BLOCK PREDICTION USING APPROXIMATE TEMPLATE MATCHING

Joaquin Zepeda[1]*, Mehmet Türkan*[1,2]*, Dominique Thoreau*[1]

[1] Technicolor, 975 Avenue des Champs Blancs, Cesson Sévigné, France
[2] Izmir University of Economics, Izmir, Turkey

ABSTRACT

Template matching methods have been shown to offer bitrate savings of up to 15% when used for in-loop prediction in compression. Yet the required nearest-template search process results in prohibitive complexity. Hence, in this paper we use approximate nearest neighbor search methods to successfully address this drawback of template matching methods. Our approach uses a template index that is updated during the decoding process, yet the incurred overhead pays off in reduced nearest-template search complexity, resulting in a significant gain in template search complexity. Rate-distortion experiments further indicate that there is no rate-distortion penalty resulting from our proposed approximate template search method, and in fact a small gain of 0.1 dB is observed.

***Index Terms*—** Template matching, intra-coding, image compression, approximate nearest neighbor, indexing

1. INTRODUCTION

Closed-loop *intra prediction* is a key component of image and video compression algorithms. The term “intra” refers to the fact that the prediction technique is performed using only the information that is contained within an image or an intra-frame in a video sequence. The underlying basic idea is to first predict a block in the image using its available self-information, and then encode the prediction residue signal instead of the block itself in order to minimize the amount of information that is encoded and transmitted to the decoder. Most of the image prediction algorithms operate on image blocks in a raster scan order. The blocks usually do not overlap so that residue signals are transformed, quantized, and entropy encoded dis-jointly. The reconstructed block is finally obtained by adding the quantized residue to the prediction. For example, in H.264/AVC standard, there are two intra prediction types called Intra-16x16, Intra-8x8 and Intra-4x4 respectively [1]. The Intra-16x16 type supports four intra prediction modes while the Intra-4x4 and Intra-8x8 type supports nine modes. A macro-block of size 16x16 pixels is divided into sixteen 4x4 blocks. Each 4x4 block is predicted from prior encoded samples from spatially neighboring blocks. In addition to the so-called DC mode which consists in predicting the entire 4x4 block from the mean of neighboring pixels, eight directional prediction modes are specified. The prediction is done by simply propagating (or extrapolating) the pixel values along the specified direction. The encoder uses a tool which is called Lagrangian rate-distortion optimization (RDO) to select the best prediction type (either Intra4x4 or Intra16x16) and also the best mode(s) which have to be then transmitted to the decoder in addition to the residue signals. The state-of-the-art video compression technology is now being considered for the new standard on high-efficiency video coding (HEVC) which gives significant gains over H.264/AVC through innovations such as improved intra-prediction, larger block sizes, more flexible ways of decomposing blocks for intra- and inter-coding [2]. Intra prediction and coding in HEVC can be considered as an extension to those of H.264/AVC. The main elements in the HEVC which differ from H.264/AVC intra coding include angular prediction with 33 prediction modes (up-to 35 intra modes in total) with larger block sizes up-to 64x64 or more, quad-tree based coding structure, planar prediction, contextual information based intra mode coding.

The H.264/AVC and HEVC intra prediction approaches (as briefly described above) are suitable in the presence of contours when the directional mode chosen corresponds to the orientation of the contour. However, they tend to fail in more complex structures and highly textured areas. Thus there is still research going on for intra prediction in the existing standard to achieve better performance in intra-frame compression. An alternative method based on template matching has been widely considered for intra image prediction [3–7]. A so-called *template* is formed from previously encoded and decoded pixels in a close neighborhood of the unknown block to be predicted. The best match between the template and the candidate texture patch neighborhood (of the same shape as template), within a causal search window, allows finding the predictor of the unknown block. In [3], a prediction scheme has been proposed by replacing the H.264/AVC (Intra4x4) DC mode with template matching. This simple replacement results in an overall performance gain of 0.1-0.4 dB in intra coding. In a similar spirit, another template matching based algorithm has been described in [4]. In this approach, template matching prediction is conducted for each 2x2 sub-blocks. The four best match candidate sub-blocks finally constitute the prediction of the 4x4 block to be predicted. This method has been tested in H.264/AVC Intra4x4 as an additional prediction mode, leading to more than 11% bit-rate savings. It has later been improved in [5] by averaging multiple template matching predictors, also including larger and directional templates, resulting in more than 15% coding efficiency. There are various extensions of template matching based image prediction in the literature, e.g., a priority-based approach [7], and an adaptive illumination compensation based method [6].

For the next-generation video compression standard, one needs to look deeply into some of the challenges faced in the current standard. Keeping in mind that the intra-coding efficiency has not been sufficiently improved from H.264/AVC to HEVC and approximately 1/3 of the bit-stream is still occupied with the intra-coded frames, in this paper focus on reducing the complexity of template matching in order to enjoy the large intra-coding bit-rate gains it offers at reasonable overhead. To this end, we use existing indexing methods [8–11] based on data partitioning to carry out an approximate nearest neighbor search instead of the exhaustive search employed by all existing methods. The indexing structures used are based on K-means and are learned offline and populated during image decoding by the templates made available by each newly decoded image block. The overhead incurred in populating the template index pays off in reduced search complexity, for a total decoding complexity up to $2.3\times$ smaller.

The remainder of this paper is organized as follows: In Section 2, we review template matching and set the basis for the presentation of our algorithm, which we do in Section 3. We then present our experimental results in Section 4 and concluding remarks in Section 5.

2. BLOCK-PREDICTION BASED ON TEMPLATE MATCHING

We assume that we are given an image of size $r \times c$ from which we extract overlapping blocks of size $b \times b$ pixels. Let $\mathbf{y}_i \in \mathbb{R}^{b^2}$ represent the i-th such block which has pixel i at its upper left corner. We are concerned with obtaining a prediction $\hat{\mathbf{y}}_i$ of the value of $\mathbf{y}_i$ given only a vector $\tilde{\mathbf{z}}_i \in \mathbb{R}^a$, referred to as the *template* for block i, consisting of pixels from the previously-decoded region, referred to as the *causal region*. Since the prediction $\hat{\mathbf{y}}_i$ is available both at the decoder and encoder side, the encoder need only send a compressed version $\tilde{\mathbf{r}}_i$ of the residual $\mathbf{r}_i = \mathbf{y}_i - \hat{\mathbf{y}}_i$, and the decoder can obtain an approximation $\tilde{\mathbf{y}}_i$ of block $\mathbf{y}_i$ from

$$\tilde{\mathbf{y}}_i = \hat{\mathbf{y}}_i + \tilde{\mathbf{r}}_i. \qquad (1)$$

Note that, for rate efficiency, the encoder only computes and sends encoded residuals for non-overlapping blocks. We will use subscript t to denote non-overlapping block indices, noting that t takes non-contiguous values, with $t = (krb+lb)$ for the block at row k and column l of the grid of non-overlapping blocks. Using the standard raster-scan (left-to-right, top-to-bottom) encoding/decoding order implies that positions $t \in \{krb + lb\}_{kl}$ are processed in order of increasing value of t.

A very common layout (used in video coding standards [2]) for the position of the pixels in $\tilde{\mathbf{z}}_t$ relative to those in block $\mathbf{y}_t$ is illustrated in Fig. 1: the template $\tilde{\mathbf{z}}_t \in \mathbb{R}^a$ is built from the $a = 2b + 1$ pixels constituting the top and left boundaries of the block $\mathbf{y}_t$. Using such a layout is compatible with raster-scan processing and incurs very low prediction complexity, as only a small number of the previously-decoded pixels are used.

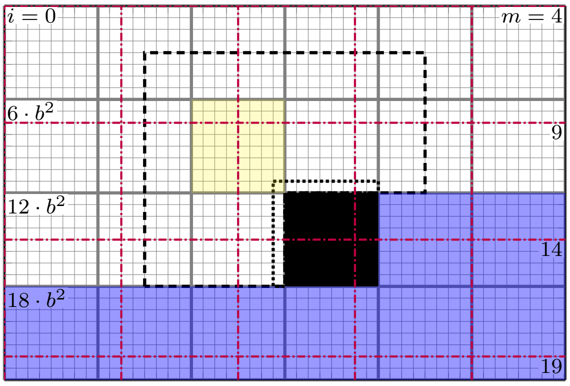

Fig. 1. Standard spatial layout of template $\tilde{\mathbf{z}}_t$ (delineated in dots) and block to predict $\mathbf{y}_t$ (in black). Each position i has an associated template $\tilde{\mathbf{z}}_i$ and block $\mathbf{y}_i$. Only non-overlapping blocks $\mathbf{y}_t, t \in \{kcb + lb\}_{kl}$ (delineated by solid, thick gray lines) are encoded, but all templates $\tilde{\mathbf{z}}_i$ are indexed. The templates $\tilde{\mathbf{z}}_i, i \in \mathcal{T}_t$ (those having upper-left pixel in the yellow region) are indexed at the same time that block $\mathbf{y}_t$ is encoded/decoded. The processing is done following a raster scan order. The *causal region* consisting of pixels decoded before block t is denoted in white, and the *causal neighborhood* of block t is delineated with a dashed line. *Spatial bins* $\mathcal{B}_m$ are delineated by dash-dotted red lines. In this example, $r = 32$, $c = 48$, $a = 17$, $b = 8$, and $|\mathcal{B}_m| = 10^2$ (except for bins at the bottom and right boundaries).

Given the template $\tilde{\mathbf{z}}_t$, many possible methods exist to obtain a prediction $\hat{\mathbf{y}}_t$ for block t. For example, intra-coding prediction used in recent video standards consists of copying the pixels in $\tilde{\mathbf{z}}_t$ along one of several possible directions across block t. In this work, however, we are interested in template matching methods that use the previously decoded blocks $\tilde{\mathbf{y}}_i$, $i \in \mathcal{S}(t)$, to predict the current block $\mathbf{y}_t$, where $\mathcal{S}(t)$ denotes all pixel positions i with a related template $\tilde{\mathbf{z}}_t$ and block $\tilde{\mathbf{y}}_t$ not falling outside the image and within the causal region, as per the layout in Fig. 1.

For example, one simple approach uses the pixels $\tilde{\mathbf{y}}_j$ corresponding to the nearest template $\tilde{\mathbf{z}}_j$ as the prediction for $\mathbf{y}_t$:

$$\hat{\mathbf{y}}_t = \tilde{\mathbf{y}}_j \text{ s.t. } j = \underset{i \in \mathcal{S}(t)}{\operatorname{argmin}} \|\tilde{\mathbf{z}}_t - \tilde{\mathbf{z}}_i\|^2 \tag{2}$$

A second approach instead uses a linear combination of the vectors $\tilde{\mathbf{y}}_{j_k}, k = 1, \ldots, K$ corresponding to the K nearest templates $\tilde{\mathbf{z}}_{j_k}$ of $\tilde{\mathbf{z}}_t$,

$$j_1, \ldots, j_K \text{ s.t. } \|\tilde{\mathbf{z}}_t - \tilde{\mathbf{z}}_{j_k}\| \leq \|\tilde{\mathbf{z}}_t - \tilde{\mathbf{z}}_i\| \quad \forall i \notin \{j_k\}_{k=1}^K, \\ \text{where } i, j_k \in \mathcal{S}(t). \tag{3}$$

Given the indices j_k, we use the matrix of nearest templates

$$\tilde{\mathbf{Z}}_t = [\tilde{\mathbf{z}}_{j_1}| \ldots |\tilde{\mathbf{z}}_{j_K}] \tag{4}$$

and the matrix containing the corresponding blocks from the causal region

$$\tilde{\mathbf{Y}}_t = [\tilde{\mathbf{y}}_{j_1}| \ldots |\tilde{\mathbf{y}}_{j_K}] \tag{5}$$

to compute an approximation for block t as follows:

$$\hat{\mathbf{y}}_t = \tilde{\mathbf{Y}}_t \left(\tilde{\mathbf{Z}}_t^+ \tilde{\mathbf{z}}_t\right), \tag{6}$$

where the superscript $+$ symbol denotes the pseudo-inverse operation.

Note that the expressions in (3) and (6) reduce to the case (2) when we set $K = 1$, and hence from now on we discuss only the general case of K nearest neighbor-based template matching.

Border effects. For the benefit of presentation, we have ommitted discussing template matching for border positions. Being a common problem to all template-matching-based prediction methods, we assume that a standard solution is employed consisting, for example, of encoding the first several rows and columns of the image using an alternative encoding, and hence we assume that a template and a causal region is available for all positions i in the image.

2.1. Analysis of template matching complexity

One of the biggest drawbacks of prediction methods based on template matching is the increased decoder complexity related to the neighbor search operation (3). In order to better understand the complexity problem, we now derive complexity bounds for the K-nearest neighbor template matching operation. Letting $r \times c$ denote the image dimensions and b the block size, the complexity of the K nearest neighbor search operation is then $\mathcal{O}\left(\log(K) a (rc)^2\right)$. This complexity bound assumes a min-heap implementation wherein a list of K best distances is kept in a sorted state and updated whenever a better neighbor template is found. Yet even with this optimization, the complexity can be quiet high.

One way to reduce this complexity is to instead consider only templates at a subset of positions in the image. For example, considering only a fraction of $\alpha \leq 1$ possible horizontal or vertical template positions reduces the complexity to

$$\mathcal{O}\left(\log(K) a (rc\alpha)^2\right), \tag{7}$$

representing a decrease factor of α^2. The possible positions can be taken to be the *causal neighborhood* of $(\alpha r) \cdot (\alpha c)$ pixel positions closest to the current block t, as illustrated by the dash-delineated region in Fig. 1. Alternatively, the positions can be taken from a $\frac{1}{\alpha}$-downsampled pixel-positions grid.

Sub-pixel template matching. An extension of template matching that has been shown to improve the prediction accuracy is to further consider patches at sub-pixel positions. In terms of complexity, this amounts to using $\alpha > 1$. Given the squared complexity dependence on α, this increased predictive accuracy comes at a steep price.

3. ACCELERATING TEMPLATE MATCHING USING APPROXIMATE SEARCH METHODS

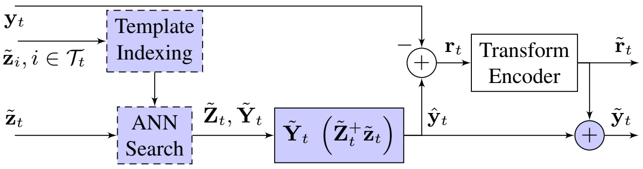
$\mathbf{y}_t$
$\tilde{\mathbf{z}}_i, i \in \mathcal{T}_t$
Template Indexing
$\tilde{\mathbf{z}}_t$
ANN Search
$\tilde{\mathbf{Z}}_t, \tilde{\mathbf{Y}}_t$
$\tilde{\mathbf{Y}}_t \left(\tilde{\mathbf{Z}}_t^+ \tilde{\mathbf{z}}_t\right)$
$-$
$+$
$\mathbf{r}_t$
Transform Encoder
$\tilde{\mathbf{r}}_t$
$\hat{\mathbf{y}}_t$
$\tilde{\mathbf{y}}_t$

Fig. 2. Encoding and decoding process for block $\mathbf{y}_t$, $t \in (krb + lb)_{kl}$. Shaded components are common to both encoder and decoder; white components are exclusive to the encoder. Components with dashed borders are novel relative to the standard predictive/residual encoder.

In this work we propose using Approximate Nearest Neighbor (ANN) search methods to accelerate the template matching process. We present a data-partitioning acceleration method based on K-means [8] clustering as well as a hierarchical extension. Hierarchical K-means [9] and related methods based on residual quantization [10] and tree-based data partitioning [11] have produced state-of-the art approximate nearest neighbor search methods in several applications related to image search. In order to make it possible to efficiently combine our ANN template matching methods with the α-subset schemes described above, we further propose arranging the templates in each data partition into spatial bins.

3.1. Data partitioning approach

We assume that we are given a training set $\mathcal{Z} = \{\mathbf{z}_l \in \mathbb{R}^a\}_l$ of templates extracted from a large number of training images. The data partitioning methods we use are based on a set of N codewords

$$\mathbf{c}_1, \ldots, \mathbf{c}_N = \underset{\mathbf{a}_1, \ldots, \mathbf{a}_N}{\operatorname{argmin}} \sum_l \min_n \|\mathbf{a}_n - \mathbf{z}_l\|^2 \tag{8}$$

learned using the K-means clustering algorithm. The codewords thus obtained implicitly define a partitioning of $\mathbb{R}^a$ into

N cells

$$\mathcal{C}_n = \{\mathbf{z} \in \mathcal{R}^a \text{ s.t. } n = \underset{m}{\operatorname{argmin}} \|\mathbf{z} - \mathbf{c}_m\|\}, n = 1, \ldots, N. \tag{9}$$

We use such a codebook at the encoder and decoder as follows (*cf.*, Fig. 2): given the template $\tilde{\mathbf{z}}_t$, an approximate search is performed (in block *ANN Search*) for the K nearest templates. The search consists of first finding the codeword $\mathbf{c}_{n_t}$ closest to $\tilde{\mathbf{z}}_t$, and then exhaustively searching the related list

$$\mathcal{L}_{n_t} = \{i | i \in \mathcal{S}(t), \tilde{\mathbf{z}}_i \in \mathcal{C}_{n_t}\}. \tag{10}$$

The resulting K approximate nearest templates and their associated blocks are used to generate $\tilde{\mathbf{y}}_t$ using (6) and (1).

Simultaneous to the decoding process, the templates $\tilde{\mathbf{z}}_i$ (those with upper-left corner in the yellow region in Fig. 1) at positions $i \in \mathcal{T}_t$, $|\mathcal{T}_t| \leq b^2$, corresponding to blocks $\tilde{\mathbf{y}}_i$ that share pixels with the block $\tilde{\mathbf{y}}_t$ currently being decoded (and the causal region) need to be added to the template index for latter processing (block *Template Index*): For every such position i, the template $\tilde{\mathbf{z}}_i$ is assigned to a list $\mathcal{L}_{n_i}$ associated to its closest codeword $\mathbf{c}_{n_i}$, incurring a cost of $\mathcal{O}(rcNa)$ operations per image. This complexity follows from assigning the templates at each position i to a list once, and includes the assignment required to find list $\mathcal{L}_{n_t}$ in (10).

Assuming that templates are equally distributed across cells (*i.e.*, $|\mathcal{C}_n| = rc/N$, $\forall n$), the cost for the entire image of exhaustively searching the lists $\mathcal{L}_{n_t}, \forall t$, when restricted to the α-causal neighborhood, is $\mathcal{O}\left(\frac{1}{N}(a + \log(K)) \cdot \left(\frac{rc\alpha}{b}\right)^2\right)$. Hence the total complexity of ANN template matching for the entire image is

$$\mathcal{O}\left(\frac{1}{N}(a + \log(K)) \cdot \left(\frac{rc\alpha}{b}\right)^2 + rcNa\right). \tag{11}$$

The last term represents the overhead of indexing all templates. In order to gain from this approach, this overhead should be comparable to the previous terms, and from this we can derive the following indication of a good value for N, where we use the approximation $(a + \log(K))/a \simeq 1$:

$$N \simeq (rc)^{1/2} \frac{\alpha}{b}. \tag{12}$$

3.2. Efficient approximate search in causal neighborhood

Enabling an efficient implementation of an approximate search method inside the causal neighborhood requires that we spatially index the lists L_n so that finding the intersection between those templates indicated by list L_{n_t} and the causal neighborhood of pixel position t can be done with negligible overhead. The approach we propose consists of first defining a uniform rectangular grid of spatial bins $\mathcal{B}_m, m = 0, \ldots, M-1$, as illustrated by the double-lined grid in Fig. 1. The list $\mathcal{L}_n$ is accordingly dividied into spatial bins $\mathcal{L}_n^m, m = 0, \ldots, M-1$, satisfying

$$\mathcal{L}_n^m = \{i | i \in B_m, i \in \mathcal{L}_n\}. \tag{13}$$

Keeping the list $\mathcal{L}_n$ organized in spatial bins accordingly during the encoding and decoding process requires negligible extra complexity. Note that the complexity bounds indicated above are for the extreme case where $|\mathcal{B}_m| \simeq 1$.

4. EXPERIMENTS

We use an efficient C implementation of our approximate template matching method to carry out evaluations. We train codebooks using the K-means algorithm on a *training set* of 10^6 templates extracted form 10^4 images randomly downloaded from *Flickr*. We use a different set of 1491 images [12] as our *test set* and compute the prediction PSNR $20 \cdot \log_{10}\left(\frac{255}{\|\mathbf{y}_t - \hat{\mathbf{y}}_t\|}\right)$ (top graph in Fig. 3 and Fig. 4) and complexity, defined as the execution time in ms/pixel (bottom graph), both averaged over all test images. We further break out complexity into four components: *compression* (Transform Encoder block and both addition operation in Fig. 2), *indexing* (Template Indexing block), *searching* (ANN Search block) and *prediction* ($\tilde{\mathbf{Y}}_t \left(\tilde{\mathbf{Z}}_t^+ \tilde{\mathbf{z}}_t\right)$ block). The default parameters used, selected empirically, are: number of codewords $N = 10$, multiple assignment factor $B = 1$, causal neighborhood size $S = \alpha_r \cdot r = \alpha_c \cdot c = 16$, number of templates $K = 2$, and block size $b = 8$.

Fig. 3 summarizes the complexity improvements that can be obtained using our method: The baseline complexity (0.583 ms) corresponds to the case $N = 1$, where an exhaustive search of all templates in the causal neighborhood is carried out. For $N = 10$, our method runs 2.23× faster (0.261 ms). Interestingly, the prediction PSNR increases slightly with increased number of codewords, and we believe this is because the approximate search results in blocks $\tilde{\mathbf{Y}}_t$ with slightly increased diversity, and this benefits the averaging done during prediction. Note that the searching complexity (respectively, indexing complexity) decreases (increases) monotonically with increasing N, as predicted in (11), which we plot as a dashed line in the figure (with adequate scaling). Note also that the value $N = 2$ predicted by (12) is within an order of magnitude of the experimental optimum given by $N = 10$.

In Fig. 4 we evaluate performance of our method as a function of the block size b. The indexing penalty incurred by our method becomes too large for increasing block sizes. This is in part because of the increase in template space dimensionality b, but also because the number of blocks to encode decreases inversely proportionally to b^2, while the number of templates to index remains constant. Methods exist that reduce the indexing complexity by using hierarchical codebooks or cartesian products of codebooks, and we plan to explore these in the future.

In Fig. 5, we plot rate-distortion curves for image *Lena* when using our approximate template matching method and the baseline exhaustive template matching approach. We use

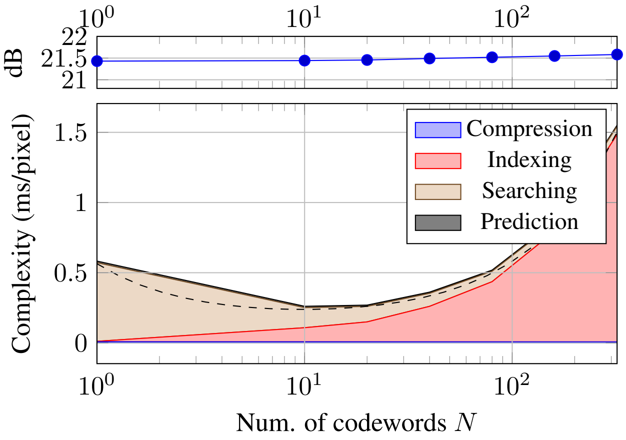

Fig. 3. Effect of number of codewords N. The dashed line represents the complexity bound in (11)

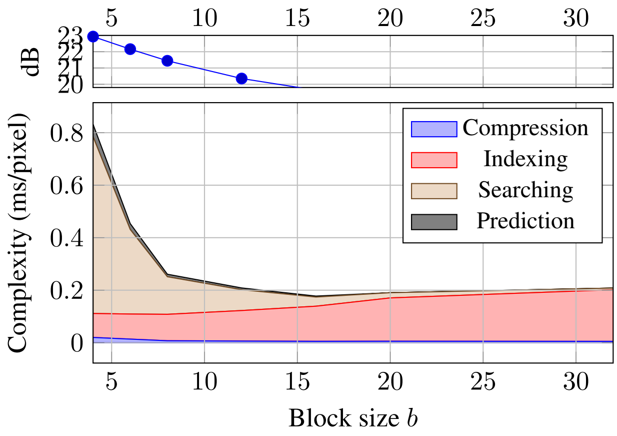

Fig. 4. Effect of block size b.

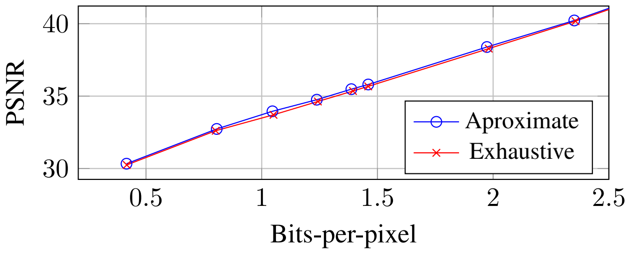

Fig. 5. Rate-distortion curve for lena image, approximate vs. exhaustive template matching.

a DCT transform with JPEG-like scaling of transform coefficients, followed by rounding and differential encoding using zig-zag scanning, plotting the entropy of the resulting stream in bits-per-pixel. As observed, our proposed method is not only faster, but it also results in slightly better performance over all rates, and this is not unexpected given the improving PSNR trend in Fig. 3.

5. CONCLUSION

Template matching methods provide increased predictive power that can result in as much as a 15% bit-rate savings. Yet the large complexity increase of searching the best matching templates limits the applicability of these methods. In this paper we use approximate nearest neighbor search techniques to successfully address this problem, resulting in complexity reductions of $2.23\times$. Experimental tests indicate that there is no rate-distortion penalty incurred by our method (in fact, there is a small ~ 0.1 dB gain for *Lena*).

REFERENCES

[1] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[2] G. J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec 2012.

[3] J. Yang, B. Yin, Y. Sun, and N. Zhang, "A block-matching based intra frame prediction for H.264/AVC," in *Proc. IEEE Int. Conf. Multimedia and Expo*, 2006, pp. 705–708.

[4] T. K. Tan, C. S. Boon, and Y. Suzuki, "Intra prediction by template matching," in *Proc. IEEE Int. Conf. Image Process.*, 2006, pp. 1693–1696.

[5] ——, "Intra prediction by averaged template matching predictors," in *Proc. IEEE Consumer Comm. Network. Conf.*, 2007, pp. 405–409.

[6] Y. Zheng, P. Yin, O. D. Escoda, X. Li, and C. Gomila, "Intra prediction using template matching with adaptive illumination compensation," in *Proc. IEEE Int. Conf. Image Process.*, 2008, pp. 125–128.

[7] Y. Guo, Y. K. Wang, and H. Li, "Priority-based template matching intra prediction," in *Proc. IEEE Int. Conf. Multimedia Expo.*, 2008, pp. 1117–1120.

[8] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," *International Conference on Computer Vision*, pp. 2–9, 2003. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1238663

[9] D. Nistér and H. Stewénius, "Scalable recognition with a vocabulary tree," pp. 2161–2168, 2006. [Online]. Available: http://www.vis.uky.edu/ stewe/ukbench/

[10] H. Jegou, F. Perronnin, M. Douze, S. Jorge, P. Patrick, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–12, 2011.

[11] A. W. Moore, "Efficient Memory Based Learning for Robot Control," in *An intoductory tutorial on KD-trees*, 1991, ch. 6.

[12] H. Jegou, "INRIA Holidays dataset." [Online]. Available: http://lear.inrialpes.fr/people/jegou/data.php