustness using many tests (transparent objects, deformable objects, rotations, translations, zoom, ...). Comparisons with a Block-Matching algorithm show that our algorithm systematically improves the quality of the prediction. The quality of the result depends on the transformation introduced into the tangent distance model. Additionally to classic affine transformations we successfully introduced a simple and efficient illumination transformation.

We have also implemented our method into the Theora codec. This was the opportunity to discuss a criteria to match blocks during the motion compensation step. This criteria not only takes into account the similarity into the spatial domain, but also into the frequency domain. This gives a more robust score function used to measure the quality of the prediction of a block. Tests performed on several standard video sequences prove the capability of our method included into a codec in terms of compression rates.

An interesting work would be the possibility to choose the transformations we want to model during the encoding step. This would permit to adapt transformations to blocks and then to get optimal results. The scoring function could also be improved by taking into account successive zeros and the norm of the translation vector. Finally, this would be useful to define the best way to record additional parameters necessary to the decoding of our prediction. With a naive entropy coding and precomputed Huffman table, we already get good results. We then think that with a better encoding scheme, our method could efficiently improve common codecs.

References

[1] Alshin, A., Alshina, E., Lee, T., 2010. Bi-directional optical flow for improving motion compensation. In: Picture Coding Symposium. pp. 422–425.

[2] Altunbasak, Y., Tekalp, A., 1997. Closed-form connectivity-preserving solutions for motion compensation using 2-D meshes. Transactions on Image Processing 6 (9), 1255–1269.

[3] Bergen, J., Anandan, P., Hanna, K., Hingorani, R., 1992. Hierarchical model-based motion estimation. European Conference on Computer Vision, 237–252.

[4] Božinović, N., Konrad, J., May 2004. Mesh-based motion models for wavelet video coding. In: International Conference on Acoustics, Speech, and Signal Processing. Vol. III. pp. 141–144.

[5] Chan, E., Panchanathan, S., 1993. Review of block matching based motion estimation algorithms for video compression. In: Canadian Conference on Electrical and Computer Engineering. pp. 151–154.

[6] Dai, C., Escoda, O. D., Yin, P., Li, X., Gomila, C., 2007. Geometry-adaptive block partitioning for intra prediction in image and video coding. In: International Conference on Acoustics, Speech, and Signal Processing. pp. 657–660.

[7] Fabrizio, J., Dubuisson, S., 2007. Motion estimation using tangent distance. In: International Conference on Image Processing. pp. 489–492.

[8] Foundation, X., Mar. 2011. Theora specification. `http://www.theora.org/doc/Theora.pdf`.

[9] Furht, B., Furht, B., Greenberg, J., 1996. Motion estimation algorithms for video compression. Kluwer Academic Publishers.

[10] G., B., 2001. Calculation of average PSNR differences between RD-curves. In: Proceedings of ITU-T Q.6/SG16 VCEG 13th Meeting. pp. 934–939.

[11] Gao, X. Q., Duanmu, C. J., Zou, C. R., 2000. A multilevel successive elimination algorithm for block matching motion estimation. Transactions on Image Processing 9 (3), 501–504.

[12] Guo, L., Yin, P., Zheng, Y., Lu, X., Xu, Q., Solé, J., 2010. Simplified geometry-adaptive block partitioning for video coding. In: International Conference on Image Processing. pp. 965–968.

[13] Gyaourova, A., Kamath, C., Cheung, S.-C., Nov. 2003. Block matching for object tracking. Tech. Rep. UCRL-TR-201054, Lawrence Livermore National Laboratory.

[14] Horn, B., Schunk, B., 1981. Determining optical flow. Artificial Intelligence 17, 185–203.

[15] Hsu, P., Liu, K. J. R., Chen, T., 2001. A low bit-rate video codec based on two-dimensional mesh motion compensation with adaptive interpolation. Transactions on Circuits and Systems for Video Technology 11 (1), 111–117.

[16] Jialu, Z., Yongdong, Z., Yanfei, S., Guangnan, N., 2007. Panoramic video coding using affine motion compensated prediction. In: Multimedia Content Analysis and Mining. pp. 112–121.

[17] Kamikura, K., Watanabe, H., Jozawa, H., Kotera, H., Ichinose, S., dec 1998. Global brightness-variation compensation for video coding. Transactions on Circuits and Systems for Video Technology 8 (8), 988–1000.

[18] Kubasov, D., Guillemot, C., Oct. 2006. Mesh-based motion-compensated interpolation for side information extraction in distributed video coding. In: Proceedings of the IEEE Conference on Image Processing. Atlanta, USA, pp. 261–264.

[19] Lopes Texeira, L., Alves, A., 1996. Block matching algorithms in MPEG video coding. In: International Conference on Pattern Recognition. pp. 934–939.

[20] Love, N., Kamath, C., Apr. 2006. An empirical study of block matching techniques for detection of moving objects. Tech. Rep. UCRL-TR-218038, Lawrence Livermore National Laboratory.

[21] Lucas, B., Kanade, T., 1981. An iterative image registration technique with an application to stereo vision. In: Proc. of 7th International Join Conference on Artificial Intelligence. Vancouver, Canada, pp. 674–679.

[22] Macherey, W., Keysers, D., Dahmen, J., Ney, H., 2001. Improving automatic speech recognition using tangent distance. In: European Conference on Speech Communication and Technology. Vol. III. Aalborg, Denmark, pp. 1825–1828.

[23] Mariani, R., 2002. A face location and recognition system based on tangent distance. In: Multimodal interface for human-machine communication. World Scientific Publishing Co., Inc., River Edge, NJ, USA, pp. 3–31.

[24] Martucci, S., 1990. Reversible compression of HDTV images using median adaptive prediction and arithmetic coding. In: International Symposium on Circuits and Systems. pp. 1310–1313.

[25] Melanson, M., 2004. VP3 bitstream format and decoding process.

[26] Molloy, D., Whelan, P., 2000. Active-mesh. Pattern Recognition Letters 21 (12), 1071–1080.

[27] Nie, Y., Ma, K.-K., 2002. Adaptive rood pattern search for fast block-matching motion estimation. Transactions on Image Processing 11 (12), 1442–1448.

[28] Nosratinia, A., 2001. New kernels for fast mesh-based motion estimation. Transactions on Circuits and Systems for Video Technology 11 (1), 40–51.

[29] Odobez, J.-M., Bouthemy, P., 1995. Robust multiresolution estimation of parametric motion models. International Journal of Visual Communication and Image Representation 6 (4), 348–365.

[30] Richardson, I. E. G., 2003. H.264 / MPEG-4 Part 10 : Intra Prediction. `http://www.vcodex.com/files/h264_intrapred.pdf`.

[31] Rodrigues, N., da Silva, V., de Faria, S., 2001. Hierarchical motion estimation compensation with spatial and luminance transformations. In: International Conference on Image Processing. pp. 518–521.

[32] Schwenk, H., Milgram, M., 1996. Constraint tangent distance for on-line character recognition. In: International Conference on Pattern Recognition. pp. 515–519.

[33] Simard, P., LeCun, Y., Denker, J., Victorri, B., 1998. Transformation invariance in pattern recognition, tangent distance and tangent propagation. In: Neural Networks: Tricks of the trade. pp. 549–550.

[34] Simard, P., Victorri, B., LeCun, Y., Denker, J., 1992. Tangent prop: a formalism for specifying selected invariances in adaptive networks. In: Advances in Neural Information Processing Systems. Denver, CO, pp. 895–903.

[35] Wang, Y., Lee, O., 1994. Active mesh—a feature seeking and tracking image sequence representation scheme. Transactions on Image Processing 3, 610–624.

[36] Wei, J., Li, Z.-N., 1997. Motion compensation in color video with illumination variations. In: International Conference on Image Processing. pp. 614–617.

[37] Weinberger, M., Seroussi, G., Sapiro, G., 1996. LOCO-I: a low complexity, context-based, lossless image compression algorithm. In: Proc. Data Compression Conference. pp. 140–149.

[38] Wiegand, T., Sullivan, G., Reichel, J., Schwarz, H., Wien, M., 2007. Joint Draft ITU-T Rec. H.264 — ISO/IEC 14496-10 / Amd.3 Scalable video coding.

[39] Xu, J.-B., Po, L.-M., Cheung, C.-K., 1999. Adaptive motion tracking block matching algorithms for video coding. Transactions on Circuits and Systems for Video Technology 9 (7), 1025–1029.

[40] Zhu, S., Ma, K.-K., 2000. A new diamond search algorithm for fast block-matching motion estimation. Transactions on Image Processing 9 (2), 287–290.