# Inter Prediction Methods based on Linear Embedding for Video Compression

Martin Alain[a,b], Christine Guillemot[b], Dominique Thoreau[a], Philippe Guillotel[a]

[a] *Technicolor, 975 Avenue des Champs Blancs, 35576 Cesson-Sevigne, France*
[b] *INRIA, 263 Avenue du General Leclerc, 35042, France*

**Abstract**

This paper considers the problem of temporal prediction for inter-frame coding of video sequences using locally linear embedding (LLE). LLE-based prediction, first considered for intra-frame prediction, computes the predictor as a linear combination of $K$ nearest neighbors ($K$-NN) searched within one or several reference frames. The paper explores different $K$-NN search strategies in the context of temporal prediction, leading to several temporal predictor variants. The proposed methods are tested as extra inter-frame prediction modes in an H.264 codec, but the proposed concepts are still valid in HEVC. The results show significant Rate-Distortion performance gains are obtained with respect to H.264 (up to $15.31$ % bit-rate saving).

*Keywords:* locally linear embedding, temporal prediction, video compression.

## 1. Introduction

Most video coding standards achieve data compression by exploiting similarities within frames (i.e., the spatial redundancy), as well as between the target frame and one or several reference frames (i.e., the temporal redundancy). Intra-frame coding techniques are used to reduce the spatial redundancy within each frame separately, whereas inter-frame coding techniques are used to reduce the temporal redundancy between successive frames of a video sequence.

Intra prediction is a powerful tool to remove spatial redundancy in intra-frame coding. In the H.264 video compression standard [1], each frame is partitioned into blocks, and for each block to be coded, a predictor block is created by extrapolating previously coded and reconstructed pixels surrounding the target block to be coded. Nine prediction modes have been defined which propagate surrounding pixels along different directions. In HEVC [2], the intra-frame prediction has been extended to support 33 directional prediction modes. The encoder selects the prediction mode which is the best in a rate-distortion (RD) sense, using a Lagrangian optimization technique, and signals the retained mode to the decoder.

Inter-frame predictors are typically obtained by motion estimation and compensation methods that match every block to be coded with a similar block in one or several reference frames, using the so-called block matching (BM) algorithm [1][2]. The position of the best matching block in a reference frame is signaled to the decoder by transmitting a motion vector. The motion vector may locate the best matching block with a fractional pixel (pel) accuracy thanks to fractional positions interpolation in the reference frames.

The motion estimation can also be performed using the so-called template matching (TM) technique [3]. The methods exploits the correlation between the current block and a pre-defined set of neighboring pixels, called the template of the current block. Rather than looking for the most correlated block in the reference frames, one looks for the most correlated template. The block which is adjacent to this template is used as a predictor for the current block. The motion compensation is performed using the exact same process, so no motion information needs to be transmitted to the decoder. This technique efficiency has also been demonstrated for intra-frame prediction [4]. The RD performance of this method can be improved by using a weighted combination of multiple predictors. Initially a simple averaging of the predictors was performed [5][6], but methods using adaptive weights, e.g. using sparse approximation [7], were shown to bring significant improvements.

In this paper, we consider an approximation method called Locally Linear Embedding (LLE), introduced in [8] for data dimensionality reduction, which we adapt to the problem of temporal prediction. The LLE technique has already been shown to be very efficient for intra-frame prediction in [9][10]. However, the derivation from intra-frame to

inter-frame prediction is not trivial, mainly because the proposed techniques are now in competition with the motion estimation/compensation, which is a more efficient prediction tool than the intra-frame directional modes. The idea is to first search for a representation of the template as a linear combination of $K$ templates (called $K$-NN templates) taken from a search window denoted SW. The linear combination coefficients (or weights) are then applied on the blocks adjacent to the $K$-NN templates to yield the current block predictor. The LLE weights are computed using a least square formulation of the template approximation problem under the constraint that they sum to one.

The $K$-NN search strategy has a strong impact on the predictor quality. In fact, the TM technique efficiency rely on the hypothesis that the template and its adjacent block are well correlated. First, we proposed a direct derivation of the TM technique, where the $K$-NN can be found by computing distances between the template of the current block and those of candidate blocks in the reference frames. This method is denoted Template Matching LLE (TM-LLE) and, as for the TM method, no side information (i.e., no motion vector) needs to be sent to the decoder. A variant of this method is introduced where the first neighbor is searched by template matching (as in TM-LLE), but the remaining $(K-1)$-NN are found by computing a distance between the complete patch formed by the template and adjacent block of the first neighbor and the candidate patches in the search window. The method is denoted Improved Template Matching LLE (ITM-LLE).

Second, to further improve the $K$-NN search, we introduce a method enforcing the correlation between the templates and their adjacent blocks, but requiring the transmission of side information to the decoder. Thus, we propose a method where the $K$-NN search is initialized with a block-matching algorithm. This implies that a motion vector is sent to the decoder. We then find the remaining $(K-1)$-NN as in ITM-LLE. This method is named Block-Matching LLE (BM-LLE).

Finally, we propose an improved variant of the ITM-LLE method, denoted optimized ITM-LLE (oITM-LLE). In this method, we basically obtain $L$ predictors by running $L$ times the ITM-LLE method. The best iteration in a RD sense is retained, and its index is sent to the decoder.

The experiments and their analysis focus on RD performance evaluations of the proposed prediction methods against the standard reference techniques: directional and motion estimated/compensated prediction modes of H.264 and template matching averaging (TM-A). This analysis is carried out using a legacy H.264 implementation, but note that the proposed techniques are still applicable in HEVC, since the inter-frame prediction tool in HEVC follow the same principles as those used in H.264. Simulation results show that significant RD performance improvements are achieved compared to the reference prediction methods. The performed analysis includes elements of complexity in terms of execution times measured at the encoder.

The rest of the paper is organized as follows. Section 2 reviews background on video compression methods (H.264, HEVC), as well as the TM-based prediction methods. Section 3 describes the proposed LLE-based temporal prediction techniques. Section 4 explains how the proposed prediction methods have been used in an H.264 codec and they could be used in HEVC. We then give the PSNR-rate performance gains compared to the reference H.264 codec. Section 5 sums up how the proposed techniques can be integrated in the HEVC codec and why they are still valid.

## 2. Temporal prediction: Background

This section first summarizes the relevant features of the temporal prediction methods in the H.264 video compression scheme [1] and the corresponding techniques in HEVC [2]. It then briefly revises state-of-the-art temporal prediction methods based on template matching.

### *2.1. Motion-compensated inter-frame prediction in H.264 and HEVC*

A motion vector (MV) is estimated for each block of the possible partitions ($4\times4$ to $16\times16$, $16\times8$ in H.264, $8\times8$ to $64\times64$ in HEVC), usually using a BM algorithm. The MV establishes a correspondence between the current block and a block in one of the reference frames. This block is used as the predictor. The reference frames are stored in two buffers called $L_0$ and $L_1$ lists. The $L_0$ list contains reference frames from the past while the $L_1$ list contains reference frames from the future. For each block, the motion vector, the reference frame index and the list index are coded and sent to the decoder. The coding efficiency of the MVs relies on their predictive coding, under the assumption that the motion vector field is continuous (at least locally). Thus, in H.264, a motion vector predictor (MVP) is computed as the median of available neighboring MVs. Only the difference between the current MV and the MVP is then coded. In

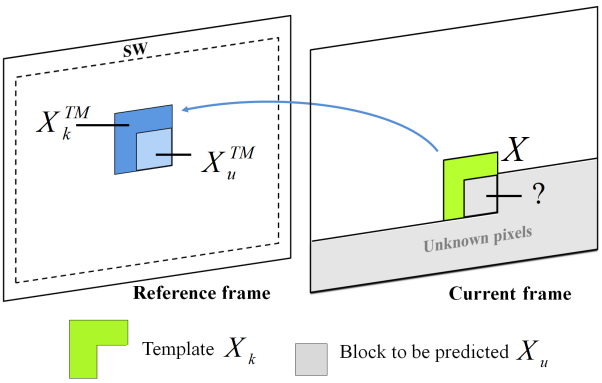


Figure 1: Template Matching for inter-frame prediction.

HEVC, the MVs coding efficiency is even improved by using the Adaptive Motion Vector Prediction list or the Merge list. The improved efficiency compared to H.264 for inter-frame prediction thus comes more from the optimization of the side information coding than from the prediction quality. In both H.264 and HEVC, the side information coding is further improved by using Context-Adaptive Binary Arithmetic Coding (CABAC).

### *2.2. Prediction based on Template Matching (TM)*

A Template Matching (TM) algorithm has been considered instead of the BM algorithm for both intra-frame [4] and inter-frame [3] prediction. The TM method is very close to the BM method, although this time it is not the pixels in the current block to be predicted which are used to find the best match but the template pixels, on the top and to the left of the blocks (see Fig. 1). The union of the template $X_k$ and of its adjacent block $X_u$ forms the patch $X$. The underlying basic idea of the algorithm is to take advantage of a supposed correlation between the pixels in the block and those in its template. For inter-frame prediction, the first step of the algorithm is to look for the NN of the template in a search window defined in one or more reference frames. Here and for the rest of this paper the metric used to find the NN is the sum of absolute difference (SAD). Once the NN $X_k^{\mathrm{TM}}$ of $X_k$ is found, the adjacent block $X_u^{\mathrm{TM}}$ of this template is used as a predictor for the current block $X_u$. The benefit of this method is that this prediction process can be reproduced at the decoder, hence no side information (such as MV) needs to be sent to the decoder anymore [11]. Although efficient in terms of bit-rate reduction in the case of homogeneous texture, the method yields low quality predictors in image areas where the block and its template are not well correlated.

The technique has been extensively studied in H.264 [3][4][11], and was considered for intra-frame prediction in the early stage of HEVC [12]. Even though it was not retained for the standard, it was later shown that it can improve the RD performance of HEVC, e.g. when used for inter-frame bi-prediction in combination with block matching [13].

In order to improve the predictor quality, a so-called template matching averaging (TM-A) method has been proposed in [6]. In this method, one looks for the $K$-NN of the current template and not only the first one. The predictor of the current block $X_u$ is then obtained by averaging the $K$ blocks adjacent to the $K$-NN of the template. This enables to smooth the predictor, which is advantageous most of the time, and computationally reasonable. Given the higher performances of TM-A compared to TM, only TM-A has been considered in the comparative assessment made in section 4.

Note that different approaches relying on $K$-NN combination have been explored for intra-frame or inter-frame coding, such as sparse representations [7][14], or Nonnegative Matrix Factorization [9][10]. However, results in [7] show that neighbor embedding techniques such as LLE or NMF outperform sparse representations in terms of RD performances. In [10], results show that NMF performs slightly better than LLE in terms of RD performance, but complexity for NMF is much higher than LLE, especially at the encoder side. These conclusions motivate the use of the LLE in this paper, described in the following section. More recently, prediction methods based on weighted template matching have been proposed to improve HEVC Intra RD performances, e.g. in [15]. The method presented in [15] demonstrates the efficiency of weighted template matching against HEVC Intra mode. However, this method is optimized to reduce complexity, e.g. by using tabulated exponential weights, while we focus on optimizing the RD performances by using optimal weights in a least square sense.

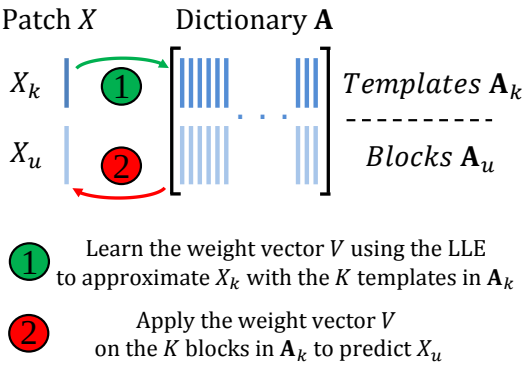


Figure 2: Predictor computation technique based on LLE. Assuming the dictionary $\mathbf{A}$ is available, a weighting coefficient vector $V$ is learned using the LLE, which approximates the current template $X_k$ with the $K$ templates in $\mathbf{A}_k$. The current block predictor $\hat{X}_u$ is obtained by applying the learned weights to the $K$ blocks in $\mathbf{A}_u$.

## 3. Inter-frame prediction based on LLE

This section describes the proposed inter-frame prediction techniques using LLE. LLE-based prediction methods search for the linear combination of $K$ nearest neighbors which best approximates in a least squares sense the template of the current block, under the constraint that the weights of the linear combination sum to one. The block predictor is computed by applying the found weights to the blocks which are adjacent to the $K$-NN templates. The section below first presents the weights computation, assuming that the $K$-NN are available. We then describe the $K$-NN search strategies which are studied in the paper.

### *3.1. LLE-based predictor computation*

The weighting coefficients are computed by formulating the template approximation problem as a least squares problem, under the constraint that the weights sum to one. The found weighting coefficients are applied in the linear combination of the adjacent block pixels in order to compute the block predictor (see Fig. 2).

Let $\mathbf{A} = [\frac{\mathbf{A}_k}{\mathbf{A}_u}]$ denote a so-called dictionary represented by a matrix of dimension $N \times K$. The columns of the dictionary $\mathbf{A}$ are constructed by stacking the $K$ candidate texture patches found after the $K$-NN search step. The sub-matrices $\mathbf{A}_k$ and $\mathbf{A}_u$ contain the pixel values of the templates and of the blocks respectively. Let $X = [\frac{X_k}{X_u}]$ be the vector composed of the known pixels of the template $X_k$ and the unknown pixels of the current block $X_u$.

The LLE-based prediction problem can be re-written as:

$$\min_{V} \|X_k - \mathbf{A}_k V\|_2^2 \text{ s.t. } \sum_m V_m = 1 \tag{1}$$

where $V$ denotes the optimal weighting coefficients vector which is computed as

$$V = \frac{\mathbf{D}^{-1}\mathbf{1}}{\mathbf{1}^{\mathrm{T}}\mathbf{D}^{-1}\mathbf{1}}. \tag{2}$$

The term $\mathbf{D}$ denotes the local covariance matrix (*i.e.*, in reference to $X_k$) of the selected $K$-NN templates stacked in $\mathbf{A}_k$, and $\mathbf{1}$ is the column vector of ones. In practice, instead of an explicit inversion of the matrix $\mathbf{D}$, the linear system of equations $\mathbf{D}V = \mathbf{1}$ is solved, then the weights are rescaled so that they sum to one.

The predictor of the current block $\hat{X}_u$ is then obtained as:

$$\hat{X}_u = \mathbf{A}_u V \tag{3}$$

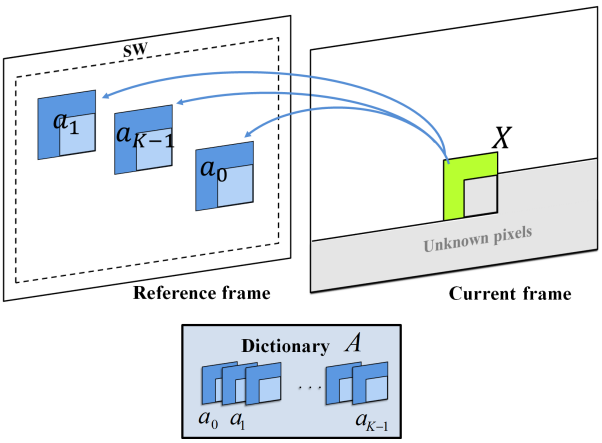


Figure 3: Search for the $K$-NN of the current template in the TM-LLE method.

### *3.2. Template-based K-NN search for LLE-based prediction*

This section first describes two LLE-based methods in which the $K$-NN search is not based on real motion estimation but rather on template matching, and thus does not need to send side information to the decoder.

The first $K$-NN search method simply looks for the $K$-NN of the current template (see Fig. 3). The $K$ patches stacked in the dictionary $\mathbf{A}$ thus consist in the $K$ templates found ($\mathbf{A}_k$), along with their adjacent blocks ($\mathbf{A}_u$). The current block can then be predicted as explained in section 3.1. This $K$-NN strategy is as simple as the one of the TM-A method, however it suffers from the same limitations in the sense that the candidate patches found may not lead to the best predictor. This method, referred to as Template Matching LLE (TM-LLE), does not require transmitting extra side information to the decoder which can perform the same $K$-NN search.

An improved variant of the previous method is to find the first NN of the current template in a first step. Then in a second step, the $(K-1)$-NN of the previous patch are found in order to build the complete dictionary $\mathbf{A}$. Note that these $(K-1)$-NN are now found with respect to the whole patch, which tends to reinforce the correlation between the templates and the blocks (inner correlation of the patch). However, if the first NN found is not well correlated with the current patch, the predictor quality will decrease. This method is referred to as Improved Template Matching LLE (ITM-LLE).

### *3.3. Block-based K-NN search for LLE-based prediction*

To better overcome the potential lack of correlation between the blocks and their templates, we propose to use the current block to guide the $K$-NN search. This implies that additional information needs to be sent to the decoder, so that it can find the exact same $K$-NN. The following methods use the current block to find the first NN. The $K-1$ remaining patch are found with respect to the previous NN. To signal the NN to the decoder, a motion vector is transmitted.

The prediction first proceeds by computing the motion vector using a classical block matching algorithm. The best matching block (the first nearest neighbor) $X_u^{\mathrm{BM}}$ of the current block $X_u$ is found. The patch containing this block $X^{\mathrm{BM}} = [\frac{X_k}{X_u^{\mathrm{BM}}}]$ is used as the first patch of the dictionary for the LLE. The second step is to search for the $(K-1)$-NN $[\mathbf{a}_1, ..., \mathbf{a}_{K-1}]$ of the patch $X^{\mathrm{BM}}$. The union of these $K$ patches $\mathbf{A} = \left[X^{\mathrm{BM}}, \mathbf{a}_1, ..., \mathbf{a}_{K-1}\right]$ forms the dictionary (see Fig. 4) used for the LLE computation described above by equations (1), (2) and (3). The MVs can then be encoded as in the reference codec (e.g. H.264 or HEVC), which does not increase the MVs coding cost.

### *3.4. Optimized template-based K-NN search for LLE-based prediction*

To further enhance the RD performance, the ITM-LLE method was optimized. First, $L$ nearest neighbors to the template of the current block are found. For each patch $c_l$ found by this $L$-NN search, a dictionary is constructed leading to a set of $L$ dictionaries $\mathbf{A}^0, ..., \mathbf{A}^{L-1}$. Each dictionary $\mathbf{A}^l$ is formed by stacking the patch formed by $c_l$ and the adjacent block and its $(K-1)-NN$ (see Fig . 5). Since the patches are found using only the template of the

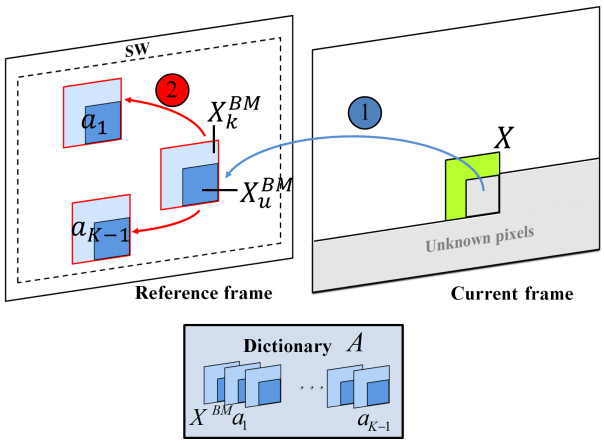


Figure 4: Illustration of the first two steps of the BM-LLE method.

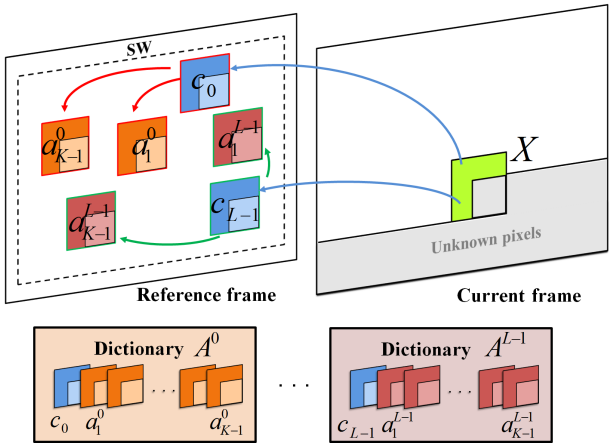


Figure 5: Example of construction of $A^0$ and $A^{L-1}$ dictionaries from candidate patches $c_0$ and $c_{L-1}$ (oITM-LLE method).

current block, the same set of dictionaries can be found by the decoder. An LLE-based predictor is computed with each dictionary and the dictionary $\mathbf{A}^{l_{opt}}$ giving the best RD performances is retained, and its index $l_{opt}$ is signalled to the decoder. The number of dictionaries $L$ is taken as a power of 2, and the index is coded with a fixed length binary code. This method is referred to as optimized ITM-LLE (oITM-LLE). The different steps of the algorithm are detailed as pseudo-code in Algorithm 1.

## 4. Simulations and results

### *4.1. Integration in the MPEG-4 AVC/H.264 coding scheme*

The reference and proposed methods presented in sections 2 and 3 respectively have been tested in a H.264 framework. The TM-A and the LLE-based prediction methods have been introduced in the coding scheme in competition with the existing H.264 inter and intra prediction modes. This solution was chosen over a simple replacement of the existing motion compensation prediction (MCP) method by an Inter LLE-based method because they are complementary. In fact the MCP method is already quite efficient, e.g. for smooth textures with little or no motion. A typical example is a static background where the Skip mode can be used. Multi-patches methods such as the proposed ones using LLE are better at predicting high frequency pseudo-periodic textures, which are not easy to reconstruct using only one block.

**Algorithm 1** oITM-LLE prediction method

**Input:** $X, K, L$
**Output:** current block predictor $\hat{X}_u$
Determine the $L$ nearest neighbors of the current template $X_k$, *i.e* the templates $\mathbf{c}_{k_0}, ..., \mathbf{c}_{k_{L-1}}$ such that $d_0 \leq ... \leq d_{L-1}$ with $d_i = ||X_k - \mathbf{c}_{k_i}||$
Retain the $L$ patches associated with the $L$ templates determined in the previous step: $\mathbf{c}_i = [\frac{\mathbf{c}_{k_i}}{\mathbf{c}_{u_i}}], i = 0, \ldots, L-1$
**for** $l = 0 \rightarrow l = L-1$ **do**
  Find $K-1$ neighbors close to $\mathbf{c}_l$, *i.e.*, the patches $\left[\mathbf{a}_1^l, ..., \mathbf{a}_{K-1}^l\right]$
  Set $\mathbf{A}^l = \left[\mathbf{c}_l, \mathbf{a}_1^l, ..., \mathbf{a}_{K-1}^l\right]$
  Retrieve $\mathbf{A}_k^l = \left[\mathbf{a}_{k_0}^l, ..., \mathbf{a}_{k_{K-1}}^l\right]$ and $\mathbf{A}_u^l = \left[\mathbf{a}_{u_0}^l, ..., \mathbf{a}_{u_{K-1}}^l\right]$ from $\mathbf{A}^l$. Note that here $\mathbf{c}_l = [\frac{\mathbf{a}_{k_0}^l}{\mathbf{a}_{u_0}^l}]$
  Solve the constrained least squares problem:
  $\min_{V^l} \left\|X_k - \mathbf{A}_k^l V^l\right\|_2^2$ s.t. $\sum_m \mathbf{V}_m^l = 1$
  Get the predictor:
  $\hat{X}_u^l = \mathbf{A}_u^l V^l$
  Compute the corresponding RD cost $RD^l$
**end for**
Select the optimum $l_{opt} = \arg\min_l RD^l$
Set $\hat{X}_u = \mathbf{A}_u^{l_{opt}} V^{l_{opt}}$

The MCP method is used for block partitions going from size $16 \times 16$ to $8 \times 8$. Experiments using the H.264 reference software show that further sub-partitioning into $8 \times 4$, $4 \times 8$ and $4 \times 4$ blocks brings little improvement in terms of coding performance, while increasing the complexity. In fact, the signaling cost of the MVs usually becomes prohibitive for such small partitions. The TM-A and our LLE-based prediction methods are applied on $8 \times 8$ blocks.

The Skip mode is allowed for both P and B frames, as well as the additional bi-predictive mode for the B frames. The Inter TM-A or LLE-based method to be tested is introduced in competition with the MCP method for $8 \times 8$ partitions only, and for both P and B frames. We set the template width to 3 pixels. The choice between the classical MCP method or the additional TM-A or LLE-based method is based on a RDO criterion. The syntax then needs to be modified to send a flag to the decoder indicating which method is selected. The rate is therefore increased by 1 bit for each partition that features an additional Inter prediction method (TM-A or LLE-based). The index $l_{opt}$ of the selected dictionary is also transmitted using a fixed length code when the oITM-LLE method is retained for predicting the current block. Experiments showed that the bit-rate corresponding to the flag for the selected method amounts in average to about two to three percents of the full bit-rate, while the bit-rate of the dictionary index reaches four to five percents. Note that this extra information bit-rate could be further reduced using the CABAC.

The search window for the Inter TM-A or Inter LLE-based prediction methods is defined in the reference frames of the $L_0$ list for the P frames, of the $L_0$ and $L_1$ lists for the B frames, and in the decoded part (causal part) of the current frame for both P and B frames. Note that if an LLE-based prediction method is applied to intra-frame prediction, the search window is only defined in the causal window. The search window is centered on the position of the current block.

Note that the solution we propose to integrate our method in the H.264 codec is still valid in HEVC. In fact, the prediction tools in HEVC follow the same principles as those used in H.264. As reported in section 2.2, it was shown that template matching based methods can outperform the intra-frame or inter-frame prediction tools of HEVC [13][15]. Our methods are optimized to further improve the prediction quality compare to these techniques, thus we expect to reach better RD performances. The use of the proposed methods in HEVC would require adapting our techniques to the different prediction units (PUs) size, especially the larger ones, which provide better RD performances [16]. This adaptation is conceptually straightforward, and can even be extended to rectangular PUs. Furthermore, additional tools providing RD gain, such as adaptive transform sizes, SAO [16], are directly compatible with the proposed methods. As mentioned above, the overhead in the bitstream corresponding to the flag indicating if the LLE-based method is used and the index for the oITM-LLE method is low for H.264, and we do not expect an increase for HEVC, since applying the proposed methods on larger PUs would reduce the number of flags or indexes to be transmitted.

Table 1: Simulations parameters

| **Parameter** | **Setting** |
|---|---|
| Sequence type: | IBBPBBP |
| Number of encoded frames: | 31 (Matrix: 34) |
| Number of reference frames: | |
| - for BM: | 4 |
| - for $K$-NN: | 4 (& causal part) |
| Search-range: | |
| - for BM: | 32 pel for CIF<br>64 pel for 720p |
| - for $K$-NN: | 32 pel for all |
| Search method: | |
| - for BM: | Fast Full Search |
| - for $K$-NN: | Full Search |
| Search accuracy: | |
| - for BM: | Quarter Pel |
| - for $K$-NN: | Full Pel |
| Block sizes for intra: | all 16x16 to 4x4 |
| Block sizes for MCP: | all 16x16 to 8x8 |
| Block sizes for LLE: | 8x8 |
| Template size: | +3 pel on width and height |
| Quantization parameter (I/P/B): | 22/23/24, 27/28/29,<br>32/33/34, 37/38/39 |
| Number $L$ of dictionaries for oITM-LLE | 32 |

### *4.2. Experimental conditions*

The proposed schemes have been implemented in the latest release of the JM-KTA software [17], in order to be compared with the H.264 prediction modes. Simulations have been run using 5 test sequences presenting different characteristics in terms of motion and texture (see Fig. 16-20 in section 7), and having different resolutions (three CIF sequences and two $1280 \times 720$ sequences). The CIF sequences are respectively made of 31 frames extracted from the Foreman sequence (frames 149-179), characterized by fast motion and smooth natural texture, 31 frames extracted from the "Rushes" sequence (frames 597-627) containing high frequency texture and complex motion, and 34 frames extracted from the trailer of the Matrix movie[1] (frames 1810-1843). This sequence was chosen because of its fast scene changes, occurring every 2-3 frames. The different scenes contain little motion but cover a wide range of textures. The $1280 \times 720$ sequences are made of 31 frames extracted from the "City" (frames 230-260) and the "Spincalendar" (frames 500-530) sequences respectively. The urban scene in the "City" sequence contains non-stochastic high frequency textures, with a slow camera motion. The "Spincalendar" sequence contains different textures following a smooth rotation motion. The modified encoder (see section 4.1) has been configured with the Main profile [1] and the parameters given in Table 1. The rate gains are obtained using the Bjontegaard measures [18]. The complexity is measured trough the percentage of the tested encoder processing time over the one of the H.264 reference encoder. Note that, due to different experimental conditions, the given complexity values correspond to averaged estimates with a standard deviation of about $10$ %.

In the following section, we first analyze the performances of the proposed methods when used for Inter prediction. The performances are analyzed as a function of the key parameters $K$, the reference frames number and the search window size. We also give the corresponding complexity. Second, the methods are also assessed when used for Intra prediction only or when used for both Intra and Inter prediction.

### *4.3. RD performance analysis & elements of complexity*

In this section we discuss the gains in terms of bit-rate reduction of the proposed methods against the H.264 reference, along with the execution times measured at the encoder. The values are averaged over the full panel of test sequences.

[1] ©1999-2015 Warner Bros.

*4.3.1. Impact of the parameter K*

Fig. 6 shows how the bit-rate savings for each method vary as a function of the parameter $K$. Fig. 7 shows the corresponding complexity for each method, except the oITM-LLE method, which reaches significantly higher levels (about 5000 to 6000 %). In terms of RD performance, the TM-LLE, ITM-LLE and oITM-LLE methods outperform the TM-A method, which demonstrates the better adaptation of the LLE weights compared to a simple averaging, especially when $K$ increases. In average, the highest bit-rate saving is achieved with the oITM-LLE method, reaching more than 8 %.

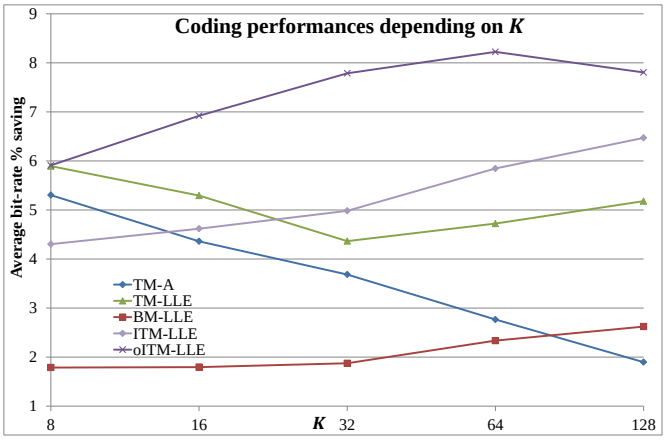


Figure 6: Coding performances obtained with the different proposed prediction methods as a function of the number $K$ of nearest neighbors.

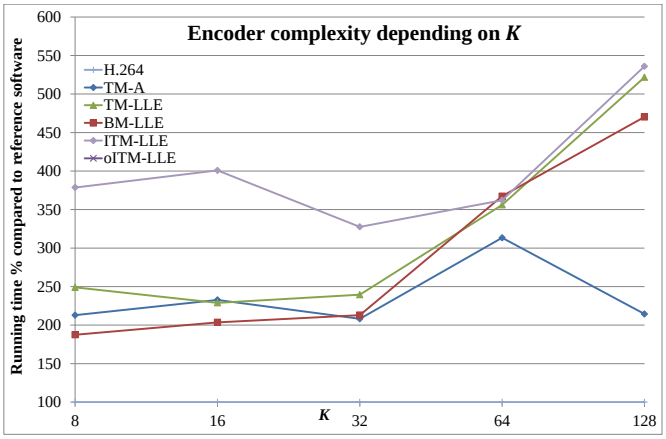


Figure 7: Encoder complexity obtained with the different proposed prediction methods as a function of the number $K$ of nearest neighbors.

In Table 2, we give, for each method, the $K$ value tuned to obtain a satisfying trade-off between the execution time and the coding gain (from Fig. 6 and Fig. 7), which are also given. We can see that the TM-LLE and ITM-LLE methods are competitive with the state-of-the-art methods, either in terms of complexity or RD performances. However, the BM-LLE is only competitive in terms of complexity, but not in terms of coding gains. The oITM-LLE does produce the best results in term of bit-rate reduction, but at a really high cost in terms of execution time. For the aforementioned reasons, the detailed analysis of the BM-LLE and oITM-LLE methods is not pushed further for inter-frame prediction. We will see however in section 4.4 that, when used for intra-frame prediction, the oITM-LLE method can be efficiently combined with less complex inter-frame prediction method such as TM-LLE. For the next

simulations, $K$ is set to the values presented in Table 2.

Table 2: $K$ values set to achieve a trade-off between RD performances and complexity.

| Method | $K$ value | Execution time in % | Bit-rate gain in % |
|---|---|---|---|
| TM-A | 8 | 213 | -5.30 |
| TM-LLE | 8 | 249 | -5.89 |
| BM-LLE | 16 | 204 | -1.80 |
| ITM-LLE | 64 | 362 | -5.85 |
| oITM-LLE | 32 | 5077 | -7.79 |

*4.3.2. Impact of the reference frames number*

Fig. 8 shows how the bit-rate gains for each method vary as a function of the number of reference frames. Fig. 9 shows the corresponding complexity. In terms of RD performance, we can see that the proposed methods can outperform the TM-A and the highest bit-rate reduction is achieved with the ITM-LLE method, reaching 6.33 %.

In Table 3, the reference frames number is set in order to achieve a satisfying trade-off between complexity and RD performances (from Fig. 8 and Fig. 9). The corresponding execution times and bit-rate reductions are given. We can see that the TM-LLE method can be competitive with the TM-A method in terms of RD performances while reducing the complexity. The ITM-LLE can achieve the highest bit-rate reduction, which requires an increased complexity.

Table 3: Reference frames number set to achieve a trade-off between RD performances and complexity.

| Method | Reference frames number | Execution time in % | Bit-rate gain in % |
|---|---|---|---|
| TM-A | 3 | 157 | -5.14 |
| TM-LLE | 3 | 156 | -5.83 |
| ITM-LLE | 1 | 276 | -6.33 |

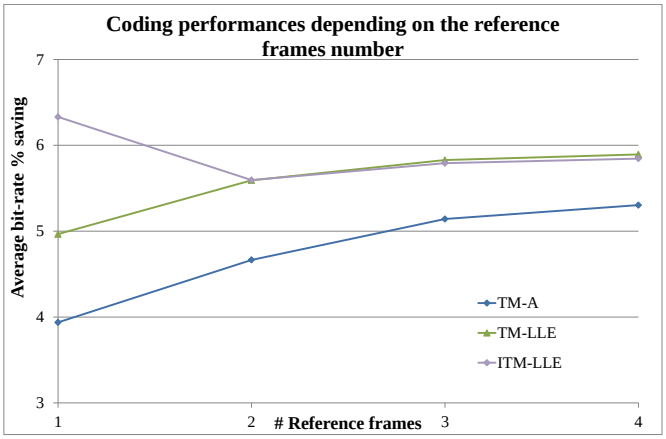


Figure 8: Coding performances obtained with the different proposed inter-frame prediction methods as a function of the number of reference frames.

*4.3.3. Impact of the search window size*

Fig. 10 shows how the bit-rate reductions for each method vary as a function of the search window range. Fig. 11 shows the corresponding complexity. Table 4 gives the coding gains and corresponding execution time when the search window range is set to achieve a satisfying trade-off between RD performances and complexity. The number

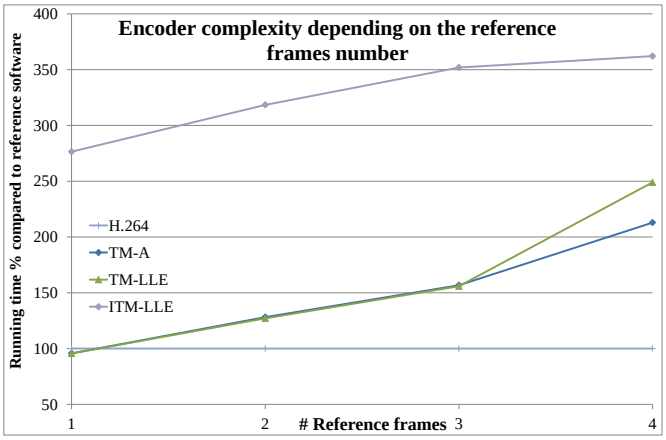


Figure 9: Encoder complexity obtained with the different proposed inter-frame prediction methods as a function of the number of reference frames.

of reference frames is not fixed, and set depending on this range. The search window range is first set to 16, with 4 reference frames, then set to 32, with 4 reference frames, and finally set to 64, with 1 reference frame. The last configuration was chosen with only 1 reference frame in order to limit the complexity. However, the same amount of patches is available for the $K$-NN search for the last two configurations.

We can see that the proposed methods can outperform the TM-A, reaching up to 7.20 % bit-rate savings for the ITM-LLE method. As for the previous results, the highest bit-rate reduction can be achieved by the ITM-LLE method, while the TM-LLE method allows a better trade-off between RD performances and complexity.

Table 4: Search window size set to achieve a trade-off between RD performances and complexity.

| Method | SW range | Execution time in % | Bit-rate gain in % |
|---|---|---|---|
| TM-A | 32 | 213 | -5.30 |
| TM-LLE | 32 | 249 | -5.89 |
| ITM-LLE | 16 | 207 | -4.94 |

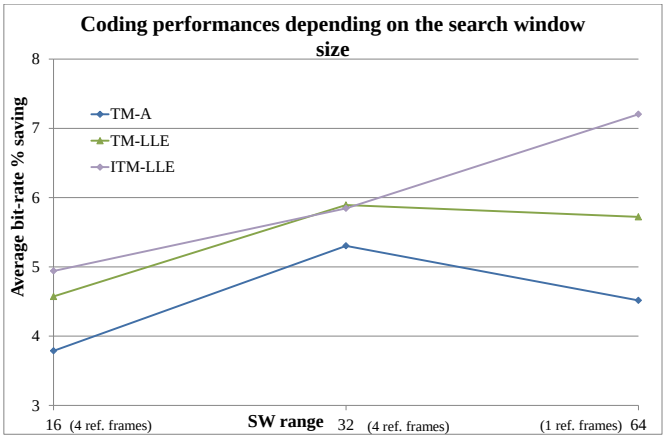


Figure 10: Coding performances obtained with the different proposed prediction methods as a function of the SW range and the number of reference frames.

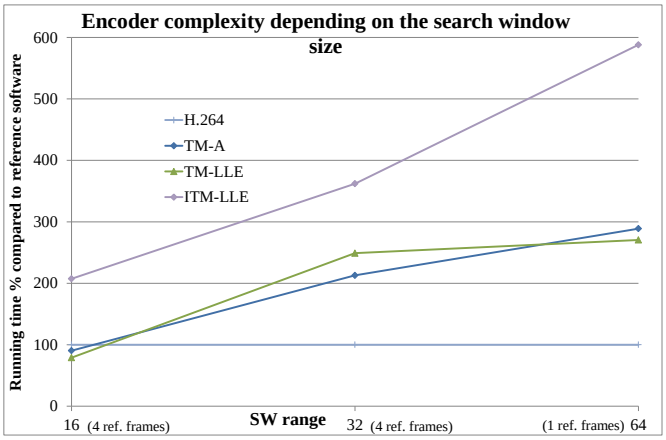


Figure 11: Encoder complexity obtained with the different proposed prediction methods as a function of the SW range and the number of reference frames.

### *4.4. Combining intra-frame and inter-frame LLE-based prediction methods*

In this section, we focus on the integration of both intra-frame and inter-frame prediction based on LLE. As in [10], the LLE-based Intra prediction method uses both TM-LLE and oITM-LLE prediction techniques in competition, integrated in H.264 by replacing two of the eight directional modes. The replaced modes are the least statistically used. The proposed Intra method is allowed for $8 \times 8$ and $4 \times 4$ partitions. The method is here denoted Intra TM/oITM-LLE. Note that this method is not only applied to I frames, but also to P and B frames. The method chosen for LLE-based Inter prediction is TM-LLE, since it allows a good trade-off between complexity and coding performances, and is here denoted Inter TM-LLE. First, the RD and complexity performances are assessed, followed by an in-depth analysis of the encoder behavior. For the Intra TM/oITM-LLE method, $K$ is set to 32, the search range is set to 32, and $L$ is set to 8. For the Inter TM-LLE method, $K$ is set to 8, the search range is set to 32, with 1 reference frame.

Table 5 shows the coding performances achieved with the Inter TM-LLE method alone, the Intra TM/oITM-LLE method alone, and the combination of both methods, against the H.264 reference. Table 6 shows the corresponding encoder complexity. The results demonstrate that the LLE-based inter-frame and intra-frame prediction methods are complementary, especially when evaluating the coding gains of the proposed methods for each sequence separately. We can see that the combined Inter TM-LLE alone and Intra TM/oITM-LLE methods can achieve up to 15.31 % bit-rate saving.

Note that, even with the combined methods, the percentage of extra-information (flag indicating if the LLE-based method is used and the index for the oITM-LLE method) only amounts in average to two to three percents of the complete bit-rate.

Table 5: Coding gains (in %) for the Inter TM-LLE, Intra TM/oITM-LLE and combined methods for each sequence.

| Sequence | Inter TM-LLE | Intra TM/oITM-LLE | Inter TM-LLE and Intra TM/oITM-LLE |
|---|---|---|---|
| Foreman | -2.73 | -1.68 | -4.40 |
| Rushes | -5.15 | -2.90 | -7.27 |
| Matrix | -2.43 | -6.53 | -6.47 |
| City | -6.57 | -3.60 | -8.80 |
| Spincalendar | -7.95 | -6.60 | -15.31 |
| Average | -4.97 | -4.26 | -8.45 |

Fig. 12 shows the coding performances, averaged over all sequences, of the combined methods as a function of the frame type. For I frames, only the Intra TM/oITM-LLE method is used, and the bit-rate saving reaches 5.92 %. For the P and B frames, the bit-rate saving reaches 7.20 % and 9.27 % respectively, which shows the gain brought by the LLE-based Inter prediction.

Table 6: Encoder complexity (in %) for the Inter TM-LLE, Intra TM/oITM-LLE and combined methods for each sequence.

| Sequence | Inter TM-LLE | Intra TM/oITM-LLE | Inter TM-LLE and Intra TM/oITM-LLE |
|---|---|---|---|
| Foreman | 101.34 | 154.81 | 210.73 |
| Rushes | 98.89 | 148.47 | 202.59 |
| Matrix | 89.85 | 187.80 | 171.14 |
| City | 94.38 | 119.13 | 233.52 |
| Spincalendar | 93.39 | 167.93 | 426.61 |
| Average | 95.57 | 155.63 | 248.92 |

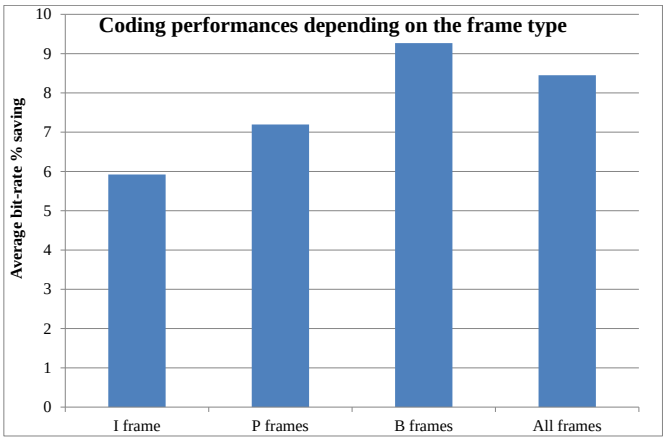

Figure 12: Coding performances of the combination of both Inter TM-LLE and Intra TM/oITM-LLE as a function of the frame type.

Fig. 13 shows the mode distribution, averaged over all sequences, as a function of the frame type. For I frames, the Intra TM/oITM-LLE method only accounts for 12.11 % of the selected modes. Paradoxically, this little amount of selection shows the method efficiency, since it still brings a significant bit-rate reduction. For the P frames, the combined LLE-based methods account for 25.98 % of the selected modes, which is more than the MCP mode (16.39 %) and the Skip mode (25.25 %). For the B frames, the combined LLE-based methods account for 19.42 % of the selected modes, which is more than the MCP mode (6.79 %), but less than the Skip mode (58.11 %). Although it is less selected than for the P frames, the combined LLE-based methods coding gains for the B frames are still significant (as shown in Fig. 12), since it is accumulated over more frames. Thus, we can see that the combined LLE-based methods amount for a significant part of the selected modes for the temporal frames.

Fig. 14 shows the mode distribution for all frames, averaged over all sequences, as a function of the quantization parameter. The results show that the mode distribution strongly varies depending on the bit-rate. The Inter TM-LLE method is much more selected at high bit-rates (QP-I 22), while at low bit-rates (QP-I 37), the Skip mode is the most selected mode. As explained in section 4.1, the LLE-based methods are mainly effective for high-frequency pseudo-periodic textures, which are well preserved at high bit-rates, but on the contrary over-smoothed at low bit-rates, which tends to favor the skip mode.

The City sequence is a good example to illustrate this kind of behavior. The RD curves of H.264 and the combined LLE-based methods for this sequence are displayed in Fig. 15, and clearly show that the RD performances of the combined LLE-based methods are better at high bit-rates.

## 5. Perspective: extension to HEVC

Even if the experiments were performed in H.264, our results still hold in HEVC. In fact, the prediction tools in HEVC follow the same principle as those used in H.264: directional propagation modes for intra-frame prediction

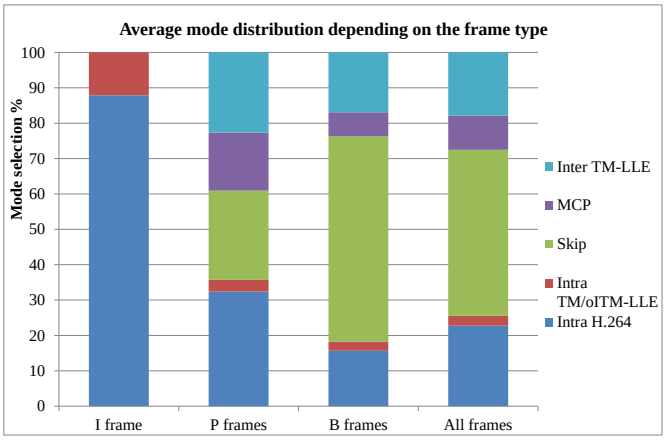


Figure 13: Distribution of the selected prediction modes as a function of the frame type.

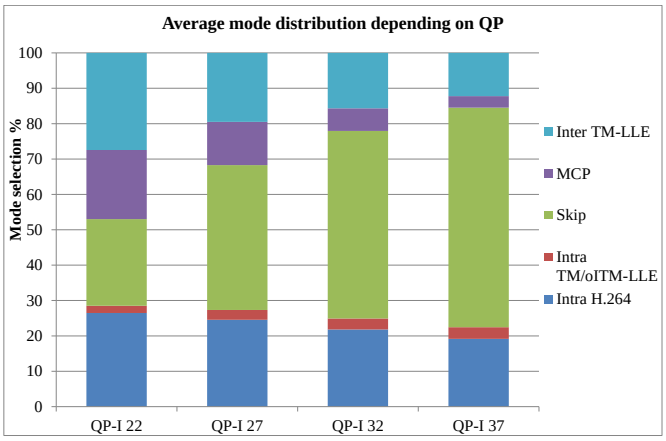


Figure 14: Distribution of the selected prediction modes as a function of the quantization parameter.

and motion estimation/compensation for inter-frame prediction. Our results show that the proposed methods are complementary with the standard tools, and improve the RD performance. Template-based prediction methods were extensively studied in H.264, and recent work shows that they can be effectively used in HEVC [13][15]. We expect our methods to outperform these techniques, as they yield a better prediction quality. The proposed methods are meant to be in competition with the standard intra-frame and inter-frame prediction tools, which would require applying them to different PU sizes. In fact, the numerous PU sizes, and especially the larger ones, are known to be an effective tool of HEVC [16]. The other tools responsible for HEVC efficiency, such as different transform sizes, SAO, are directly compatible with our methods.

## 6. Conclusion

In this paper, we have introduced new inter-frame prediction methods for video compression based on LLE. The proposed methods rely on the same multi-patches combination with different $K$-NN search strategies which are aided or not by motion information. It is shown that the methods that are not aided by the motion information give the best coding performances. This shows that our approach is an interesting alternative mode, complementary to current video compression techniques. The different methods can significantly improve the coding efficiency compared to the H.264 reference software, and the best methods outperform the state-of-the-art TM-A.

Through different experiments, we showed that the proposed methods are not extremely sensitive to the key parameters in terms of RD performances. However, the tuning of these parameters can be used to reduce the complexity.

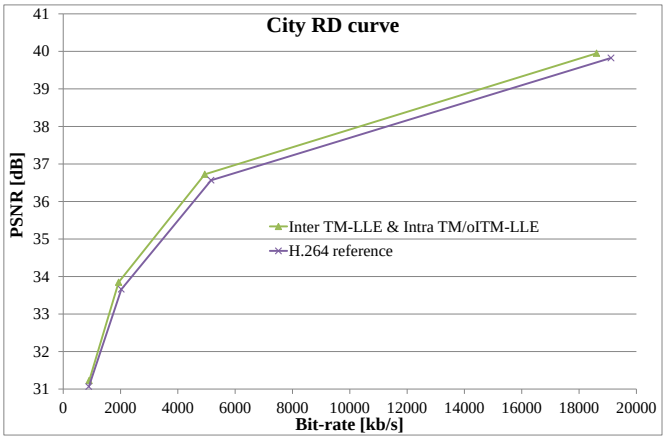


Figure 15: Rate-Distortion performances of the City sequence.

The oITM-LLE method reaches the highest bit-rate reduction, which requires in return a high computation time. Nevertheless, when applying the oITM-LLE method to intra-frame prediction, and combined with a less complex method such as TM-LLE for inter-frame prediction, we showed that significant bit-rate saving can be obtained for a reasonable complexity cost.

Furthermore, the complexity essentially comes from the $K$-NN search for which efficient and fast methods exist [19][20], as well as hardware acceleration modules. In particular, matching methods based on hash functions have been recently introduced to perform efficient NN search [21][22], and have been used to improve HEVC for screen content coding. This process is also highly parallelizable and much reduced execution times can therefore be expected. The study instead focused on the assessment of the coding performances and not on the development of an optimized paralellized implementation. Note that such optimized implementation could also allow to refine the $K$-NN search to sub-pel level, so LLE-based methods could potentially be combined with up-sampling filters.

## 7. Annex


Figure 16: Frame from the “Foreman” sequence.


Figure 17: Frame from the “Rushes” sequence.

Figure 18: Frame from the “Matrix” sequence (not representative of the whole sequence).

Figure 19: Frame from the “City” sequence.

Figure 20: Frame from the “Spincalendar” sequence.

[1] T. Wiegand, G. J. Sullivan, B. Bjontegaard, A. Luthra, Overview of the H.264/AVC video coding standard, IEEE Trans. Circuits Syst. Video Technol. 13-7 (2003) 560–576.
[2] G. J. Sullivan, J. R. Ohm, W. J. Han, T. Wiegand, Overview of the high efficiency video coding (HEVC) standard, IEEE Trans. Circuits Syst. Video Technol. 22-12 (2012) 1649–1668.
[3] K. Sugimoto, M. Kobayashi, Y. Suzuki, S. Kato, C. S. Boon, Inter frame coding with template matching spatio-temporal prediction, in: IEEE Int. Conf. Image Process. (ICIP), Vol. 1, 2004, pp. 465–468.
[4] T. K. Tan, C. S. Boon, Y. Suzuki, Intra prediction by template matching, in: IEEE Int. Conf. Image Process. (ICIP), 2006, pp. 1693–1696.
[5] T. K. Tan, C. S. Boon, Y. Suzuki, Intra prediction by averaged template matching predictors, in: IEEE Conf. Consumer Comm. Network. Conf. (CCNC), 2007, pp. 405–409.
[6] Y. Suzuki, C. S. Boon, T. K. Tan, Inter frame coding with template matching averaging, in: IEEE Int. Conf. Image Process. (ICIP), Vol. 3, 2007, pp. 409–412.
[7] M. Turkan, C. Guillemot, Sparse approximation with adaptive dictionary for image prediction, in: IEEE Int. Conf. Image Process. (ICIP), 2009, pp. 25–28.
[8] S. Roweis, L. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (2000) 2323–2326.
[9] M. Turkan, C. Guillemot, Image prediction based on neighbor embedding methods, IEEE Trans. on IP 21 (2011) 1885–1898.
[10] S. Cherigui, C. Guillemot, D. Thoreau, P. Guillotel, P. Perez, Correspondence map-aided neighbor embedding for image intra prediction, IEEE Transactions on Image Processing. 22-3 (2013) 1161–1174.
[11] S. Kamp, M. Evertz, M. Wien, Decoder side motion vector derivation for inter frame video coding, in: IEEE Int. Conf. Image Process. (ICIP), 2008, pp. 1120–1123.
[12] F. Bossen, V. Drugeon, E. Francois, J. Jung, S. Kanumuri, M. Narroschke, H. Sasai, J. Sole, Y. Suzuki, T. K. Tan, T. Wedi, S. Wittmann, P. Yin, Y. Zheng, Video coding using a simplified block structure and advanced coding techniques, IEEE Trans. Circuits Syst. Video Technol. 20-12 (2010) 1667–1675.
[13] W. H. Peng, C. C. Chen, An interframe prediction technique combining template matching prediction and block-motion compensation for high-efficiency video coding, IEEE Trans. Circuits Syst. Video Technol. 23-8 (2013) 1432–1446.
[14] Y. Shen, J. Li, Z. Zhu, Motion estimation for video coding based on sparse representation, in: IEEE Conf. Acoustics, Speech, and Signal Process. (ICASSP), 2013, pp. 1394–1398.
[15] E. Wige, G. Yammine, P. Amon, A. Hutter, A. Kaup, Sample-based weighted prediction with directional template matching for HEVC lossless coding, Picture Coding Symposium (PCS) (2013) 305–308.
[16] J. R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, T. Wiegand, Comparison of the coding efficiency of video coding standards-including high efficiency video coding (HEVC), IEEE Trans. Circuits Syst. Video Technol. 22-12 (2012) 1669–1684.
[17] KTA Software, Ver. JM 14.2 KTA 1.0. Available: http://iphome.hhi.de/suehring/tml/download/KTA/.
[18] G. Bjontegaard, Calculation of average psnr differences between rd curves, in: document VCEG-M33, ITU-T VCEG Meeting, April 2001.
[19] J. Bentley, Multidimensional binary search trees used for associative searching, Commun. ACM 18 (9) (1975) 509–517.
[20] C. Barnes, E. Shechtman, D. B. Goldman, A. Finkelstein, The generalized PatchMatch correspondence algorithm, in: European Conference on Computer Vision (ECCV), 2010.
[21] Z. Weijia, W. Ding, J. Xu, Y. Shi, B. Yin, 2-D dictionary based video coding for screen contents, in: Data Compression Conference (DCC), 2014.
[22] A. Cherian, S. Sra, V. Morellas, N. Papanikolopoulos, Efficient nearest neighbors via robust sparse hashing, IEEE Trans. Image Proc.