Interframe Coding of Feature Descriptors for Mobile Augmented Reality

Mina Makar, *Member, IEEE*, Vijay Chandrasekhar, *Member, IEEE*, Sam S. Tsai, *Member, IEEE*, David Chen, *Member, IEEE*, and Bernd Girod, *Fellow, IEEE*

***Abstract*— Streaming mobile augmented reality applications require both real-time recognition and tracking of objects of interest in a video sequence. Typically, local features are calculated from the gradients of a canonical patch around a keypoint in individual video frames. In this paper, we propose a temporally coherent keypoint detector and design efficient interframe predictive coding techniques for canonical patches, feature descriptors, and keypoint locations. In the proposed system, we strive to transmit each patch or its equivalent feature descriptor with as few bits as possible by modifying a previously transmitted patch or descriptor. Our solution enables server-based mobile augmented reality where a continuous stream of salient information, sufficient for image-based retrieval, and object localization, is sent at a bit-rate that is practical for today's wireless links and less than one-tenth of the bit-rate needed to stream the compressed video to the server.**

***Index Terms*— Augmented reality, canonical patch coding, descriptor coding, image matching.**

I. Introduction

MOBILE Augmented Reality (MAR) systems are becoming more important with growing interest in head-mounted displays [1] and applications that use image-based retrieval on mobile devices [2]. Streaming MAR applications require both real-time recognition and tracking of objects of interest in a video scene and many of these applications utilize robust local image features. Examples of these local features include the Scale-Invariant Feature Transform (SIFT) [3], Speeded Up Robust Features (SURF) [4] and Compressed Histogram of Gradients (CHoG) [5]–[7].

In an image matching framework, keypoints are detected in both database and query images. These keypoints should be shift-, scale- and rotation-invariant and also should be repeatably detectable in different images of the same object or scene. After keypoint detection, feature descriptors are calculated for every keypoint. Similarity of two descriptors is evaluated using a suitable distance metric such as the L_2 norm or symmetric KL divergence [8]. Many feature descriptors [3]–[8] share the common framework that the descriptor consists of histograms of gradients in a patch located around the detected keypoint. For scale and rotation invariance, patches are oriented along the dominant gradient direction and resized according to the scale of the detected feature. We refer to these oriented and scaled patches as canonical patches.

For applications where data are transmitted over a network for feature detection and image matching, it is desirable that the amount of data sent is as low as possible to reduce the latency and the cost of the system. Several algorithms for feature descriptor compression have been presented [9]. The emerging MPEG standard, Compact Descriptors for Visual Search (CDVS) [10] targets low bit-rate descriptors that achieve state-of-the-art image matching performance. Several state-of-the-art techniques [11]–[14] have been proposed and compared in a common evaluation framework [15]. In our previous work [16], [17], we show that one can also directly transmit compressed canonical patches and perform descriptor computation at the receiving end with comparable performance. This has the advantage of allowing interoperability between systems using different feature descriptors.

In a streaming MAR system, the independent detection of keypoints and encoding of feature descriptors for every video frame does not exploit the temporal redundancy in the video, and can lead to a bit-rate even larger than the bit-rate needed for an efficiently compressed video stream. To exploit this temporal redundancy, we design a temporally coherent keypoint detector, and strive to transmit each patch or its equivalent feature descriptor with as few bits as possible by simply modifying a previously transmitted patch or descriptor. Initial results of our work have been presented in [18] and [19]. In this paper, we briefly review the work in [18] and [19], propose novel encoding techniques for interframe coding of feature descriptors and keypoint locations, and provide detailed experimental results for image matching and retrieval performance.

A lot of work has been done on improving the performance of image retrieval based on the bag of words approach by Sivic and Zisserman [20], and developing feature descriptor compression techniques when the query is a snapshot image of the object of interest [9]–[17]. However, no viable solutions

Manuscript received February 11, 2014; revised April 21, 2014; accepted June 3, 2014. Date of publication June 17, 2014; date of current version July 1, 2014. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Anthony Vetro.

M. Makar is with Qualcomm Inc., San Diego, CA 92121 USA, and also with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: minamakar@alumni.stanford.edu).

V. Chandrasekhar is with the Institute for Infocomm Research, Singapore 138632, and also with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: vijay.cmu@gmail.com).

S. S. Tsai, D. Chen, and B. Girod are with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: sstsai@stanford.edu; dmchen@stanford.edu; bgirod@stanford.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2014.2331136

have been reported in the literature for streaming MAR that exploit the temporal correlation between feature descriptors in successive video frames to reduce the bit-rate. To the best of our knowledge, this problem has been solved for the first time by the work reported in this paper.

The remainder of the paper is organized as follows. Section II reviews the proposed temporally coherent keypoint detector. Section III presents different patch encoding modes that exploit temporal correlation. In Section IV, we propose inter-descriptor coding techniques for lattice-coded and uncompressed feature descriptors. A solution for differential location coding is presented in Section V. Section VI compares the performance of our proposed system to a system that streams the whole video to the server, as well as a system that relies on the independent detection of keypoints frame-by-frame. Finally, in Section VII, we present image retrieval results from a large database, and propose system architectures that exploit the temporal correlation in the video to reduce the complexity of image retrieval.

II. Temporally Coherent Keypoint Detector

Keypoints that are detected independently in successive video frames have an associated location, scale, and orientation of limited accuracy. Moreover, when the response of keypoint detector is near the detection theshold, keypoints might even disappear briefly and then reappear, or only appear for a few frames. These temporal variations preclude effective interframe coding of canonical patches or feature descriptors.

A temporally coherent keypoint dectector can overcome these challenges. To achieve temporal coherence, we propagate patches to consecutive video frames. Tracking of interest points in a video has been studied extensively, e.g., Kanade-Lucas-Tomasi tracker [21] and Mean-Shift tracker [22]. Our work [18], [19] introduces the idea of tracking to propagate canonical patches. We perform *canonical patch matching* for efficient patch propagation.

A. Canonical Patch Matching

We divide the video frames into two categories: *Detection frames* or *D-frames* and *Forward Propagation frames* or *FP-frames*. In D-frames, we run a conventional keypoint detection algorithm to detect the interest points in the video frame. In this paper, we use the SIFT keypoint detector [3], and refer to the patches corresponding to the detected keypoints as D-Patches. In FP-frames, patch matching is performed and, as a result, each patch is connected with a patch in the previous frame. We refer to these patches as Forward Propagation Patches or FP-Patches. For each patch in the previous video frame, we search for a corresponding patch in the current frame that minimizes a distortion measure. In this paper, we use the Sum of Absolute Differences (SAD) as a distortion measure. To search for the minimum distortion patches, we apply patch matching as follows.

Each patch in the previous frame is associated with a location, an orientation and a scale. We start from the location of the patch in the previous frame and define a search range to search for the most correlated patch in the current frame.

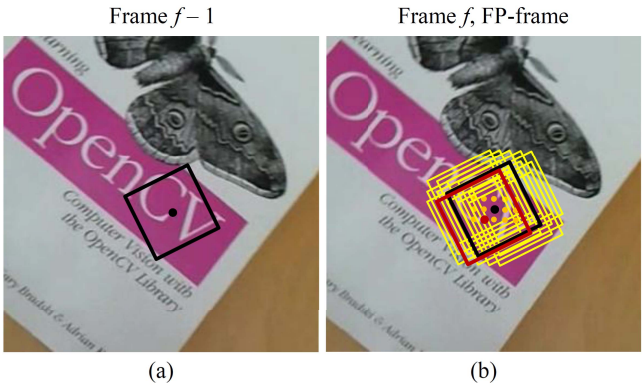


Fig. 1. Illustration of patch matching. (a) Example patch in the previous frame to be tracked in the current frame. (b) Patch matching searches for the patch of minimum SAD with respect to the patch in (a). We vary the keypoint location to cover the search range. The patch highlighted with a red square represents the result that minimizes the Sum of Absolute Differences (SAD).

Fixing the orientation and scale, we vary the x and y values in small steps around the starting location until we cover the whole search range. For each value of x and y, we extract the corresponding patch in the current video frame and measure its SAD distortion relative to the patch in the previous frame. The patch with the minimum SAD is selected to be the corresponding FP-Patch in the current frame. An FP-Patch then inherits orientation and scale from the initial D-Patch. This idea is illustrated in Fig. 1.

In some cases, we cannot find a good match to propagate an FP-Patch from the previous frame. This occurs when the minimum SAD during patch matching exceeds a threshold SAD_{term}. In this case, we terminate patch propagation, and reduce the number of patches in the current FP-frame through a termination flag for the patch in the previous frame. Patch termination reduces the bit-rate and eliminates gratuitous patches which might deteriorate image retrieval. After propagating the patches from frame $f-1$ to frame f, the patches in frame f are used as a starting point and are propagated to frame $f+1$ and so on. After a few frames, the propagated patches begin to deviate from the actual interest points that they correspond to. Thus, after a pre-defined number of frames, we insert a D-frame by running the keypoint detector instead of propagating patches. We refer to the number of frames between consecutive D-frames as the *detection interval* Δ.

When we only vary the x and y locations of the patches during patch matching, we observe that the patch propagation fails where there is complex motion in the video, and the result is a large drop in the number of feature matches. This problem can be mitigated by using a short detection interval Δ, but this in turn causes a large increase in the transmission bit-rate. We introduce a more sophisticated motion model to accurately track over longer time periods. Since each patch is characterized by its location, orientation and scale, we vary orientation and scale along with location during patch matching. Thus, we permit a similarity transform with four degrees of freedom between corresponding patches in consecutive video frames. The similarity transform improves patch tracking and introduces a significant gain in image matching

performance, especially when using a large Δ and during periods of camera rotation around the optical axis, looming motion and zoom.

Since patch matching is inspired by the block matching technique used in modern video coding standards such as H.264 [23], we can comment on the computational complexity of patch matching by comparing it to the computational complexity of motion estimation using block matching in H.264. H.264 real-time encoders are available in modern smart phones and tablets with reasonable computational complexity. Compared to a conventional H.264 encoder, patch matching requires much lower computational complexity. First, we use fixed size patches, while H.264 uses variable block sizes to better describe the macroblock to be encoded. Second, we usually perform patch matching for about 200 patches in each frame for efficient image matching and retrieval performance. This number is lower than the number of macroblocks in common video resolutions encoded on mobile devices (VGA resolution video contains 1200 macroblocks/frame). Finally, for each macroblock, H.264 performs mode decision in a rate-distortion optimal framework, which is not the case for patch matching.

In the video coding literature, the problem of fast motion estimation has been studied extensively. Many techniques that speed-up block matching by more than an order of magnitude, compared to brute force search, are available [24], [25]. Hence, fast search methods can be applied to obtain temporally coherent patches as well.

B. Adaptive Detection Interval

A fixed Δ may be inefficient when the tracked FP-Patches are no longer well representing the objects of interest in a particular video frame. This happens with occlusions or when objects of interest are entering or leaving the scene. We propose an adaptive Δ where we insert a D-frame whenever patch matching deteriorates. To decide whether the current video frame is a D-frame or an FP-frame, we acquire both types of patches for the frame. For each patch in the D-Patches we search for a corresponding patch in the FP-Patches with the minimum SAD. We define the deviation measure $D\text{-}FP_{SAD}$ as the average minimum SADs between the D-Patches and their closest corresponding FP-Patches.

A small deviation means that the FP-Patches are representing the scene well; thus, we keep the FP-Patches and decide an FP-frame. A large deviation indicates that the D-Patches contain information not well captured by the FP-Patches and hence, we decide a D-frame. The decision is controlled by a threshold $D\text{-}FP_{th}$. If $D\text{-}FP_{SAD}$ is smaller than the threshold, we decide an FP-frame; otherwise, we decide a D-frame. Once the frame type is decided, the other type of patches is discarded. To avoid the build-up of small deviations, we limit the detection interval to Δ_{max}. If the number of consecutive FP-frames reaches Δ_{max}, we automatically insert a D-frame.

We report experiments on eight videos from the *Stanford Streaming MAR dataset* [26]. These videos, which are shown in Fig. 2, contain a single object of interest, recorded with a hand-held mobile device with different amounts of camera

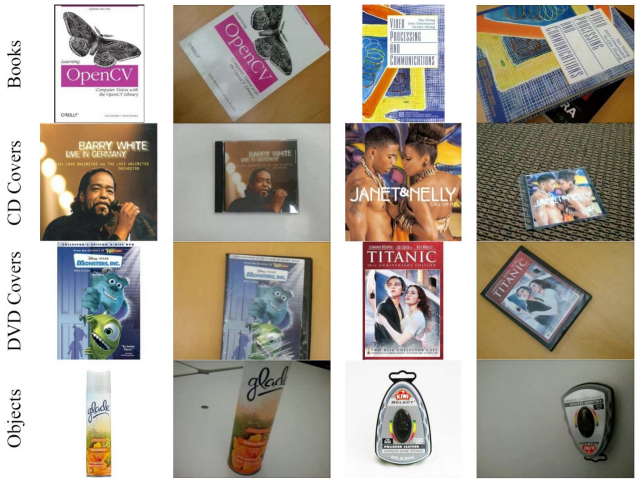


Fig. 2. Database images and example video frames from the *Stanford Streaming MAR dataset*.

motion, glare, blur, zoom, rotation and perspective changes. Each video is 100 frames long, recorded at 30 fps with resolution 640 $\times$ 480. The videos, in the order shown in figure, are *OpenCV Book*, *Wang Book*, *Barry White*, *Janet Jackson*, *Monsters Inc*, *Titanic*, *Glade* and *Polish*. We present the clean database image alongside an example frame from the corresponding video. From the decoded patches, we calculate modified SIFT descriptors as explained in Section IV, and use symmetric KL divergence as a distance metric for descriptor matching, and RANSAC [27] for geometric consistency check.

In our experiments, we apply a coarse-to-fine search strategy for the x and y locations during patch matching. The initial search range is from -24 to 24 in steps of 2 with respect to the original pixel raster of the video frame. The finer search stage uses patches with varying x and y locations between -1.75 and 1.75 in steps of 0.25 around the x and y locations of the best initial guess to fine-tune the matching patch to quarter pixel accuracy. These numbers are chosen based on the video resolution and the extent of the motion of the object of interest. We also find that the changes in orientation and scale are small between consecutive frames, and a small search range with few search values is enough.

To minimize the bit-rate, we only transmit patches corresponding to features that are more likely to match. We use the same feature selection algorithm as the CDVS Test Model [11] and we set the maximum number of extracted patches from a video frame to 200. The feature selection algorithm presented in [28] chooses the features with higher matching probabilities based on their scale, orientation, output of the DoG pyramid and distance from the image center. We study the feature matching performance of 8×8, 16×16 and 32×32 patches. For patch termination, we use $SAD_{term} = 450$ for 8×8 patches, 1800 for 16×16 patches and 7200 for 32×32 patches.

Fig. 3 presents the results of using an adaptive detection interval with $D\text{-}FP_{th} = 450$ for 8×8 patches, 1800 for 16×16 patches or 7200 for 32×32 patches, and $\Delta_{max} = 30$. We choose the values of $D\text{-}FP_{th}$, Δ_{max} and SAD_{term} based on a training phase. We use the video sequences from [18] for training purpose. These video sequences are not part

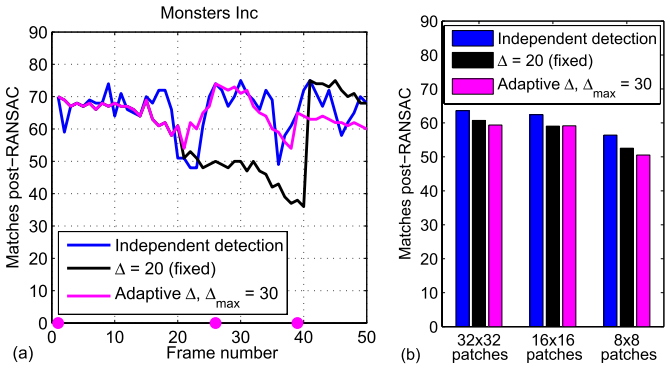


Fig. 3. Matching performance of patch matching technique when using a fixed versus an adaptive detection interval. Adaptive Δ better follows the performance of independent detection. (a) Example video: *Monsters Inc*, D-frames are shown as magenta dots. (b) Average performance over all videos with different patch sizes.

of the *Stanford Streaming MAR dataset*. The values of $D\text{-}FP_{th}$ and Δ_{max} are chosen to obtain a reasonable number of D-frames that significantly reduces the bit-rate but still maintains a good matching performance. Fig. 3(a) uses *Monsters Inc* as an example test sequence with 8×8 patches. Comparing the performance to the case of fixed $\Delta = 20$, we find that the performance of the adaptive Δ technique better follows the independent detection case. During periods when the use of a fixed Δ causes a large drop in matching (frames 26 to 40), we observe that the adaptive Δ technique solves this problem by using a D-frame at a more suitable time instance and results in a larger number of feature matches.

Fig. 3(b) presents the average matching performance over all the test sequences and compares different patch sizes. The average number of feature matches for adaptive Δ is slightly lower than $\Delta = 20$. This is because we allow a larger upper bound on the adaptive Δ value ($\Delta_{max} = 30$) to further reduce the bit-rate than $\Delta = 20$ case when patch or descriptor encoding is considered. Comparing different patch sizes, we see that reducing the size from 32×32 to 16×16 causes about 2% drop in performance for the uncompressed patches case, while reducing from 32×32 to 8×8 causes a 14% drop. However, using a smaller patch size helps maintaining a better patch quality when applying a bit-rate constraint in the case of transmitting encoded canonical patches.

Note that although adaptive Δ improves the image matching and retrieval performance especially at the time instants when an object of interest is entering or leaving the scene, it adds more computational complexity on the mobile device because the device needs to extract both types of patches (D-Patches and FP-Patches) for each video frame. Therefore, using an adaptive detection interval is optional and depends on the computational resources of the mobile device.

III. Canonical Patch Encoding

We use our own Patch ENCoder (PENC) [17] to encode both FP-Patches and D-Patches. PENC has a similar structure to JPEG [29]. However, PENC applies a pre-processing step of Gaussian blurring and mean removal on the patches and always uses a 2D DCT transform with the same size of the patch in order to avoid producing blocking artifacts within the patch. Fig. 4 summarizes the patch encoding modes. These modes are described in detail in the following subsections.

A. Coding of Forward Propagation Patches

Instead of encoding the FP-Patches independently, we apply predictive coding to lower the bit-rate. From the patch matching stage we know the patch correspondences between consecutive video frames and hence, each patch is predicted from the corresponding patch in the previous frame and only the residual signal between patches is encoded using PENC. In Section II-A, we mention that the original patches are always used in patch matching. However, during the encoding stage, the patches of the current frame are predicted from the reconstructed patches of the previous frame (see Fig. 4(a)) to avoid any drift between the encoder and the decoder. We refer to this encoding mode as *Predictive Coding* or *P* mode.

We observe that the residual energy between the FP-Patches of two consecutive frames is usually less than the residual energy between a patch in a video frame and the matching patch in the database image containing the object of interest. This suggests that in many mobile augmented reality applications, it is not necessary to transmit the prediction residuals in FP-frames. We update the keypoint locations, orientation and scale and still use the descriptors extracted from the patches of the previous D-frame after removing the descriptors corresponding to terminated patches. This has the advantage of significantly lowering the transmission bit-rate with a minor effect on the image matching performance. We refer to this encoding mode as *Skip* or *S* mode as shown in Fig. 4(a).

B. Coding of Detection Patches

The first D-frame is always encoded independently which means that the actual patches are passed to PENC and the bitstream that is generated is decodable without the need of any extra information. We refer to this encoding mode as *Intra Coding* or *I* mode. For the following D-frames, we compare two encoding methods. First, these D-frames can also be encoded independently using *I* mode.

The second mode for encoding patches in D-frames is *Predictive Coding with Patch Selection* or *PS* mode. Although the D-Patches are generated by running the keypoint detector independently and there is no one-to-one correspondence to the patches of the previous frame, we still can search for a patch in the previous reconstructed frames that minimizes some distortion measure (say, SAD). For correct decoding, the residual of the D-Patch is transmitted along with the patch number that is used for prediction. There are two different options in the *PS* mode depending on the reference frame we use for patch selection. In the first option, D-Patches are predicted from the reconstructed FP-Patches of the previous frame. We refer to this mode as PS_P mode where the subscript *P* highlights the use of the *previous* frame for patch selection. The second option involves using the previous D-frame for patch selection. This encoding mode for D-Patches is used in conjunction with *Skip* mode for FP-Patches (Section III-A), because we only use D-Patches for prediction and do not

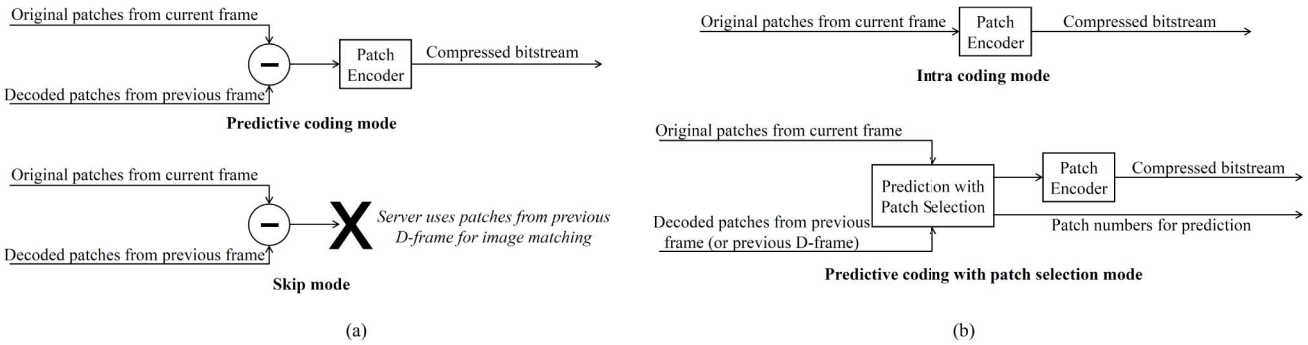


Fig. 4. Canonical patch encoding modes for (a) FP-Patches and (b) D-Patches.

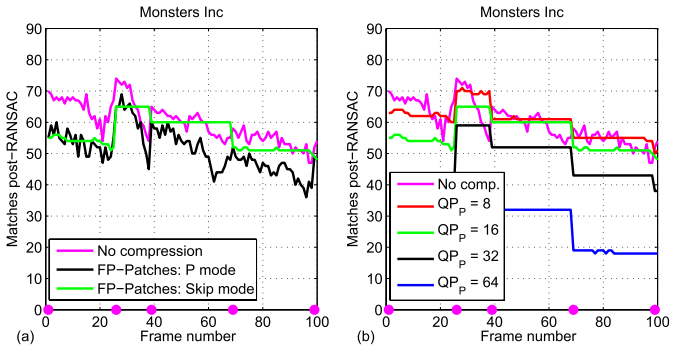


Fig. 5. Matching performance after interframe patch encoding for the test sequence *Monsters Inc*. The figure uses an adaptive Δ and compares the performance of the descriptors extracted from original patches against those extracted from encoded patches. D-Patches are encoded using PS_D mode. (a) compares P mode and *Skip* mode for FP-Patches at the same QP_P and (b) illustrates the effect of varying QP_P on the matching performance.

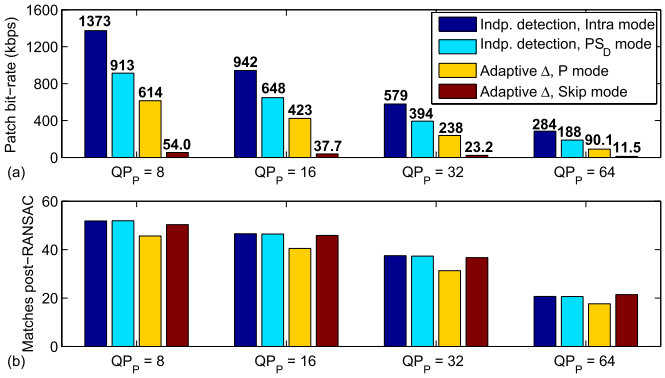


Fig. 6. Patch bit-rates and image matching performance for canonical patch encoding. We vary the value of QP_P and compare independent detection against an adaptive Δ. FP-Patches are encoded using P mode or *Skip* mode. *Skip* mode achieves 9× bit-rate reduction over P mode and 24× reduction over independent detection at a comparable matching performance. (a) Patch bit-rates. (b) Average number of feature matches post-RANSAC.

need the residuals of FP-Patches. We refer to this mode as PS_D mode where the subscript D highlights the use of the previous *D-frame* for patch selection. Fig. 4(b) presents block diagrams for D-Patches encoding modes. Combining an encoding mode for FP-Patches (Fig. 4(a)) and another for D-Patches (Fig. 4(b)), we construct different encoding schemes.

We start from the matching results of the uncompressed patches for the adaptive Δ case and use PENC [17] to compress the patches with the different encoding modes illustrated in Fig. 4. In Fig. 5, we compare different encoding modes and different values of the patch compression quantization parameter QP_P using the example sequence *Monsters Inc*. D-Patches are encoded with PS_D mode since it usually outperforms I mode due to exploiting the temporal correlation in the video. In Fig. 5(a), we fix $QP_P = 16$ and compare encoding the FP-Patches using P mode versus *Skip* mode. The matching performance of the original uncompressed patches is plotted for reference. The image matching performance of *Skip* mode is very close to P mode, where *Skip* mode is characterized by having less fluctuation in the number of matches between consecutive frames since it is using the exact same descriptors over the whole detection interval and only changing the keypoint locations during RANSAC.

Fig. 5(b) applies *Skip* mode for FP-Patches and studies the effect of varying QP_P on the matching accuracy. We observe a consistent drop in the number of feature matches as QP_P increases from $QP_P = 8$ to $QP_P = 64$. Fig. 6 presents the patch encoding bit-rates averaged over the eight test sequences and the average number of feature matches post-RANSAC for different values of QP_P. We compare the adaptive Δ case where FP-Patches are encoded with P mode or *Skip* mode to the independent detection case. For the adaptive Δ case, D-Patches are encoded using PS_D mode. For independent detection, we compare encoding patches using I mode versus PS_D mode, and we observe that PS_D mode results in 30% bit-rate reduction. With temporally coherent patches, *Skip* mode results in a large bit-rate reduction with a matching performance comparable to both P mode and independent detection. We obtain bit-rate savings of about 9× compared to P mode or 24× compared to independent detection with I mode. We observe that $QP_P = 16$ and $QP_P = 32$ achieve a good compromise between the matching accuracy and bit-rate. We target a bit-rate of few tens of kbps, reasonable for practical transmission systems. We achieve acceptable matching results at this bit-rate by using patch termination, small size patches, adaptive Δ with large Δ_{max} and combining PS_D and *Skip* modes.

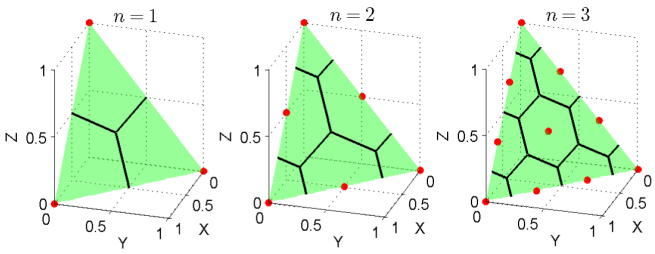


Fig. 7. A_{m-1} lattice for quantizing histograms with the corresponding Voronoi regions ($m = 3$, $n = 1, 2, 3$).

IV. Inter-Descriptor Coding

In this section, we propose encoding techniques that exploit the temporal correlation between feature descriptors calculated from the FP-Patches extracted as described in Section II. These encoding techniques can be applied in the case of the calculation of the feature descriptors at the transmitter [8]. We perform our experiments on the basis of SIFT descriptors due to their state-of-the-art performance for image matching and retrieval applications. In [30], the author shows that omitting the magnitude weighting slightly improves the image matching performance. The resulting descriptor is a Histogram of Gradient (HoG) descriptor with angular gradient bins. In this paper, we use this modified SIFT descriptor, which allows us to build on the elegant lattice coding of histograms [6], [30].

Each histogram can be represented as a point on the probability simplex in m-dimension space where m represents the number of histogram bins ($m = 8$ in case of SIFT). In lattice coding, we define an A_{m-1} lattice over the probability simplex and quantize each histogram to the closest point on the lattice. The total number of lattice points K depends on the number of histogram bins m and the lattice quantization parameter n (see Fig. 7) and is given by the *multiset coefficient*

$$K = \left(\binom{m}{n}\right) = \binom{n+m-1}{m-1} \tag{1}$$

For entropy coding, we enumerate all possible lattice points and use an arithmetic coder to signal the indices of the lattice points corresponding to the histograms of gradients in each spatial bin of the SIFT descriptor. Fig. 7 illustrates the lattice coding idea for $m = 3$ and $n = 1, 2, 3$. For a detailed discussion the reader is referred to [6], [7], and [30].

To exploit the temporal correlation between SIFT descriptors, we consider two different approaches. First, we consider a scheme where we use lattice-quantized SIFT descriptors and we strive to improve entropy coding based on temporal prediction. Second, we consider Differential Pulse Code Modulation (DPCM) style lossy coding of SIFT descriptor differences. These schemes are described in detail in the following subsections.

A. Context-Adaptive Predictive Lattice Coding

We consider a system that applies SIFT keypoint detection and patch matching to detect keypoints in D-frames and FP-frames respectively, then calculates modified SIFT descriptors on the extracted canonical patches, and approximates each histogram of gradients to its closest lattice point. A system, oblivious of the temporal correlation, uses an entropy coder that is optimized for the individual probabilities of the indices of the lattice points (Fig. 7). This is the technique used for still images in [6] and [7]. We refer to this technique as *independent lattice coding* or *intra-descriptor coding*.

1) Full Context Predictive Lattice Coding: Let $\mathbf{a}$ be the lattice point to be encoded in the current FP-frame, and $\mathbf{b}$ be the lattice point of the SIFT descriptor of the corresponding patch in the previous frame. Based on the work in [6] and [7], we can multiply the lattice point components by n and represent $\mathbf{a}$ and $\mathbf{b}$ as vectors of integer numbers as follows.

$$\mathbf{a} = \left[\, a_1 \; a_2 \; \ldots \; a_m \,\right] \tag{2}$$

$$\mathbf{b} = \left[\, b_1 \; b_2 \; \ldots \; b_m \,\right] \tag{3}$$

$$a_i, b_i \epsilon \mathbb{N}, 0 \le a_i, b_i \le n, \sum_{i=1}^{m} a_i = n, \sum_{i=1}^{m} b_i = n \tag{4}$$

We propose to use a context-adaptive predictive lattice coder. We collect statistics for the conditional probabilities $P(\mathbf{a}|\mathbf{b})$ for all possible values of $\mathbf{b}$ through a training phase. We use a context-adaptive entropy coder based on these conditional probabilities to encode the lattice points in FP-frames. Lattice points of D-frames are encoded using *independent lattice coding*. This idea is illustrated in Fig. 8(a).

We observe a large bit-rate reduction due to the use of context-adaptive predictive lattice coding. The only drawback is the need to train and store K probability tables, each containing K entries. Fixing $m = 8$ for SIFT descriptors; our proposed method is practical for only small values of n. For example, $n = 5$ requires training 792 probability tables (1) and we consider this as an acceptable upper bound for the values of n with a reasonable training effort and storage requirement. For larger values of n and for systems which have restrictive memory requirements, we propose another technique for context-adaptive coding that approximates the conditional probabilities and avoids the need of large tables.

2) Distance-Based Context Predictive Lattice Coding: We define d to be the L_1 norm distance between $\mathbf{a}$ and $\mathbf{b}$, thus $d = \|\mathbf{a} - \mathbf{b}\|_1$. From the properties of $\mathbf{a}$ and $\mathbf{b}$ shown in (4), d can only take $n+1$ values $0, 2, 4 \ldots 2n$. Due to the similarity of corresponding descriptors, small values of d are more probable than large values. Also, for small values of d, $\mathbf{a}$ has fewer possibilities if $\mathbf{b}$ is known. We exploit these observations in the design of the distance-based context lattice coding scheme. For example, the lattice in Fig. 9(a) covers all possible vectors $\frac{1}{n}(\mathbf{a} - \mathbf{b})$ with $m = 3$, $n = 2$. Lattice points with the same L_1 value are shown in the same color. There exists 1 point with $d = 0$, 6 points with $d = 2$ and 12 points with $d = 4$.

The decoder has access to the lattice point from previous frame $\mathbf{b}$. Instead of directly encoding $\mathbf{a}$ as in Section IV-A1, we encode the distance d using an arithmetic coder. If $d = 0$, we stop and the decoder reconstructs $\mathbf{a} = \mathbf{b}$. If $d > 0$, we only enumerate the subset of lattice points $\mathcal{A}$ at the specific distance d from the lattice point $\mathbf{b}$, and transmit the index of the lattice point $\mathbf{a}$ from the subset $\mathcal{A}$ using a fixed length code. An encoding example is shown in Fig. 9(b). Our coding

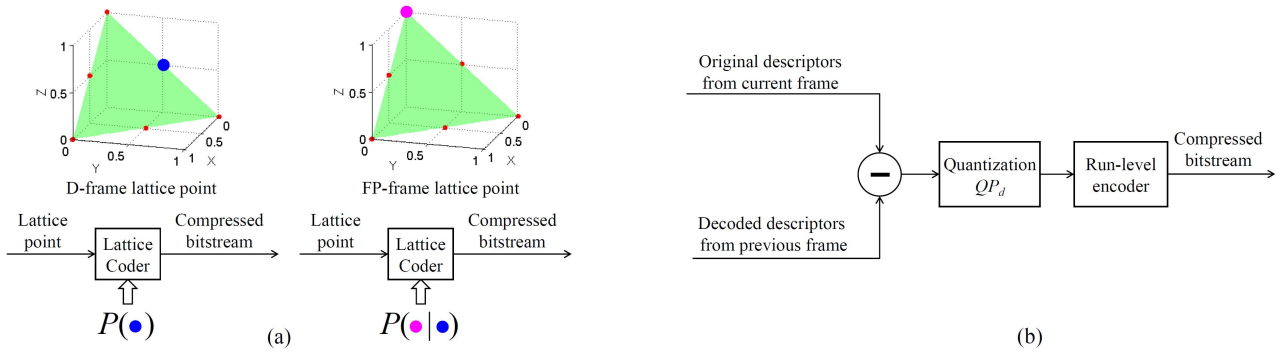


Fig. 8. Inter-descriptor coding. (a) Context-adaptive predictive lattice coding. (b) DPCM-based predictive inter-descriptor coding.

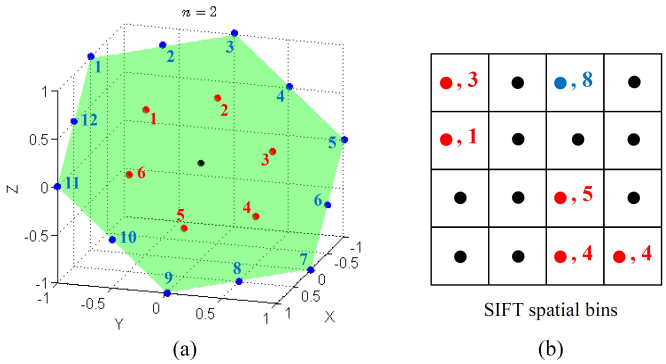


Fig. 9. Distance-based context lattice coding. (a) Lattice covering all possible vectors $\frac{1}{n}(\mathbf{a}-\mathbf{b})$ with $m=3$, $n=2$. Lattice points with the same L_1 value $d=|\mathbf{a}-\mathbf{b}|_1$ are shown in the same color. (b) Encoding example for each SIFT spatial bin. Given $\mathbf{b}$, we signal the distance d and only enumerate the subset of lattice points at this distance to encode $\mathbf{a}$.

scheme is akin to gain-shape vector quantization [31], and can be viewed as an approximation of the full context predictive lattice coding idea presented in Section IV-A1. It requires a slightly higher bit-rate which is usually within 10% of the bit-rate required with full context. It only requires training and saving $n+1$ probability values for the distances d and hence, it is more practical to use with large values of n.

Since we only change the entropy coding technique from *independent lattice coding* to context-adaptive lattice coding, the image matching performance is exactly the same. We can also use *Skip* mode similar to patch encoding (see Section III-B). In *Skip* mode, we update the x and y locations for accurate localization in FP-frames but we use the SIFT descriptors from the last D-frame for image matching.

B. DPCM-Based Predictive Inter-Descriptor Coding

We consider a scheme similar to Section IV-A, but use as input the modified SIFT descriptors in their original unquantized format, so each spatial bin is represented by a histogram of gradients. This allows different quantization schemes for the descriptors in D-frames and FP-frames. Based on the observation that *Skip* mode still achieves good image matching performance, we conjecture that descriptors of FP-frames can be quantized coarsely.

We scale the modified SIFT descriptors, so that the histogram describing each spatial bin consists of 8 values that sum to 255. SIFT descriptors in D-frames are encoded using *independent lattice coding*. We propose to use DPCM to encode FP-frame descriptors. Fig. 8(b) presents the block digram of DPCM-based predictive inter-descriptor coding. The first step is to decode the SIFT descriptors from the previous frame and the descriptors corresponding to terminated patches are removed. We then subtract the decoded descriptors of the previous frame from the original descriptors of the current frame to obtain the descriptor residuals.

Comparing Fig. 9(a) to Fig. 7 ($n=2$ case), we observe that for the same values of m and n, the number of points on the lattice representing $\mathbf{a}-\mathbf{b}$ points is much larger than the number of lattice points on the lattice covering the probability simplex. For larger values of m and n, it becomes impractical to design a lattice quantizer and enumerate lattice points to quantize the descriptor residuals at a sufficient accuracy, due to the large range of the residual values; therefore, we use a scalar quantizer. The descriptor residuals are coarsely quantized using a mid-tread uniform scalar quantizer with a step-size QP_d. Typical values of QP_d are 32, 64, 128 and 256.

The next step is entropy coding of the quantized residuals. We observe that the quantized residuals have long runs of zeros due to the temporal correlation between descriptors and the relatively large values of QP_d. We also observe that the non-zero values are restricted in a small range. For example, if $QP_d=128$, the quantized residuals can only take values $-2,-1,0,1,2$. We design a run-level entropy coder as follows.

We scan the quantized residuals of the whole frame into a contiguous sequence of symbols. Descriptors are scanned in the order imposed by the keypoint location coder [32], [33]. Quantized residuals within a descriptor are scanned in the order of the spatial bins. We convert the sequence of quantized residuals to symbols which concatenate a run of zeros and the following non-zero value. We refer to these symbols as (R,V) symbols. Depending on the value of QP_d, a maximum run R_{max} is defined. We also define a special symbol for the case of R_{max} consecutive zeros and use it to chop a run longer than R_{max} into shorter runs.

An arithmetic coder is used to encode these (R,V) symbols. For example if $QP_d=128$, we define $R_{max}=128$.

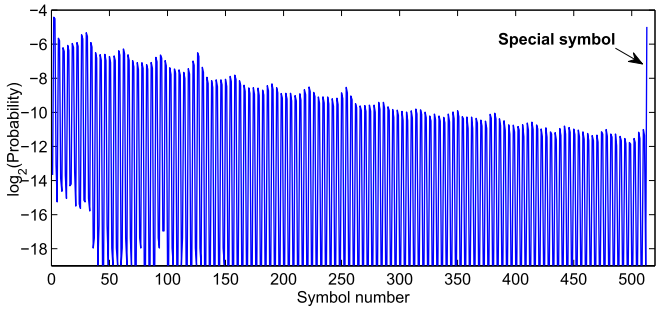


Fig. 10. Symbol probabilities for run-level coding the inter-descriptor residuals. $n = 2$ for D-frames and $QP_d = 128$ for FP-frames.

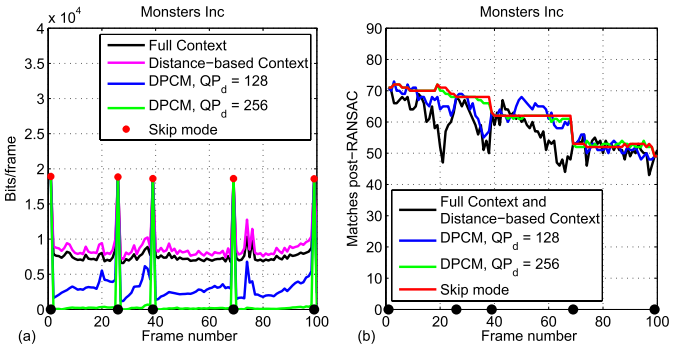


Fig. 11. Descriptor bit-rates and matching performance of inter-descriptor coding techniques for the test sequence *Monsters Inc*. (a) Bit-rate comparison between different encoding methods. (b) Image matching performance for the methods in (a).

Thus, our encoding table consists of 513 symbols: The first 512 symbols are $(0,-2), (0,-1), (0,1), (0,2), (1,-2), \ldots,$ $(127,2)$ and the last symbol indicates 128 consecutive zeros. We train probabilities for this symbol space. These probabilities depend on QP_d and the value of n used for encoding D-frame descriptors. Fig. 10 plots these probabilities for $QP_d = 128$ and $n = 2$. Short runs with small non-zero values are more probable but the probability of the special symbol indicates the existence of very long runs with this coarse quantization.

We run encoding experiments to compare different inter-descriptor coding techniques. Descriptors extracted from D-frames are always encoded using *independent lattice coding*. Fig. 11 presents a detailed comparison of bit-rates and matching performance using the example test sequence *Monsters Inc*. We use $n = 3$ to compress D-frame descriptors. In Fig. 11(a), we plot the number of bits spent on encoding the descriptors of each video frame. Black dots indicate the positions of D-frames, which consume the same number of bits in all curves. The bits spent on encoding D-frames are highlighted with red asterisks, to represent the required descriptor bit-rate for *Skip* mode.

Comparing the bit-rates for full context and distance-based context predictive lattice coding techniques, we observe that full context achieves more efficient entropy coding where the bit-rate gap between the two techniques is about 10%. This gap is reduced with smaller values of n because distance-based context better approximates full context. In DPCM-based

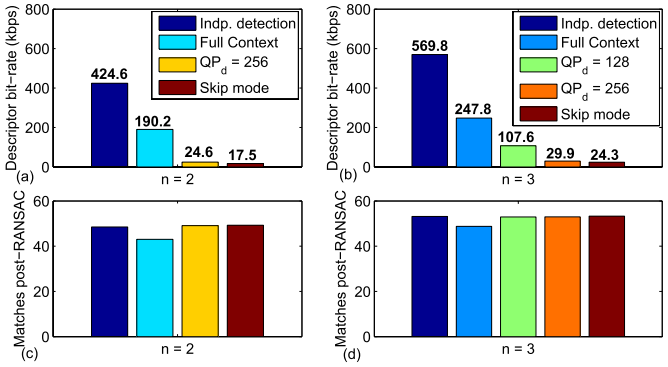


Fig. 12. Descriptor bit-rates and image matching performance for inter-descriptor coding. *Skip* mode achieves 10× bit-rate reduction over full context predictive lattice coding and more than 20× reduction over independent detection at a comparable matching performance. (a) Descriptor bit-rates with $n = 2$ for D-frame descriptors. (b) Descriptor bit-rates with $n = 3$ for D-frame descriptors. (c) Matching performance corresponding to the bit-rates in (a). (d) Matching performance corresponding to the bit-rates in (b).

predictive coding, we have more control of the quantization of the descriptor residuals. We vary the coarsenesss of quantization by varying $QP_d = 128$ and 256. For $n = 3$, these QP_d values ensure that DPCM results in a coarser quantization than the quantization resulting from lattice coding; and thus, $QP_d = 128$ and 256 require less bit-rate to encode the descriptor residuals than context-adaptive predictive lattice coding techniques.

Fig. 11(b) shows the image matching performance for all the encoding methods in Fig. 11(a). Full context and distance-based context predictive lattice coding have the same image matching performance. All the techniques under comparison have a similar matching performance. This agrees with our observations on *Skip* mode in previous experiments and suggests that the descriptor residuals in FP-frames can be either more coarsely quantized or completely ignored.

In Fig. 12, we present the descriptor encoding bit-rates averaged over all eight sequences and the average number of feature matches post-RANSAC for different inter-descriptor coding techniques. We compare four cases: independent detection, full context predictive lattice coding (Section IV-A1), DPCM-based predictive coding (Section IV-B), and *Skip* mode. Figs. 12(a) and 12(b) show the average descriptor coding bit-rate when $n = 2$ and $n = 3$ are used for encoding D-frame descriptors respectively. Figs. 12(c) and 12(d) present the image matching performance for the corresponding experiments in Figs. 12(a) and 12(b). Techniques under comparison have a comparable image matching performance, where $n = 2$ causes a drop of 10% in the average number of feature matches compared to $n = 3$.

For DPCM-based predictive coding to achieve a bit-rate reduction over context-adaptive lattice coding, we quantize the descriptor residuals more coarsely than the quantization resulting from lattice coding. For the values of QP_d that we use, this happens at $QP_d = 256$ when $n = 2$ is used for D-frame descriptors, and at $QP_d = 128$ and $QP_d = 256$ when $n = 3$ is used for D-frame descriptors. DPCM-based predictive coding represents a graceful transition between

context-adaptive predictive lattice coding and *Skip* mode. *Skip* mode achieves 10× bit-rate reduction over full context predictive lattice coding and more than 20× reduction over independent detection at a comparable image matching performance.

We achieve the best results in terms of bit-rate and matching performance when D-frame descriptors are encoded using *independent lattice coding* with $n = 2$ or $n = 3$, and FP-frame descriptors are not transmitted (*Skip* mode). At this configuration, we only require descriptor bit-rates around 20 kbps to achieve good image matching performance. These bit-rates are very promising for streaming MAR applications.

V. Differential Location Coding

In *Skip* mode, we avoid the transmission of the residuals of FP-Patches or SIFT descriptors of FP-frames and use the descriptors extracted from the previous D-frame for image matching. However, in streaming MAR applications, we need to track the object of interest and localize it in all video frames. Hence, we need to update the locations for the keypoints detected in the FP-frames. Some applications may also need the transmission of the orientation and scale of these keypoints.

For the keypoint locations in D-frames, we use the independent location coder developed by Tsai et al. [32], [33]. Our applications usually consider a rigid body and the update in the keypoint locations between consecutive video frames can be approximated by an affine transform. For FP-frames, we propose to predict the original keypoint locations of the current frame by applying an affine transform on the decoded locations of the previous frame, and then we encode the location differences. Let λ_C represent the original locations of the current frame and $\lambda_{P,dec}$ represent the decoded locations of the previous frame. We assume that λ_C and $\lambda_{P,dec}$ are related by an affine transform $\mathbf{T}$ so that, $\lambda_C = \mathbf{T}\lambda_{P,dec}$, or in a more detailed homogeneous-coordinate notation

$$\begin{bmatrix} x_C \\ y_C \\ 1 \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{P,dec} \\ y_{P,dec} \\ 1 \end{bmatrix}. \quad (5)$$

We form an over-determined system of equations relating points in λ_C and $\lambda_{P,dec}$ and calculate $\mathbf{T}$ as the least squares solution of these equations. We quantize the six affine parameters in $\mathbf{T}$ and signal them to the server side. Since there is a relatively small variation between consecutive video frames, the parameters t_{11} and t_{22} are usually close to 1 and the parameters t_{12} and t_{21} are usually close to 0. The variation in the shift parameters t_{13} and t_{23} depends on the frame dimensions and object speed. We use different uniform quantizers for different affine parameters. We observe that using 7-bit quantizers provides a reasonable accuracy for signaling these parameters. For t_{11} and t_{22}, we have 127 quantization levels between 0.9 and 1.1. For t_{12} and t_{21}, we have 127 levels between -0.1 and 0.1. Finally, for t_{13} and t_{23}, we have 127 levels between -63 and 63 (for VGA resolution video frames). We refer to the quantized version of the least squares solution of the affine parameters as $\mathbf{T}_q$.

We then calculate λ_A, the affine transformed version of $\lambda_{P,dec}$, so $\lambda_A = \mathbf{T}_q\lambda_{P,dec}$. The location differences

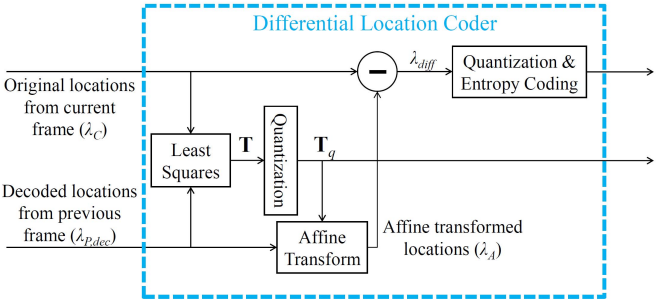


Fig. 13. Block diagram of differential location coding.

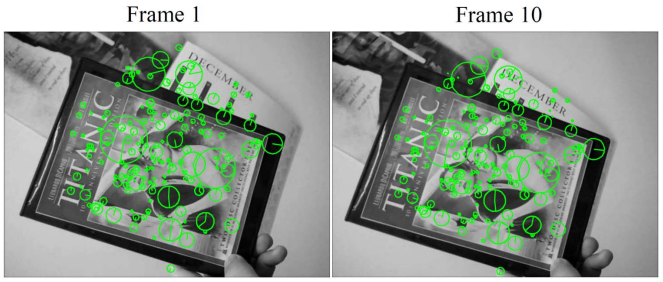


Fig. 14. Example affine location update for a moving object recorded with a moving camera. Location update keeps following the correct interest point locations for the DVD, while the correct locations deviate from the word "DECEMBER" in the background.

$\lambda_{diff} = \lambda_C - \lambda_A$ are quantized using a quantization step Q_{loc} and encoded using an arithmetic coder. The encoded locations need to be decoded back at the transmitter side in order to be used in the predictive coding of the keypoint locations of the following FP-frame. Fig. 13 shows a block diagram of the proposed differential location coder.

To further lower the location bit-rate, we propose the *Skip with Affine update* or *SA* mode for encoding keypoint locations. In this mode, we do not encode the location differences λ_{diff} and only transmit the affine parameters $\mathbf{T}_q$. The receiver uses λ_A as the decoded values of the keypoint locations. Using 7-bit quantizers, each FP-frame only needs 42 bits to signal $\mathbf{T}_q$. We observe that *SA* mode introduces a large bit-rate reduction at a comparable image matching performance.

A global affine location update should be sufficient for rigid planar objects especially if the object is static and the camera is moving. In this case, the background clutter does not affect the accuracy of location update, since there is no relative motion between the object and the background. To study the accuracy of the affine location update, we collect videos for a moving object recorded with a moving camera in the *Stanford Streaming MAR dataset* [26]. In these videos, the interest points in the background do not conform with the affine transform that best describes the object motion. If the number of interest points on the background is small, they have a negligible effect on the least squares solution $\mathbf{T}$.

In Fig. 14, we present example frames 1 and 10 from the test sequence *Titanic Moving*. These frames use our temporally coherent keypoint detector for interest point detection, and they belong to the same detection interval. The figure

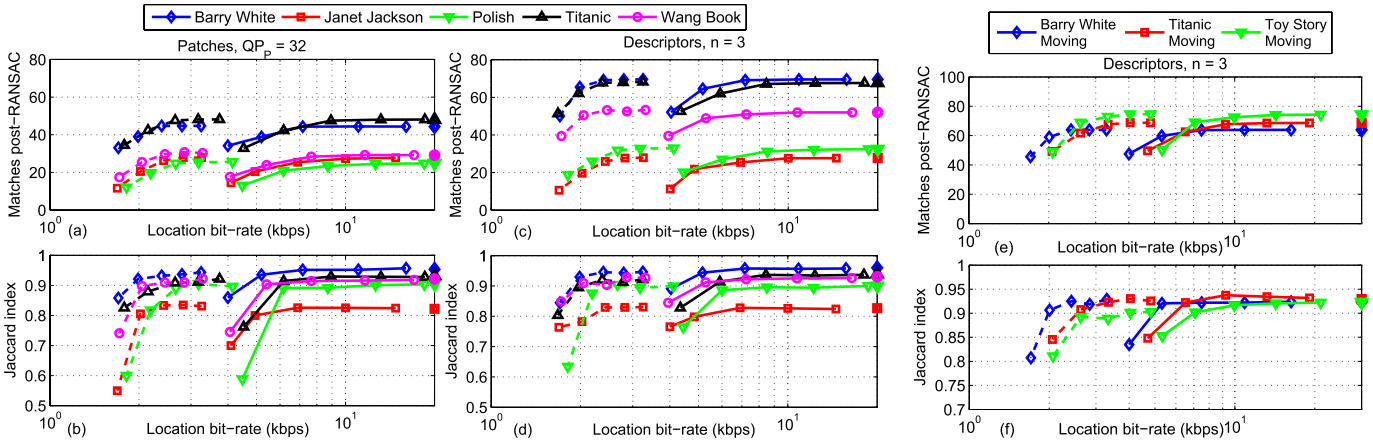


Fig. 15. Image matching performance and localization accuracy after location coding. $Q_{loc} = 2, 4, 8, 16$ and 32 values are used, and good performance is observed up to $Q_{loc} = 8$. (a) and (b) report the number of feature matches post-RANSAC and the Jaccard index for patches encoded at $QP_P = 32$ and (c) and (d) report the same results for descriptors encoded at $n = 3$. (e) and (f) are the results for video sequences with relative motion between the object of interest and the background.

shows the keypoint locations on the server side after decoding the affine transform location updates. We observe that the keypoint locations on the DVD keep following the correct locations detected in frame 1. However, the correct locations on the background deviate with time. For example, there are two keypoints detected on the letters **B** and **R** in the word "DECEMBER" in the background in frame 1. We observe the deviation in these two keypoint locations in frame 10. Least squares solution is sufficient in this example due to the small number of keypoints from the background. Note that in applications with large background clutter, the encoder can estimate the affine transform **T** using more accurate methods at the expense of more computational complexity. For example, the encoder can apply RANSAC [27] between λ_C and $\lambda_{P,dec}$ and remove the outliers from background clutter.

We end this section by reporting experiments to measure the effect of location quantization on the image matching performance. We investigate two different metrics: the number of feature matches post-RANSAC and the localization accuracy. We manually generate the ground-truth object location by defining a bounding quadrilateral around the object of interest in each video frame. From RANSAC results, we project the corners of the object in the database image on each video frame. The localization accuracy is measured by calculating the *Jaccard* index. The Jaccard index is defined as the ratio between the area of intersection between the ground-truth and the projected quadrilaterals, and the area of their union. Jaccard index values closer to 1 indicate better localization.

The results reported in the previous sections use unquantized locations. We quantize and encode locations with our proposed differential location coder and also with the *SA* mode. We run the location coder along with the proposed canonical patch encoder (Section III) or with our inter-descriptor encoder (Section IV), using the encoding modes that achieve the best results for the unquantized locations case. For canonical patches, PS_D mode is used with D-Patches and *Skip* mode is used with FP-Patches. We present results at a fixed patch encoding $QP_P = 32$. For feature descriptors, we also use *Skip* mode for the descriptors extracted from FP-frames and report example results using $n = 3$ for encoding D-frame descriptors.

In Figs. 15(a) and 15(b), we study the effect of encoding keypoint locations on a system that streams encoded canonical patches. In Figs. 15(c) and 15(d), we repeat the experiment for a system that streams encoded descriptors. Figs. 15(a) and 15(c) present the relation between the location bit-rates and the average number of feature matches post-RANSAC for five test sequences: *Barry White*, *Janet Jackson*, *Polish*, *Titanic* and *Wang Book*. The figures compare the results of the differential location coder (solid lines) to the results of *SA* location coding mode (dashed lines). For the differential location coder, we use the same Q_{loc} for both D-frames and FP-frames. We vary the location bit-rates by varying $Q_{loc} = 2, 4, 8, 16$ and 32. $Q_{loc} = 8$ results in good image matching accuracy at an acceptable location bit-rate. For comparison, the figures state the average number of matches in case of using unquantized locations using a separate marker on the right edge. We observe a negligible drop due to location quantization up to $Q_{loc} = 8$.

In Figs. 15(b) and 15(d), we compare the Jaccard index at different location bit-rates for the same video sequences and the same encoding parameters in Figs. 15(a) and 15(c). The Jaccard index is averaged over all the frames in each video sequence. Again, the average Jaccard index using unquantized locations is plotted using a separate marker on the right edge. The localization accuracy drops with coarser quantization of keypoint locations, and $Q_{loc} = 8$ results in acceptable compromise between bit-rate and accuracy. We observe that a larger number of feature matches usually results in a better Jaccard index because of obtaining more accurate geometric transformations from RANSAC. The *SA* location coding mode introduces a large location bit-rate reduction of about $3\times$ when compared to our differential location coder at a comparable performance in terms of image matching and object localization accuracy.

To study the effect of a relative motion between the object of interest and the background, we repeat the same experiments

in Figs. 15(c) and 15(d) on different test sequences containing a moving object and recorded with a moving camera [26]. These sequences are *Barry White Moving*, *Titanic Moving* and *Toy Story Moving* and the results are presented in Figs. 15(e) and 15(f). Running the temporally coherent detector on these sequences, it results in 4, 7 and 7 D-frames respectively. Similar performance and bit-rate reduction are observed for these sequences, and we conclude that our location coding techniques are robust to moderate background clutter.

Note that at $Q_{loc} = 8$, we achieve accurate localization results at location bit-rates of about $7 - 10$ kbps with our proposed differential location coder and about $2 - 3.5$ kbps with *Skip with Affine update* or *SA* mode. On average, encoding a keypoint location in a D-frame requires 5 bits/location, and in an FP-frame requires 1.2 bits/location. This indicates that differential encoding of keypoint locations results in around $4\times$ bit-rate reduction when compared to non-differential methods [33]. Moreover, using *SA* mode, the location bit-rate from FP-frames is negligible and location coding bits are mainly spent on the keypoint locations from D-frames.

VI. Overall Performance

An alternative to the proposed keypoint detection and predictive coding techniques is to stream the whole video to the MAR server and perform keypoint detection, feature extraction and image matching on the decoded frames on the server side. We refer to this method as the *Send Video* scheme. We implement the *Send Video* scheme and compare its image matching performance to our proposed schemes. Videos are encoded using H.264 standard [23] using the following parameters: IPPP... structure, intra-period $=$ 50 frames, $QP_{Pframes} = \{38, 42, 46, 50\}$ and $QP_{Iframes} = QP_{Pframes} - 3$. This encoding structure is better suited for mobile applications because of its low complexity and low latency. We run SIFT keypoint detector and calculate modified SIFT descriptors on the decoded videos. No further compression is applied on the extracted patches, SIFT descriptors or keypoint locations.

Fig. 16 uses the eight sequences from Fig. 2, and represents an overall comparison between the proposed architectures *Send Patches* and *Send Descriptors* versus *Send Video*. We always utilize the encoding modes that achieve the best image matching performance at the lowest streaming bit-rate. In *Send Patches*, we use PS_D mode for D-Patches with $QP_P = \{8, 16, 32, 64\}$, *Skip* mode for FP-Patches, and *SA* mode with $Q_{loc} = 8$ for location coding. The reported bit-rate represents the sum of bit-rates of canonical patches and keypoint locations. In *Send Descriptors*, we use *independent lattice coding* to encode D-frame descriptors with $n = \{1, 2, 3, 4\}$. We use *Skip* mode for FP-frames, and *SA* mode with $Q_{loc} = 8$ for location coding. Similarly, the reported bit-rate represents the sum of bit-rates of encoded descriptors and keypoint locations. In *Send Video*, we vary the bit-rate by varying H.264 encoding parameters as mentioned above.

At bit-rates below 100 kbps for *Send Video*, the keypoint detector fails to detect good interest points due to the excessive blur and compression artifacts in the decoded videos. Hence, the image matching performance drops significantly and our proposed architectures achieve a large improvement over the *Send Video* architecture. At the same average number of feature matches, *Send Patches* achieves $4\times$ bit-rate reduction and *Send Descriptors* achieves $10\times$ bit-rate reduction over the *Send Video* architecture. The performance of *Send Patches* saturates at a lower value compared to other architectures due to the loss resulting from using smaller size 8×8 patches to lower the bit-rate as illustrated in Fig. 3(b). Using *Send Descriptors*, we can achieve efficient streaming mobile augmented reality at bit-rates around $20 - 30$ kbps. These bit-rates are practical for today's wireless links and much better than streaming the whole video.

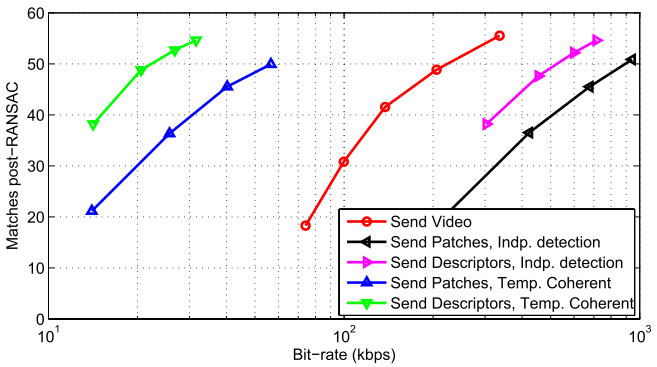


Fig. 16. Comparison between different streaming MAR architectures. *Send Patches* and *Send Descriptors* achieve around $4\times$ and $10\times$ bit-rate reduction respectively, when compared to *Send Video* architecture at the same average number of feature matches. When compared to independent detection of keypoints frame-by-frame, *Send Patches* and *Send Descriptors* with a temporally coherent detector achieve $15\times$ and $20\times$ bit-rate reduction respectively.

We also compare our proposed systems to the case of independent detection. We implement *Send Patches* and *Send Descriptors* which run the keypoint detector frame-by-frame (all video frames are D-frames) and encode canonical patches, feature descriptors and keypoint locations with the same D-frame encoding parameters used above. No *Skip* mode is used due to the absence of FP-frames. We present the results in Fig. 16. Using a temporally coherent keypoint detector, we observe a very large bit-rate reduction of $15\times$ in the case of *Send Patches* and up to $20\times$ in the case of *Send Descriptors*, compared to the independent detection of keypoints, at the same average number of feature matches.

Note that the *Send Video* architecture outperforms *Send Patches* and *Send Descriptors* if these architectures use independent detection (i.e., a temporally non-coherent keypoint detector). This is because video coding standards like H.264 achieve excellent compression by exploiting the temporal redundancy in the video. This comparison emphasizes the importance of our proposed temporally coherent keypoint detector as a key ingredient towards exploiting the same redundancy for canonical patches and feature descriptors.

VII. Image Retrieval

A. Experimental Results

Previous experiments focus on the pairwise matching performance between the query video frames and the clean database image. The number of feature matches

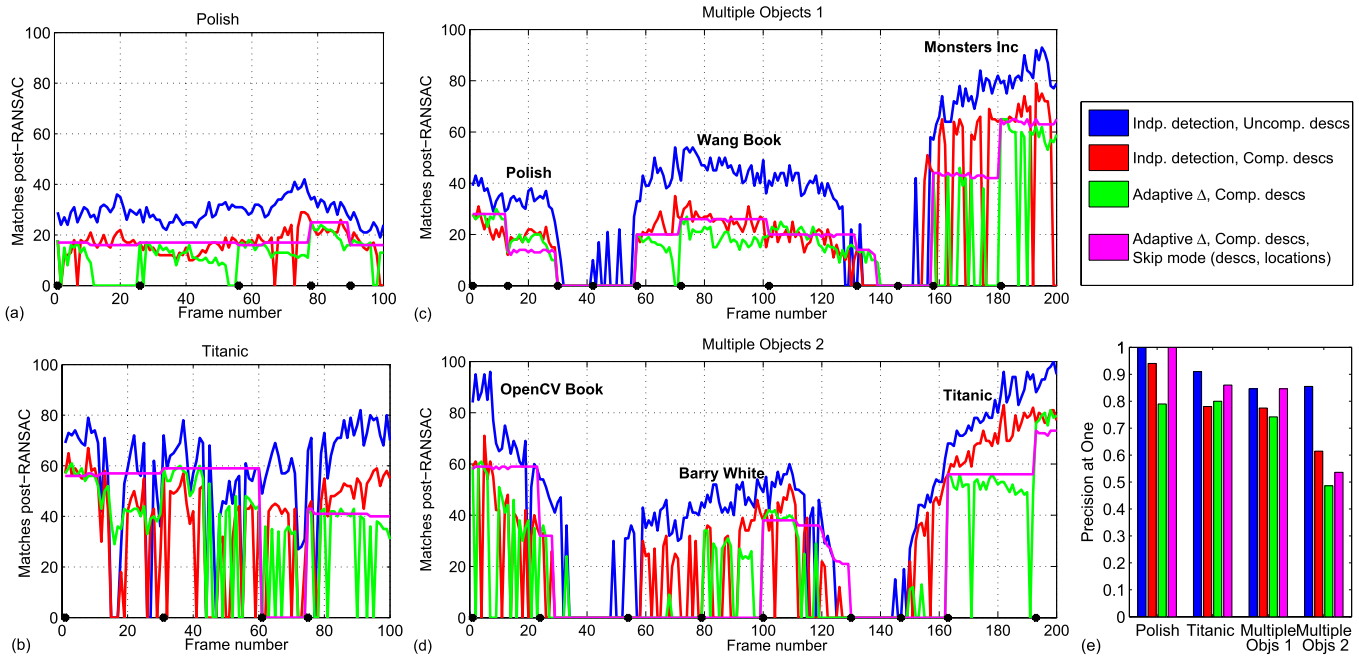


Fig. 17. Image retrieval results. The figure compares running retrieval every frame, with descriptors representing four different cases of keypoint detection and feature compression. (a) *Polish* sequence, (b) *Titanic* sequence, (c) *Multiple Objects 1* sequence, (d) *Multiple Objects 2* sequence, and (e) Precision at One for the video sequences in (a), (b), (c) and (d).

post-RANSAC and the localization accuracy both help to study the effect of using our temporally coherent keypoint detector versus the independent detection of keypoints. These results also evaluate the importance of encoding the residuals of the canonical patches, feature descriptors and keypoint locations of FP-frames. In this section, we run image retrieval experiments against a large database of nearly 1 million images. We assume that the streaming MAR server runs retrieval for every video frame. This may seem redundant because the retrieval results should be highly correlated for consecutive video frames. However, this experiment is important to study the sensitivity of our system to an object leaving, or a new object entering the scene. Therefore, we run the retrieval experiment on videos with multiple objects from the *Stanford Streaming MAR dataset* [26].

Our database for image retrieval experiments consists of the distractor images from MPEG CDVS evaluation framework [15], plus 23 clean images, without background clutter, of the objects of interest. We train a Scalable Vocabulary Tree (SVT) [34] with branch factor $C = 10$ and number of levels $L = 6$ using about 7.5 million uncompressed descriptors from the training set MIRFLICKR-25000 database [35]. For each image in the database, we extract up to 300 uncompressed modified SIFT descriptors, and classify them along the tree. Image retrieval is performed for every video frame using the conventional SVT voting scheme [8]. From the SVT results, we obtain a shortlist of the candidate matching images. We run RANSAC on the top 500 images in the shortlist. To declare a matching image, we set the minimum number of post-RANSAC matches to 6.

In Fig. 17, we plot the number of post-RANSAC matches against the frame number of each video sequence only if our system retrieves the correct image. A value of zero in these plots indicates either no matching image or a wrong matching image is found. The figure compares four different configurations. First, the independent detection of keypoints frame-by-frame with uncompressed modified SIFT descriptors. Second, the independent detection of keypoints with lattice-coded descriptors ($n = 3$). Third, keypoints detected with our temporally coherent detector (adaptive Δ) with lattice-coded descriptors in both D-frames and FP-frames. Finally, temporally coherent detection, with lattice-coded descriptors in D-frames, *Skip* mode for descriptors in FP-frames, and *SA* mode for keypoint locations. D-frames are marked with black dots on each plot.

Figs. 17(a) and 17(b) present results for videos with a single object, *Polish* and *Titanic*. Comparing the first two curves, we observe that descriptor compression causes a drop in the number of matches post-RANSAC. The third curve with temporally coherent detection has a larger drop in performance. The performance in each detection interval depends a lot on its D-frame. A D-frame with good retrieval performance results in FP-frames with good retrieval performance and vice versa. Finally, the last curve with no residuals for descriptors and keypoint locations shows that the retrieval performance is entirely controlled by the performance of the D-frames. If no match is found for a D-frame, then the whole detection interval suffers a no-match (frames 61 to 75 in *Titanic*). If encoding residuals is harming the performance, *Skip* mode for descriptors and locations prevents the performance drop, since it keeps the retrieval result of the D-frame (frames 11 to 25 in *Polish*). We conclude that using *Skip* mode for descriptors and locations achieves good retrieval results but puts more emphasis on the retrieval performance of the D-frames.

Figs. 17(c) and 17(d) present results for videos with multiple objects, *Multiple Objects 1* and *Multiple Objects 2* respectively. We are interested in the response of the temporally coherent detector, when the object of interest enters or leaves the scene. In the case of independent detection of keypoints, the server detects the entrance of a new object before the case of temporally coherent keypoints. This is due to the propagation of patches that do not represent the new object from previous frames. The new object is recognized at the first D-frame with a correct retrieval result after the entrance of the object. The entrance of the new object should lead the temporally coherent detector to insert a D-frame. However, this does not guarantee that this D-frame has the correct retrieval result. In general, with our temporally coherent detector, the detection of the new object happens within few frames from the actual entrance of the object. In only one case, we observe bad retrieval performance with the detection of *Barry White* CD in Fig. 17(d) due to the excessive glare on the CD.

Finally, in Fig. 17(e), we plot the Precision at One (PAO) for all video sequences. PAO is defined as the ratio of query images correctly retrieved at the top rank position from the retrieval system. Comparing the first two values for each sequence, we observe a drop in PAO due to the compression of feature descriptors for the independent detection case. When applying temporally coherent detection (third value for each sequence), we observe a slight drop in PAO. This is because the temporally coherent detector propagates canonical patches with poor retrieval performance. The last value for each sequence shows an improvement when we use *Skip* mode, when compared to sending the descriptor residuals. Performance of *Skip* mode totally depends on the D-frame at the start of each detection interval.

B. Server-Based Tracking

In a practical streaming MAR system, we do not need to run image retrieval every frame because the retrieval decisions are highly correlated due to the temporal correlation in the video. To exploit this temporal correlation, we define two states for the image recognition operations on the streaming MAR server: *Retrieval State* and *Pairwise State*. In *Retrieval State*, the server compares the descriptors from the query video frame to the descriptors of all database images using an SVT, and obtains a shortlist of candidate images. The server then performs pairwise matching for this shortlist, and either finds a matching image or goes over all candidate images and declares a no match. In *Pairwise State*, there is always a database image that matches the previous query video frame. The server performs pairwise matching against this database image only.

Initially, the system is in *Retrieval State*, and the streaming MAR server runs image retrieval against the whole database for every video frame until an object is recognized. After an object is detected, the system switches to *Pairwise State* and the server performs pairwise matching against the database image of the recognized object from the previous frame. If this pairwise image matching operation fails to find the object in the current frame, this may be due to motion blur, occlusion, or the current object is leaving the scene. The system switches back to *Retrieval State*, until a new object is recognized in the video. We refer to this technique as *Server-Based Tracking* since the server tracks an object of interest to avoid running the whole retrieval pipeline for every video frame. Fig. 18 illustrates the state diagram for *Server-Based Tracking*.

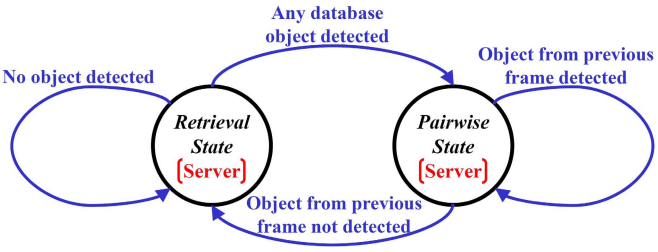


Fig. 18. State diagram for *Server-Based Tracking*.

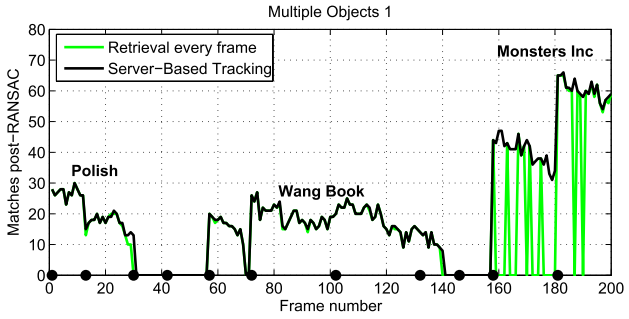


Fig. 19. Image retrieval results for *Multiple Objects 1* video sequence. The figure compares performing retrieval every frame to using the *Server-Based Tracking* mechanism. The figure studies the case of streaming temporally coherent feature descriptors, compressed with lattice coding at $n = 3$.

In Fig. 19, we start from the retrieval results for every video frame (green curve in Fig. 17(c)), and add the retrieval result obtained by applying our technique for *Server-Based Tracking*, for the case of streaming temporally coherent feature descriptors, compressed using lattice coding with $n = 3$. For the first object (*Polish*), both techniques have nearly the same retrieval performance, and always manage to retrieve the correct object. For the second object (*Wang Book*), both techniques fail to retrieve the correct object at frames 70 and 71. This is due to the failure in the pairwise matching although the correct object appears in the shortlist of 500 images, when retrieval is performed every frame. Finally, when we study the third object (*Monsters Inc*), we find that *Server-Based Tracking* results in a large improvement in retrieval performance. Encoded descriptors may fail to retrieve the correct object to the shortlist, yet these descriptors are good enough for pairwise matching. Hence, *Server-Based Tracking* technique has two advantages. First, it reduces the computational complexity of the retrieval process on the MAR server by exploiting the temporal correlation in the video sequence. Second, it improves the image retrieval performance in the case of SVT retrieval failure with successful pairwise matching.

C. On-Device Tracking

In the previous section, we assumed that both image retrieval and object tracking are performed on the MAR server.

TABLE I
COMPARISON BETWEEN THE REQUIRED BIT-RATES AND THE PAO FOR *Server-Based Tracking* AND *On-Device Tracking*

	Server-Based Tracking			*On-Device Tracking*				
Video sequence	Number of D-frames	Uplink bit-rate (kbps)	PAO	Number of D-frames sent	Uplink bit-rate (kbps)	Number of objects	Downlink bit-rate (kbps)	PAO
Polish	5	31.22	1.0	1	5.92	1	6.17	1.0
Titanic	4	25.24	1.0	1	5.98	1	6.08	1.0
Multiple Objects 1	11	35.11	0.87	5	15.19	3	9.33	0.85
Multiple Objects 2	9	28.34	0.57	7	20.80	3	9.14	0.56

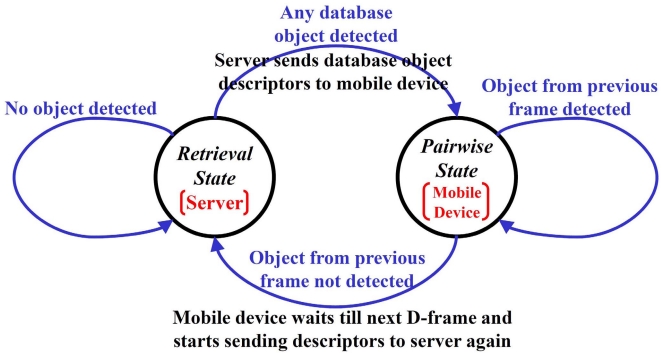


Fig. 20. State diagram for *On-Device Tracking*.

For many MAR applications, it is possible to perform object tracking on the mobile device. Fig. 20 illustrates the idea of *On-Device Tracking*. The mobile device runs our proposed temporally coherent keypoint detector, and sends the encoded descriptors to the MAR server. Image retrieval against a large database is performed on the server; hence, the system is in *Retrieval State*. Once an object is recognized in a video frame, the server sends a compressed version of the descriptors and keypoint locations of the database image containing the object of interest to the mobile device.

For the next video frames, the mobile device stops sending feature descriptors to the server and locally performs two operations. First, it performs pairwise matching between the compressed feature descriptors of the database image and the original feature descriptors extracted from the video frames following the detection of the database object. Therefore, in *Pairwise State*, the mobile device checks whether the object is still in the scene or not. Second, the mobile device performs object tracking using the uncompressed keypoint locations of subsequent video frames.

When the pairwise matching fails on the mobile device, the device waits till the next D-frame to avoid the excessive insertion of independently detected feature descriptors, especially in periods with either no object of interest or an object of interest that is difficult to be recognized. The mobile device then starts sending the compressed feature descriptors back to the server. The server performs image retrieval (*Retrieval State*) again till it detects an object in the video. Hence, in *On-Device Tracking*, image retrieval in the *Retrieval State* is performed on the server, while pairwise matching in the *Pairwise State* is performed on the mobile device. If the new detected object is different from the previously detected object, the server sends the compressed database descriptors of the new object to the mobile device, and so on. The downlink bit-rate is the sum of the bit-rates needed to stream all the database objects that are detected in the whole video sequence. When *Skip* mode is used for encoding descriptors, the uplink bit-rate is the sum of the bit-rates of the D-frames sent when pairwise matching fails on the mobile device.

We start with the most bit-rate efficient image retrieval results shown in Fig. 17. These results use the temporally coherent detector with *Skip* mode for descriptors and locations, and are represented by the magenta curves in the figure. We compare a system that applies *Server-Based Tracking* to a system that performs *On-Device Tracking*, and present the results in Table I. Both systems have comparable PAO values. When using *Server-Based Tracking*, the bit-rate mainly depends on the number of D-frames in the video sequence. For the video sequences *Polish* and *Titanic*, there is a single object present in the entire sequence. Hence, for *On-Device Tracking*, the uplink bit-rate is the bit-rate for sending the descriptors of the first frame to the server, and the downlink bit-rate is the bit-rate for sending the descriptors of the database object to the mobile device. There is no further communication with the server as long as the pairwise matching operation is successful on the mobile device.

For the sequences *Multiple Objects 1* and *Multiple Objects 2*, the uplink bit-rate is the sum of the bit-rates of the D-frames sent from the mobile device when pairwise matching is not detecting an object. Since *Multiple Objects 2* has poor image matching and retrieval performance (see Fig. 17(d)), 7D-frames out of the total 9 D-frames are sent to the server. The downlink bit-rate is the bit-rate for sending the descriptors of all the detected objects to the mobile device. Compared to videos with a single object, these videos require a higher uplink bit-rate because of having periods with no objects of interest, where the mobile device still needs to send D-frames. These videos also require a higher downlink bit-rate because there are more objects of interest than videos with a single object, if we compare similar periods of time. Hence, *On-Device Tracking* can achieve bit-rate reduction compared to *Server-Based Tracking* when objects of interest are present for long periods of time in the video, and when most frames contain an object of interest.

VIII. CONCLUSIONS

We present an efficient method for obtaining canonical image patches that are temporally coherent to exploit the temporal correlation in videos used in streaming mobile augmented reality applications and achieve accurate content-based

retrieval and localization at a low bit-rate. We also propose methods for encoding these canonical patches and their corresponding feature descriptors and keypoint locations using efficient predictive coding techniques. Finally, we present two techniques that exploit the temporal correlation in the video to reduce the complexity of the image retrieval process, by only performing pairwise matching with the correct database object, either on the server or on the mobile device.

We compare the performance with temporally coherent keypoint detection against the independent detection of keypoints frame-by-frame and also to streaming the whole video to the server. Experimental results show that the proposed temporally coherent detection mechanism results in an image matching and retrieval performance comparable to the oblivious detection of new keypoints every video frame.

Both our interframe canonical patch encoder, and inter-descriptor encoder achieve more than 20× bit-rate reduction compared to sending independently detected canonical patches or feature descriptors every frame. Differential location coding results in 4× bit-rate reduction over independent coding of keypoint locations, and we obtain another 3× reduction in location bit-rate if we skip encoding location residuals.

When we compare performing the pairwise matching process on the server (*Server-Based Tracking*) versus on the mobile device (*On-Device Tracking*), we deduce that *On-Device Tracking* can further reduce the required streaming bit-rate if the mobile device can handle the added complexity of pairwise matching. The overall system can achieve efficient streaming mobile augmented reality at bit-rates of about 20–30 kbps, practical for today's wireless links and less than one-tenth of the bit-rate needed to stream the whole video.

Streaming MAR systems should perform efficient real-time object recognition and tracking at low bit-rates. Our paper provides powerful tools for the low bit-rate aspect of the system. A lot of work in computer vision literature designs fast features for object recognition and tracking. A natural extension of our research is to combine the techniques for low bit-rate streaming MAR with these fast features to achieve the benefits of both. For example, D-frames can use Rotation Invariant Fast Features or RIFF keypoint detector developed in [36] and [37] for speeding-up the system. Performing inter-descriptor compression for temporally coherent RIFF features is another idea. Moreover, it is interesting to study the ability to perform patch matching with fewer/no pixel interpolations.

REFERENCES

[1] (2013). *Google Glass* [Online]. Available: http://www.google.com/glass/start/

[2] (2011). *Amazon Flow* [Online]. Available: http://a9.amazon.com/-/company/flow.jsp

[3] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[4] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: Speeded up robust features," in *Proc. Eur. Conf.. Comput. Vis.*, Graz, Austria, May 2006.

[5] V. Chandrasekhar, G. Takacs, D. Chen, S. Tsai, R. Grzeszczuk, and B. Girod, "CHoG: Compressed histogram of gradients—A low bit-rate feature descriptor," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Florida, FL, USA, Jun. 2009, pp. 2504–2511.

[6] V. Chandrasekhar *et al.*, "Quantization schemes for low bitrate compressed histogram of gradients descriptors," in *Proc. IEEE Comput. Soc. Conf. Comp. Vis. Pattern Recognit. Workshops*, San Francisco, CA, USA, Jun. 2010.

[7] V. Chandrasekhar *et al.*, "Compressed histogram of gradients: A low-bitrate descriptor," *Int. J. Comput. Vis.*, vol. 94, no. 5, pp. 348–399, May 2011.

[8] B. Girod *et al.*, "Mobile visual search," *IEEE Signal Process. Mag.*, vol. 28, no. 4, pp. 61–76, Jul. 2011.

[9] V. Chandrasekhar *et al.*, "Survey of SIFT compression schemes," in *Proc. 2nd Int. Workshop Mobile Multimedia Process. (WMMP)*, Istanbul, Turkey, Aug. 2010.

[10] B. Girod, V. Chandrasekhar, R. Grzeszczuk, and Y. A. Reznik, "Mobile visual search: Architectures, technologies, and the emerging MPEG standard," *IEEE Multimedia*, vol. 18, no. 3, pp. 86–94, Mar. 2011.

[11] G. Francini, S. Lepsoy, and M. Balestri, *Description of Test Model Under Consideration for CDVS*, document ISO/IEC JTC1/SC29/WG11/N12367, Dec. 2011.

[12] V. Chandrasekhar *et al.*, *Improvements to the Test Model Under Consideration with Low Memory Descriptors*, document ISO/IEC JTC1/SC29/WG11/M23580, Feb. 2012.

[13] S. Paschalakis *et al.*, *CDVS CE2: Local Descriptor Compression Proposal*, document ISO/IEC JTC1/SC29/WG11/M25929, Jul. 2012.

[14] J. Lin *et al.*, *Peking University Response to CE 1: Performance Improvements of the Scalable Compressed Fisher Codes (SCFV)*, document ISO/IEC JTC1/SC29/WG11/M28061, Jan. 2013.

[15] Y. Reznik, G. Cordara, and M. Bober, *Evaluation Framework for Compact Descriptors for Visual Search*, document ISO/IEC JTC1/SC29/WG11/N12202, Jul. 2011.

[16] M. Makar, C.-L. Chang, D. Chen, S. S. Tsai, and B. Girod, "Compression of image patches for local feature extraction," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Taipei, Taiwan, Apr. 2009, pp. 821–824.

[17] M. Makar, H. Lakshman, V. Chandrasekhar, and B. Girod, "Gradient preserving quantization," in *Proc. 19th IEEE Int. Conf. Image Process.*, Orlando, FL, USA, Sep./Oct. 2012, pp. 2505–2508.

[18] M. Makar, S. S. Tsai, V. Chandrasekhar, D. M. Chen, and B. Girod, "Interframe coding of canonical patches for mobile augmented reality," in *Proc. IEEE Int. Symp. Multimedia*, Irvine, CA, USA, Dec. 2012.

[19] M. Makar, S. S. Tsai, V. Chandrasekhar, D. Chen, and B. Girod, "Interframe coding of canonical patches for low bit-rate mobile augmented reality," *Int. J. Semantic Comput.*, vol. 7, no. 1, pp. 5–24, Mar. 2013.

[20] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proc. 9th IEEE Int. Conf. Comput. Vis. (ICCV)*, Washington, DC, USA, Oct. 2003, pp. 1470–1477.

[21] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. 7th Int. Joint Conf. Artif. Intell.*, Vancouver, BC, Canada, 1981, pp. 674–679.

[22] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, p. 13, Dec. 2006.

[23] 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.

[24] J. Jain and A. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. 29, no. 12, pp. 1799–1808, Dec. 1981.

[25] H.-Y. C. Tourapis and A. M. Tourapis, "Fast motion estimation within the H.264 codec," in *Proc. Int. Conf. Multimedia Expo*, Baltimore, MD, USA, Jul. 2003, pp. III-517–III-520.

[26] (2012). *Stanford Streaming Mobile Augmented Reality Dataset* [Online]. Availabile: http://purl.stanford.edu/ph459zk5920

[27] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting," *Commun. ACM*, vol. 24, no. 6, pp. 381–384, Jun. 1981.

[28] G. Francini, S. Lepsoy, and M. Balestri, "Selection of local features for visual search," *Signal Process., Image Commun.*, vol. 28, no. 4, pp. 311–322, Apr. 2013.

[29] *Digital Compression and Coding of Continuous-Tone Still Images*, document Rec. ISO/IEC 10918-1-ITU-T T.81, Sep. 1992.

[30] V. Chandrasekhar, "Low bitrate image retrieval with compressed histogram of gradients descriptors," Ph.D. dissertation, Dept. Elect. Eng., Stanford Univ., Stanford, CA, USA, Mar. 2013.

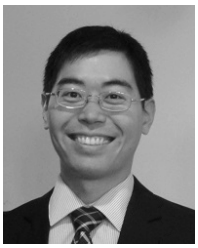
[31] K. L. Oehler and R. M. Gray, "Mean-gain-shape vector quantization," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Minneapolis, MN, USA, Apr. 1993, pp. 241–244.

[32] S. S. Tsai, D. Chen, G. Takacs, V. Chandrasekhar, J. P. Singh, and B. Girod, "Location coding for mobile image retrieval," in *Proc. 5th Int. Mobile Multimedia Commun. Conf.*, London, U.K., Sep. 2009.
[33] S. S. Tsai *et al.*, "Improved coding for image feature location information," in *Proc. SPIE*, San Diego, CA, USA, Aug. 2012.
[34] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, New York, NY, USA, Jun. 2006.
[35] M. J. Huiskes, B. Thomee, and M. S. Lew, "New trends and ideas in visual concept detection: The MIR flickr retrieval evaluation initiative," in *Proc. ACM Int. Conf. Multimedia Inform. Retr.*, Philadelphia, PA, USA, 2010.
[36] G. Takacs, V. Chandrasekhar, S. Tsai, D. Chen, R. Grzeszczuk, and B. Girod, "Rotation-invariant fast features for large-scale recognition and real-time tracking," *Signal Process., Image Commun.*, vol. 28, no. 4, pp. 334–344, Apr. 2013.
[37] G. Takacs, "Unified tracking and recognition with rotation-invariant fast features," Ph.D. dissertation, Dept. Elect. Eng., Stanford Univ., Stanford, CA, USA, Feb. 2012.

Mina Makar is a Senior Engineer with the Office of the Chief Scientist, Qualcomm Inc., San Diego, CA, USA. He received the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 2013. His research was in the areas of low bit-rate image-based retrieval, streaming mobile augmented reality, and video quality monitoring. He received the B.Sc. and M.Sc. degrees in electrical engineering from Alexandria University, Alexandria, Egypt, in 2004 and 2006, respectively.

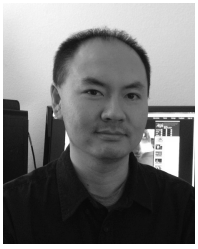
Dr. Mina was a recipient of the Top 10% Paper Award from the 13th IEEE International Workshop on Multimedia Signal Processing. His papers at international conferences have been finalists for the Best Student Paper Award three times, including ICIP 2012, ICASSP 2009, and Asilomar 2009 conferences. He has over 20 publications in the form of journal papers, conference papers, and MPEG contributions.

Vijay Chandrasekhar is currently a Scientist with the Institute for Infocomm Research, Singapore. He received the B.Sc. and M.Sc. degrees from Carnegie Mellon University, Pittsburgh, PA, USA, in 2005, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 2013. His research interests include mobile audio and visual search, large-scale image and video retrieval, machine learning, and data compression. He has authored more than 50 papers/MPEG contributions in a wide range of top-tier journals/conferences, such as IJCV, ICCV, CVPR, the IEEE SPM, ACM MM, the IEEE TIP, DCC, ISMIR, and MPEG-CDVS, and holds seven U.S. patents (one granted and six pending). His Ph.D. work on feature compression led to the MPEGCDVS (Compact Descriptors for Visual Search) standard, which he actively contributed from 2010 to 2013. He was a recipient of the A*STAR National Science Scholarship in Singapore in 2002.



Sam S. Tsai is currently pursuing the Ph.D. degree with the Department of Electrical Engineering, Stanford University, Stanford, CA, USA, where he is currently with the Image, Video and Multimedia Systems Group. He is also a Brown Fellow with the Brown Institute of Media Innovation. His research interests include mobile augmented reality systems and mobile visual search applications, in particular, how to efficiently use geometrical information and higher level text features.

He received the M.Sc. degree in electronics engineering from the Institute of Electronics, National Chiao Tung University, Hsinchu, China, in 2003. In his graduate studies, he was involved in the area of error resilient video coders and scalable video coding algorithms. He received the B.Sc. degree in electronics engineering at National Chiao Tung University in 2001.



David Chen is an Engineering Research Associate with the Department of Electrical Engineering, Stanford University, Stanford, CA, USA. He also serves as a fellow of the Brown Institute for Media Innovation.

His research focuses on image and video retrieval for mobile visual search applications, and he is generally interested in computer vision, signal processing, and machine learning. For his teaching, he received the Centennial TA Award in 2012 and the Outstanding TA Award in 2010. He was also a recipient of the Capocelli Prize for the Best Student Paper from the Data Compression Conference in 2014. He received the B.Sc., M.Sc., and Ph.D. degrees in electrical engineering from Stanford University in 2006, 2008, and 2014, respectively.

Bernd Girod (F'98) is the Robert L. and Audrey S. Hancock Professor of Electrical Engineering with Stanford University, Stanford, CA, USA. Until 1999, he was a Professor with the Department of Electrical Engineering, University of Erlangen-Nuremberg, Erlangen, Germany. His research interests are in the area of image, video, and multimedia systems. He has authored over 500 conference and journal papers, and six books. He was a recipient of the *EURASIP Signal Processing* Best Paper Award in 2002, the IEEE MULTIMEDIA COMMUNICATION Best Paper Award in 2007, the *EURASIP Image Communication* Best Paper Award in 2008, the *EURASIP Signal Processing* Most Cited Paper Award in 2008, the EURASIP Technical Achievement Award in 2004, and the Technical Achievement Award of the IEEE Signal Processing Society in 2011. As an Entrepreneur, he has been involved in several startup ventures, among them Polycom, Vivo Software, 8×8, and RealNetworks. He received the Ph.D. degree in engineering from the University of Hannover, Hannover, Germany, and the M.Sc. degree from the Georgia Institute of Technology, Atlanta, GA, USA. He is an EURASIP Fellow and a member of the German National Academy of Sciences (Leopoldina). He currently serves with the School of Engineering, Stanford University, Stanford, CA, USA, as the Senior Associate Dean of Online Learning and Professional Development.