International Journal of Semantic Computing
Vol. 7, No. 1 (2013) 5–24
©World Scientific Publishing Company
DOI: 10.1142/S1793351X13400011

INTERFRAME CODING OF CANONICAL PATCHES FOR LOW BIT-RATE MOBILE AUGMENTED REALITY

MINA MAKAR*, SAM S. TSAI†, VIJAY CHANDRASEKHAR‡,
DAVID CHEN§ and BERND GIROD¶

Department of Electrical Engineering
Stanford University, Stanford, CA 94305, USA
*mamakar@stanford.edu
†sstsai@stanford.edu
‡vijayc@stanford.edu
§dmchen@stanford.edu
¶bgirod@stanford.edu

Local features are widely used for content-based image retrieval and augmented reality applications. Typically, feature descriptors are calculated from the gradients of a canonical patch around a repeatable keypoint in the image. In this paper, we propose a temporally coherent keypoint detector and design efficient interframe predictive coding techniques for canonical patches and keypoint locations.

In the proposed system, we strive to transmit each patch with as few bits as possible by simply modifying a previously transmitted patch. This enables server-based mobile augmented reality where a continuous stream of salient information, sufficient for image-based retrieval and localization, can be sent over a wireless link at a low bit-rate. Experimental results show that our technique achieves a similar image matching performance at 1/15 of the bit-rate when compared to detecting keypoints independently frame-by-frame and allows performing streaming mobile augmented reality at low bit-rates of about 20–50 kbps, practical for today's wireless links.

Keywords: Augmented reality; canonical patch coding; image matching.

1. Introduction

Streaming Mobile Augmented Reality (MAR) applications require both real-time recognition and tracking of objects of interest in a video scene. Many of these applications utilize local image features for example, Scale-Invariant Feature Transform (SIFT) [1], Speeded Up Robust Features (SURF) [2], or Compressed Histogram of Gradients (CHoG) [3–5].

In an image matching framework, keypoints are detected from 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 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 [6]. Many feature descriptors [1–6] 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 with their maximum gradient along the same 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 [7]. The emerging MPEG standard, Compact Descriptors for Visual Search (CDVS) [8] targets designing low bit-rate descriptors that achieve state-of-the-art image matching performance. Many interesting proposals [9–11] have been compared against each other in a common evaluation framework [12]. In our previous work [13, 14], we show that one can alternatively 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 this paper, we extend the patch compression idea to motion video. To exploit the temporal correlation in a video, we design a temporally coherent keypoint detector that allows efficient interframe predictive coding for canonical patches and keypoint locations. We strive to transmit each patch or its equivalent feature descriptor with as few bits as possible by simply modifying a previously transmitted patch. The goal is to enable server-based streaming MAR where a continuous stream of salient information, sufficient for image-based retrieval and localization, can be sent over a wireless link at the smallest possible bit-rate. Initial results of our work on interframe patch encoding and differential location coding have been presented in [15]. In this paper, we review the work in [15], and provide detailed experimental results comparing different interframe patch coding techniques. We also compare our proposed technique to a system that streams the whole video to the server side.

The remainder of the paper is organized as follows. Section 2 explains the proposed temporally coherent keypoint detector including the patch matching technique and the improvements we achieve through using a similarity transform and an adaptive detection interval. Section 3 presents different patch encoding modes that exploit temporal correlation. The idea of differential location coding is presented in Sec. 4. Finally, in Sec. 5, we present experimental results showing the performance of the temporally coherent keypoint detector and predictive coding of canonical patches and keypoint locations in terms of achieving similar image matching performance at more than 15 × bit-rate reduction compared to detecting keypoints independently frame-by-frame.

2. Temporally Coherent Keypoint Detector

The first step in the extraction of local feature descriptors [1–6] on a still image is keypoint detection. These keypoints are represented by their location, orientation and scale and each keypoint corresponds to a canonical patch. To achieve temporal coherence, we propagate patches to consecutive video frames. Tracking of interest points in a video has been studied extensively, e.g., [16–19]. Our work introduces the idea of tracking to propagate canonical patches. We propose performing *canonical patch matching* for efficient patch propagation.

2.1. *Patch matching with displacement model*

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 [1] which relies on extrema detection in a difference of Gaussian (DoG) scale space representation of the frame. The patches corresponding to the keypoints detected by running a conventional detection algorithm are referred to 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. Figure 1 presents a block diagram of the proposed image matching pipeline for FP-Patches. The architecture we consider directly encodes canonical FP-Patches and extracts feature descriptors from the decoded patches at the receiver. Encoding techniques for FP-Patches are presented in Sec. 3. To detect FP-Patches, we do not run the keypoint detection algorithm again. Instead, 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 then

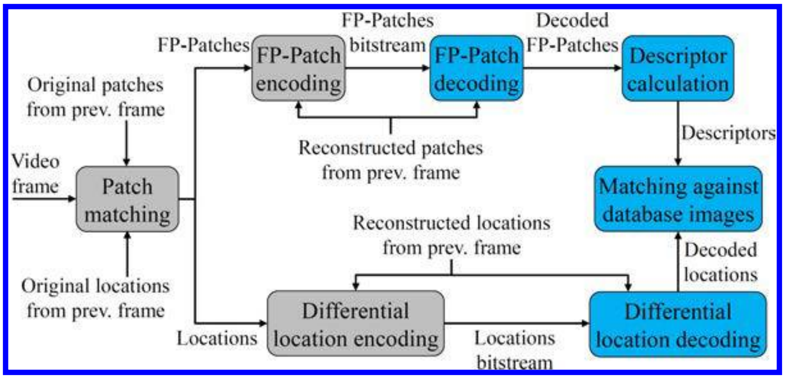

FP-Patches
FP-Patch encoding
FP-Patches bitstream
FP-Patch decoding
Decoded FP-Patches
Descriptor calculation
Original patches from prev. frame
Reconstructed patches from prev. frame
Descriptors
Video frame
Patch matching
Matching against database images
Reconstructed locations from prev. frame
Decoded locations
Original locations from prev. frame
Locations
Differential location encoding
Locations bitstream
Differential location decoding

Fig. 1. Image matching pipeline for FP-frames. Gray blocks on the left represent the transmitter (camera phone) side and blue blocks on the right represent the receiver (server) side.

define a search range to search for the most correlated patch in the current frame. 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. SAD distortion is calculated as in (1) where $\mathbf{P}_f$ and $\mathbf{P}_{f-1}$ are patches in frames f and $f-1$ respectively and N is the patch width or height in pixels. The patch with the minimum SAD is selected to be the corresponding FP-Patch in the current video frame. In this version of the algorithm, an FP-Patch inherits orientation and scale from the initial D-Patch.

$$\mathrm{SAD}(\mathbf{P}_f, \mathbf{P}_{f-1}) = \sum_{i=1}^{N}\sum_{j=1}^{N} |\mathbf{P}_f(i,j) - \mathbf{P}_{f-1}(i,j)| \tag{1}$$

Figure 2 illustrates the patch matching technique. It is similar to the idea of block matching used in video coding but applied to canonical patches. In Fig. 2(a), the black dot near the letter **C** corresponds to the x and y locations of a patch in the previous frame in *OpenCV* test sequence. The size and rotation angle of the square define the scale and orientation of this patch respectively. In Fig. 2(b), we search among candidate patches in frame f, with the same scale and orientation and varying x and y in a defined search range, for the patch with minimum SAD. The patch marked by the red thick square minimizes the distortion, and is thus selected as the FP-Patch corresponding to the patch in Fig. 2(a).

A finer search stage with smaller variations in x and y can follow the initial search stage once we obtain an initial good estimate of the matching patch. This is similar to

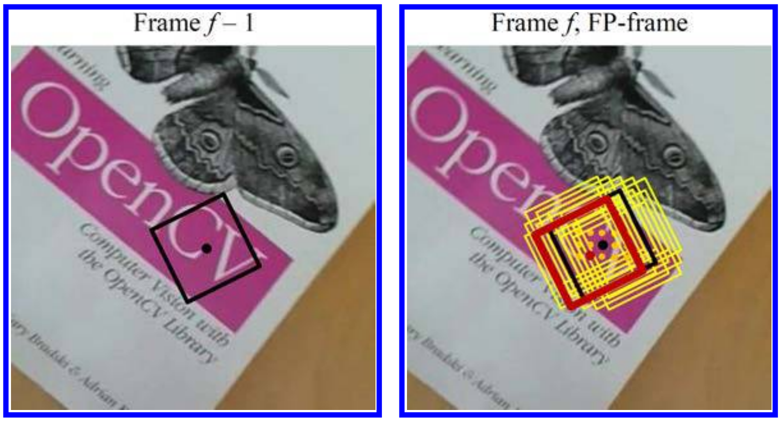


Fig. 2. Illustration of patch matching. (a) Example patch in the previous frame to be tracked in the current frame. (b) Patch matching searches for the patch of minimum SAD with respect to the patch in (a). We vary the keypoint location to cover the search range. The patch highlighted with a red thick square represents the result that minimizes the SAD distortion.

coarse-to-fine motion estimation in video coding [20]. After settling on the best patch with large varying steps, we test more patches around the initial matching patch estimate in a smaller search range with smaller varying steps. We refer to this operation as *hierarchical patch matching*. In our experiments, 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.

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 in the previous frame 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. This is because of the assumption that the patches only move with translational motion in a specific search range, while practical videos may contain more complex motion. Thus, after a pre-defined number of frames, we insert a D-frame by running the keypoint detector instead of propagating patches from the previous frame. We refer to the number of frames between consecutive D-frames as the *detection interval* Δ.

2.2. *Patch matching with similarity transform*

In Sec. 2.1, we only vary the x and y locations of the patches during patch matching. In the later image matching stage, we use keypoint locations for geometric consistency check and an update in the locations of the keypoints between frames is needed to accurately localize the object of interest in a particular video frame. 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 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 Δs. 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. This idea is illustrated in Fig. 3 for the same example patch shown in Fig. 2. Affine motion models have been explored before in motion estimation for video coding, see, e.g., [21, 22]. Although these models only result in a minor bit-rate reduction in video coding, we observe that for our problem, they introduce a significant gain in the tracking and image matching performance

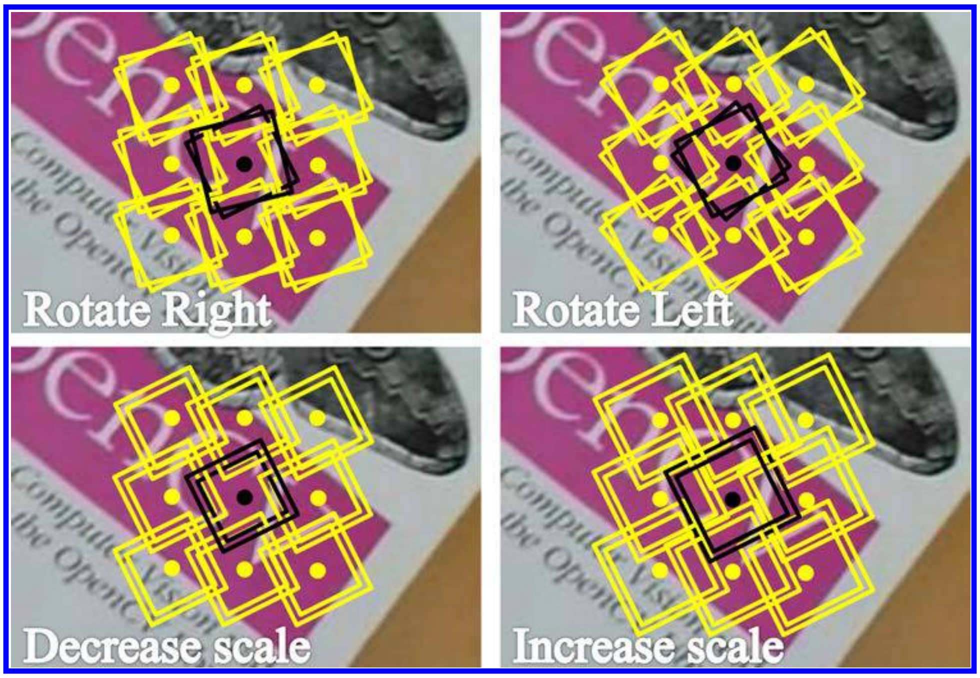


Fig. 3. Patch matching with similarity transform. We vary the keypoint location, orientation and scale during the search for the closest patch in the previous frame. The upper row shows varying keypoint orientation and the lower row shows varying keypoint scale.

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

To decide the search range for orientation and scale values, we extracted D-Patches from all the video frames in a training set and try to match the descriptors for each two consecutive video frames. We study the changes in orientation and scale between each two matching keypoints to indicate a reasonable search range for orientation and scale in patch matching. We generally find that the changes in orientation and scale are small between consecutive frames and a small search range with few search values is enough. Hence, we search 5 orientation differences (-0.1, -0.03, 0, 0.03 and 0.1 corresponding to $-5.7°$, $-1.7°$, $0°$, $1.7°$ and $5.7°$) and 5 scale differences (-0.5, -0.15, 0, 0.15 and 0.5).

We apply a brute force search for all different combinations of varying x, y, orientation and scale. Varying orientation and scale during patch matching generally improves the image matching performance. However, we observe that using the four-parameter model during the initial search phase, that includes the whole search range for locations, sometimes hurts the geometric consistency check which only considers keypoint locations. We achieve the best image matching results when we include the orientation and scale search only during the fine location search which considers a 2×2 search range for x and y.

2.3. *Adaptive detection interval*

An advantage of the proposed patch matching technique is avoiding the computational complexity of performing keypoint detection on FP-frames. However, 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 when there are occlusions or scene-cuts or when objects of interest are entering or leaving the scene. To solve this problem we propose an adaptive Δ where we insert a D-frame whenever patch matching deteriorates.

We acquire both types of patches for each video frame. D-Patches are acquired through running the keypoint detector and FP-Patches are acquired through patch matching. We measure the deviation between D-Patches and FP-Patches and refer to the deviation measure as D-FP$_{SAD}$. A small deviation means that the FP-Patches are well representing the scene; 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 deviation measure is calculated as follows.

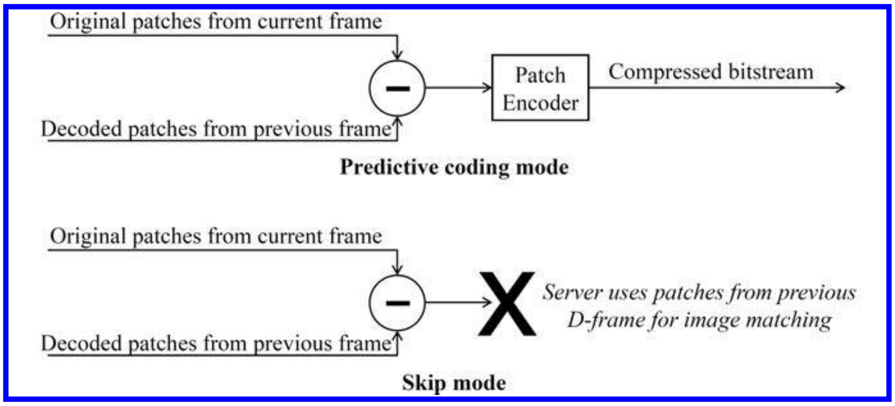
The first frame in the video is always a D-frame. For each new frame, we calculate both FP-Patches and D-Patches. For each patch in the FP-Patches we search for a corresponding patch in the D-Patches with the minimum SAD. The deviation measure D-FP$_{SAD}$ for a certain frame equals the average SAD between the FP-Patches and their closest corresponding D-Patches. A threshold D-FP$_{th}$ is defined. If D-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 define another threshold $\Delta_{\max}$ for the maximum length of the detection interval. If the number of consecutive FP-frames reaches this threshold, the next frame is decided to be a D-frame.

3. Canonical Patch Encoding

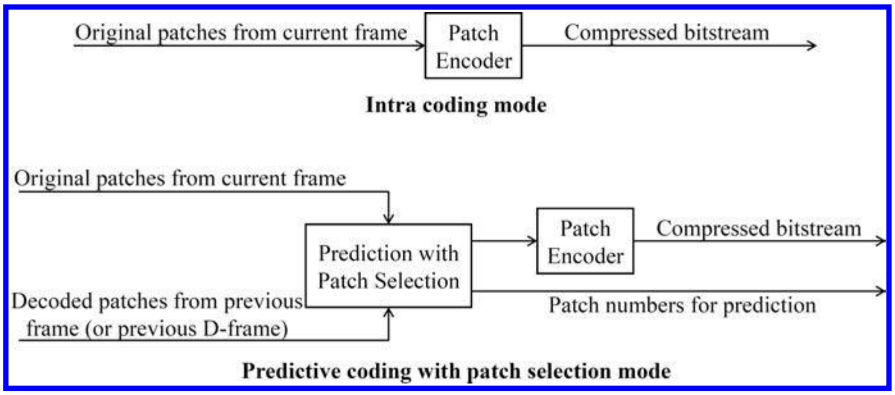
We use our own Patch ENCoder (PENC) [14] to encode both FP-Patches and D-Patches. PENC is similar to JPEG [23]. However, PENC applies a pre-processing step of Gaussian blurring and mean removal on the patches and always uses a 2D DCT transform with the same size of the patch in order to avoid producing blocking artifacts within the patch. PENC uses Huffman tables that are trained for patch statistics which are different from natural image statistics. Figure 4 summarizes the patch encoding modes. These modes are described in details in the following subsections.

3.1. *Coding of FP-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


Original patches from current frame
Decoded patches from previous frame
Patch Encoder
Compressed bitstream
Predictive coding mode
Original patches from current frame
Decoded patches from previous frame
Server uses patches from previous D-frame for image matching
Skip mode

(a) FP-Patches encoding modes


Original patches from current frame
Patch Encoder
Compressed bitstream
Intra coding mode
Original patches from current frame
Prediction with Patch Selection
Patch Encoder
Compressed bitstream
Decoded patches from previous frame (or previous D-frame)
Patch numbers for prediction
Predictive coding with patch selection mode

(b) D-Patches encoding modes

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

the corresponding patch in the previous frame and only the residual signal between patches is encoded using PENC. In Sec. 2.1, 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. 1 and 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 clean 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).

3.2. *Coding of D-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 usually used in conjunction with *Skip* mode for FP-Patches (Sec. 3.1), because we only use D-Patches for prediction and do not 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. Figure 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.

4. Differential Location Coding

In *Skip* mode (Sec. 3.1), we avoid the transmission of the residuals of FP-Patches 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 transmit the new 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 location coder developed by Tsai *et al.* [25, 26]. Since the keypoint locations in FP-frames are predicted from the corresponding keypoint locations in the previous frame, it is more efficient to only transmit the difference between the keypoint locations of corresponding patches in consecutive frames. Measuring the statistics of the location differences between consecutive video frames, we sometimes observe a global shift for the object of interest. This results in a probability distribution of the difference values that follows a Laplacian distribution centered on the global shift value. We propose a differential location coder that works as follows.

First, we measure the differences between the original keypoint locations of the current FP-frame and the reconstructed corresponding keypoint locations of the previous frame. These location differences are quantized using a quantization step Q_{loc} in both x and y directions. Second, we measure the histograms of the quantized location differences in x and y in the current frame, and estimate the values of global shift from the peaks of these histograms. We refer to the global shift values in x and y directions as G_x and G_y respectively. The values of G_x and G_y are encoded separately using a signed exponential Golomb code where the symbols 0, 1, -1, 2, -2, 3, $-3\ldots$ are encoded using the codewords 1, 010, 011, 00100, 00101, 00110, 00111 ... and so on.

The last step in the differential coding of locations is the subtraction of G_x from the quantized location differences in x direction and G_y from the quantized location differences in y direction. The quantized location differences after removing the global shift are 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.

To measure the effect of location quantization on the image matching performance, we use RANSAC [27] for geometric consistency check and investigate two different metrics: the number of feature matches post-RANSAC and the localization accuracy. The localization accuracy is measured by calculating the *Jaccard* index between the ground-truth location of the object of interest and the projected location that is inferred after matching. We manually generate the ground-truth location by defining a bounding quadrilateral around the object of interest in each video frame. From RANSAC results, we project the vertices of the object in the clean image on each video frame. The Jaccard index is defined as the ratio between the area of overlap between the ground-truth and the projected quadrilaterals, and the area of the union of the two quadrilaterals. Hence, Jaccard index values close to 1 indicate excellent localization.

For applications where we need to transmit orientation and scale, we also apply differential coding by transmitting differences from the previous frame. Since we search few values for orientation and scale (Sec. 2.2), we do not apply further quantization on the encoded values. The statistics of the difference values in orientation and scale usually follow a Laplacian distribution and thus, we use a signed exponential Golomb code to encode them. For example, if we allow the five values -0.5, -0.15, 0, 0.15 and 0.5 for the differences in keypoint scale, then the corresponding codewords are 00101, 011, 1, 010 and 00100 respectively.

5. Experimental Results

We introduce the *Stanford streaming MAR dataset* [28]. The dataset consists of 23 videos, divided to four categories: *Books, CD covers, DVD covers* and *Objects.*

All these videos contain a single object of interest, recorded with a hand-held mobile phone with different amounts of camera motion, glare, blur, zoom, rotation and perspective changes. Each video is 100 frames long, recorded at 30 fps with resolution 640×480. The frames of each video sequence are matched against a still image of the corresponding clean object of interest. We provide the ground-truth localization information for 8 videos where we manually define a bounding quadrilateral around the object of interest. This localization information is used in the calculation of the Jaccard index (see Sec. 4). The dataset is available for download at [28].

In this paper, we perform experiments on eight videos from the *Stanford streaming MAR dataset*, two videos from each category. Figure 5 presents the videos used in our experiments. These videos, in the order shown in figure, are *OpenCV, Wang Book, Barry White, Janet Jackson, Monsters Inc, Titanic, Glade* and *Polish*. We first present the clean database image followed by an example frame from the corresponding video. In experiments that require training of probability tables, we use video sequences from [15] for training purposes. These videos are not part of the *Stanford streaming MAR dataset*.

In the next subsections, we first evaluate different parts of the proposed streaming MAR system separately, then finally present the overall system performance in Sec. 5.4. In all experiments, we calculate modified SIFT descriptors without

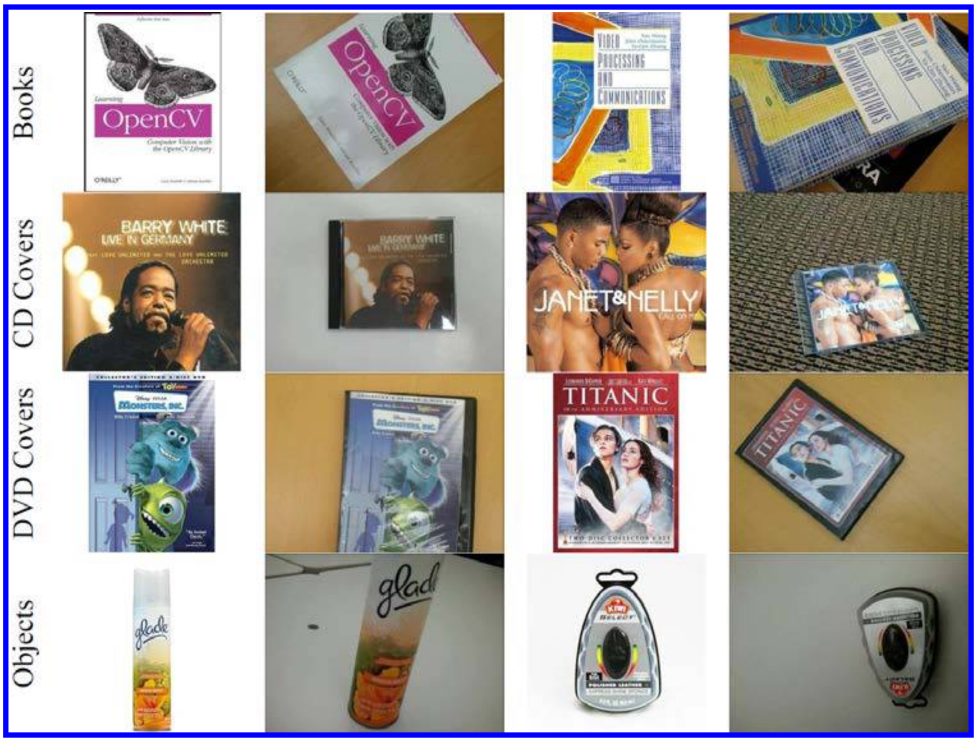


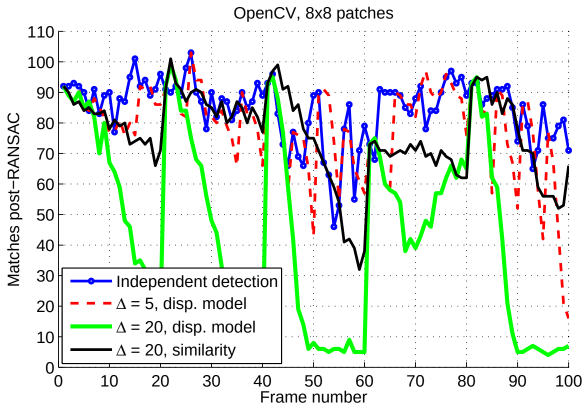
Fig. 5. Database images and example video frames from the *Stanford streaming MAR dataset*.

magnitude weighting due to their state-of-the-art performance for image matching and retrieval applications [24], and use symmetric KL divergence as a distance metric for descriptor matching.

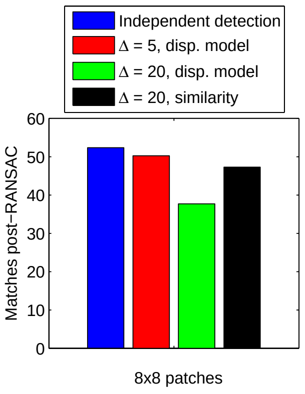
5.1. *Performance of temporally coherent detection*

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 in the CDVS Test Model (TM) [9] and we set the maximum number of extracted patches from a video frame to 200. This feature selection algorithm chooses the features with higher matching probabilities based on their scale, orientation, output of the DoG pyramid and distance from the image center. First, we study the feature matching performance of 8×8, 16×16 and 32×32 patches without compression of patches or keypoint locations. This is to illustrate the performance of the proposed patch matching technique and its improvements as explained in Sec. 2. For patch termination, we use $SAD_{term} = 450$ for 8×8 patches, 1800 for 16×16 patches and 7200 for 32×32 patches.

Figure 6 compares the image matching performance for different values of the detection interval Δ and the use of a displacement model versus a similarity transform for patch matching. Figure 6(a) shows the plot for *OpenCV* test sequence where the x-axis represents the frame number and the y-axis represents the number of feature matches post-RANSAC. This experiment uses 8×8 patches; however, to perform image matching, we always upsample the patches to size 32×32 using bilinear interpolation and then calculate descriptors on the upsampled patches.



(a)



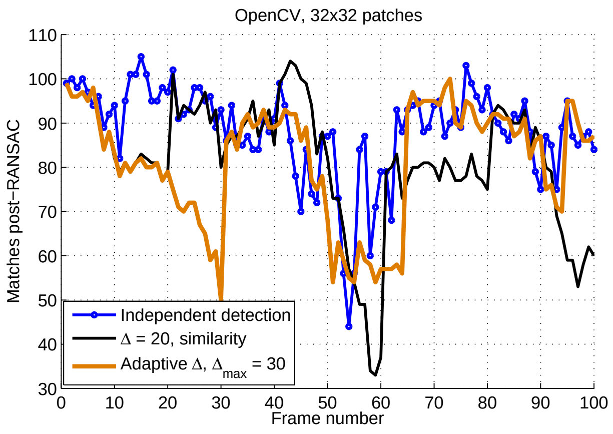
(b)

Fig. 6. Matching performance of the proposed patch matching technique. We compare running the keypoint detector every frame (independent detection) to performing patch matching with different detection intervals. '**disp. model**' refers to only varying x and y in the search range while '**similarity**' refers to including the keypoint orientation and scale in the search. (a) Example video: *OpenCV*. (b) Average performance over all videos with patch size 8×8.

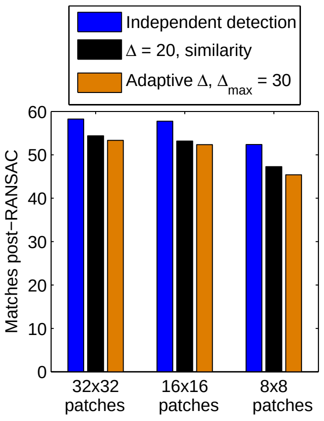
The plot compares four different curves. First, we plot the number of matches if we run the SIFT keypoint detector every frame; i.e., all frames are D-frames or $\Delta = 1$. We refer to this result as *independent detection* and it serves as a benchmark for other techniques since our goal is to obtain a matching performance close to this curve when applying the temporally coherent detector. The second curve uses a short Δ of 5 frames and only searches keypoint locations as in Sec. 2.1. The third curve uses a longer Δ of 20 frames and searches the same locations as the second curve. The last curve includes 5 orientation differences and 5 scale differences in the search range as described in Sec. 2.2.

If only searching keypoint locations, we find that a short Δ maintains a good matching performance compared to the independent detection case. However, a large drop in matching is observed when increasing Δ to 20. Adding orientation and scale search for a similarity transform motion model helps reducing most of the performance drop and results in an acceptable matching performance. An example drawback of using a fixed Δ is observed at frames 61 to 80. Although the performance of independent detection improves at frame 63, the performance of the other curves does not improve until the next D-frame. Figure 6(b) shows the average number of feature matches over the eight test sequences. We observe a large drop in performance when using $\Delta = 20$ with a displacement model for patch matching.

Figure 7 presents the results of the adaptive detection interval idea described in Sec. 2.3 with $D - FP_{th} = 450$ for 8×8 patches, 1800 for 16×16 patches or 7200 for 32×32 patches, and $\Delta_{\max} = 30$. These parameters are chosen to obtain a reasonable number of D-frames that significantly reduces the bit-rate but still maintains a good matching performance. Figure 7(a) uses *OpenCV* as an example test sequence. Comparing the performance to the case of fixed Δ, we find that the performance of



(a)



(b)

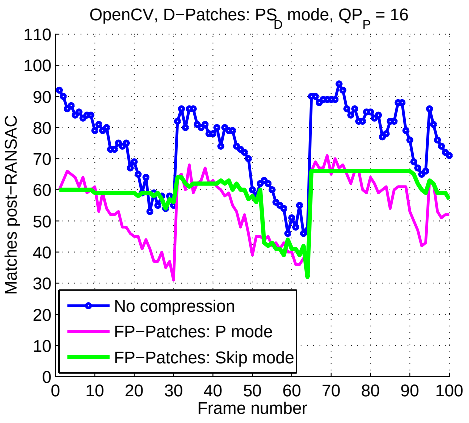
Fig. 7. Matching performance of patch matching technique when using a fixed versus an adaptive detection interval. Using an adaptive Δ better follows the performance of independent detection. (a) Example video: *OpenCV*. (b) Average performance over all videos with different patch sizes.

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 55 to 60, 65 to 80 and 94 to 100), we observe that the adaptive Δ technique solves this problem by using D-frames at more suitable time instances and results in a larger number of feature matches.

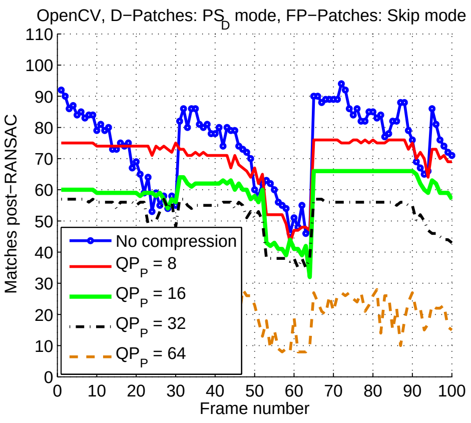
Figure 7(b) presents the average matching performance over all the test sequences and compares different patch sizes. The average number of feature matches for adaptive Δ is slightly lower than $\Delta = 20$. This is because we allow a larger upper bound on the adaptive Δ value ($\Delta_{\max} = 30$) to further reduce the bit-rate than $\Delta = 20$ case when patch encoding is considered. Comparing different patch sizes, we see that reducing the size from 32×32 to 16×16 causes about 2% drop in the average number of post-RANSAC matches 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. Therefore, our following experiments which consider encoded patches will use 8×8 patches.

5.2. *Effect of canonical patch encoding*

We start from the matching results of the uncompressed patches for the adaptive Δ case and use PENC [14] to compress the patches with the different encoding modes illustrated in Fig. 4. In Fig. 8, we compare different encoding modes and different values of the patch compression quantization parameter QP_P using the example test



(a)



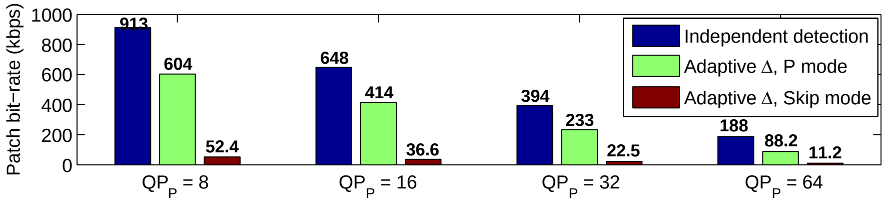
(b)

Fig. 8. Matching performance after interframe patch encoding for the test sequence *OpenCV*. 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.

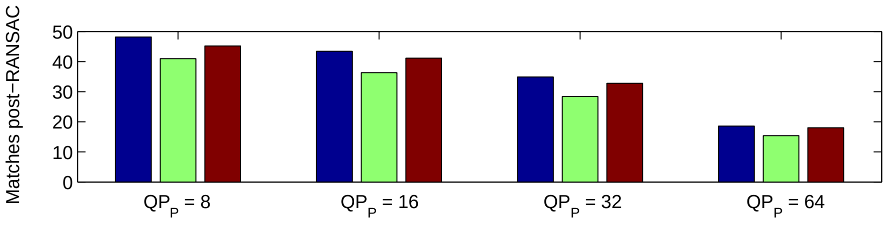
sequence *OpenCV*. D-Patches are encoded with PS_D mode since it usually outperforms I mode due to exploiting the temporal correlation in the video.

In Fig. 8(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 uses the exact same descriptors over the whole detection interval (after removing the terminated patches) and only changes the keypoint locations used for geometric verification. Figure 8(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 $QP_P = 64$.

Figure 9 presents the patch encoding bit-rates averaged over all 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. D-Patches are always encoded using PS_D mode in the three cases. *Skip* mode results in a large bit-rate reduction with a matching performance comparable to both P mode and independent detection. With *Skip* mode, we obtain bit-rate savings of about $9\times$ compared to P mode or $15\times$ compared to independent detection. We observe that $QP_P = 16$ and $QP_P = 32$ achieve a good compromise between the matching accuracy and bit-rate. We target



(a)



(b)

Fig. 9. 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 $9\times$ bit-rate reduction over P mode and $15\times$ reduction over independent detection at a comparable matching performance. (a) Patch bit-rates. (b) Average number of feature matches post-RANSAC.

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 $\Delta_{\max}$ and combining PS_D and *Skip* encoding modes.

5.3. *Effect of lossy location coding*

The results reported in the previous subsections use unquantized locations. In this subsection, we quantize and encode locations as described in Sec. 4. We run the location coder along with the proposed canonical patch encoder (see Fig. 1) using the encoding modes that achieve the best results in Sec. 5.2. PS_D mode is used with D-Patches and *Skip* mode is used with FP-Patches. We present results at a fixed patch encoding quantization parameter $QP_P = 32$.

Figure 10(a) presents 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*. We use the same Q_{loc} for both D-frames and FP-frames and vary the location bit-rates by varying $Q_{loc} = 2, 4$ and 8. $Q_{loc} = 4$ results in good image matching accuracy at an acceptable location bit-rate. In Fig. 10 (b), we compare the Jaccard index at different location bit-rates for the same video sequences and the same encoding parameters in Fig. 10(a). The Jaccard index is averaged over all the frames in each video sequence. The localization accuracy drops with coarser quantization of keypoint locations and $Q_{loc} = 4$ results in acceptable compromise between bit-rate and accuracy. Comparing Fig. 10(b) to Fig. 10(a), 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.

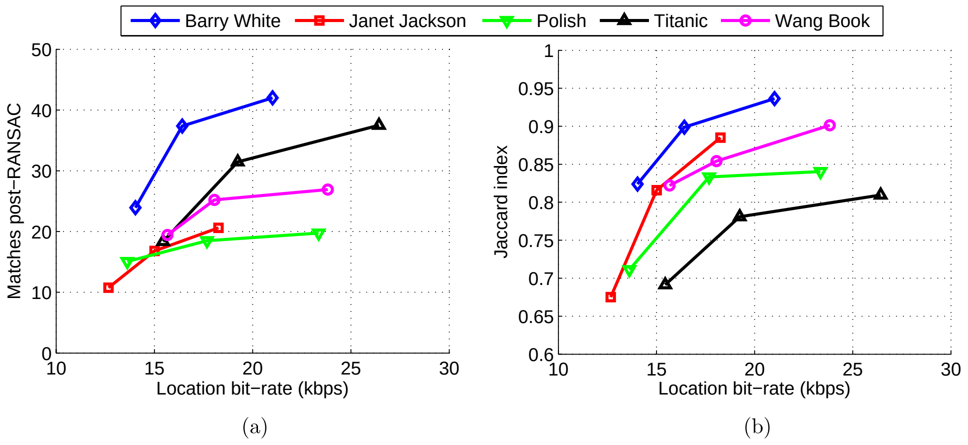


Fig. 10. Image matching performance and localization accuracy after location coding. $Q_{loc} = 2, 4$ and 8 values are used and a large drop at $Q_{loc} = 8$ is observed. (a) reports the matching performance and (b) the Jaccard index for patches encoded at $QP_P = 32$.

At $Q_{loc} = 4$, we achieve accurate localization results at location bit-rates of about 20 kbps. At this quantization step, the average bit-rate used for encoding a keypoint location in a D-frame = 6.65 bits/location. However, the average bit-rate for a keypoint location in an FP-frame = 3.1 bits/location. This indicates that differential encoding of keypoint locations results in more than 50% bit-rate reduction when compared to non-differential methods [26].

5.4. *Comparison to video streaming based scheme*

An alternative to the proposed keypoint detection and predictive coding techniques is to stream the whole video to the augmented reality server and perform keypoint detection, feature extraction and image matching on all video frames on the server side. We refer to this method as *Send Video* scheme. We implement *Send Video* scheme and compare its image matching performance to our proposed schemes. Videos are encoded using H.264 standard [29] using the following encoding parameters: VGA resolution, IPPP... structure, intra-period = 50 frames, $QP_{P-frames} = \{38, 42, 46, 50\}$ and $QP_{I-frames} = QP_{P-frames} - 3$. 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 canonical patches or their corresponding keypoint locations.

Figure 11 represents an overall comparison between the proposed schemes *Send Patches* (see Fig. 1) versus *Send Video*. In *Send Patches*, we use PS_D mode for D-Patches with $QP_P = \{8, 16, 32, 64\}$, *Skip* mode for FP-Patches and $Q_{loc} = 4$ for location coding. The reported bit-rate represents the sum of bit-rates of patches and 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

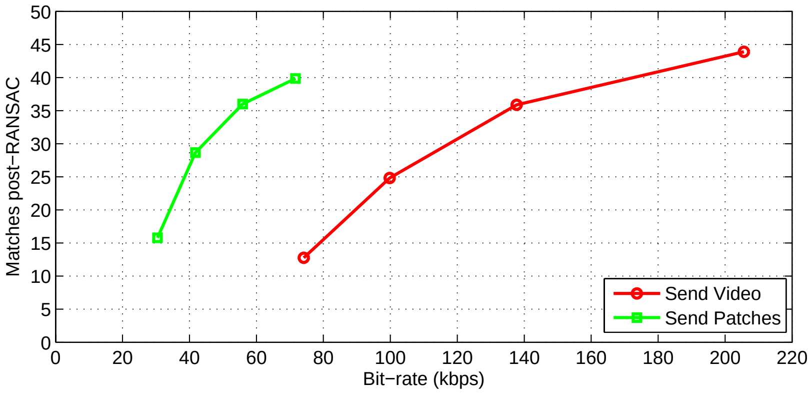


Fig. 11. Comparison between different streaming mobile augmented reality schemes. The bit-rate for *Send Patches* includes the encoded keypoint location information. *Send Patches* achieves 2.5 × bit-rate reduction, compared to *Send Video* scheme at the same average number of feature matches.

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 scheme achieves a huge improvement over *Send Video* scheme. At the same average number of feature matches, *Send Patches* achieves $2.5 \times$ bit-rate reduction over *Send Video* scheme. The performance of *Send Patches* saturates at a lower value than *Send Video* due to the loss resulting from using smaller size 8×8 patches to lower the bit-rate as illustrated in Fig. 7(b). Using *Send Patches*, we can achieve efficient streaming mobile augmented reality at bit-rates around 40 kbps. These bit-rates are practical for today's wireless links and better than streaming the whole video. For applications which do not require accurate localization, we may transmit the keypoint locations at a lower frame rate, and thus operate at a lower bit-rate of about 20 kbps.

6. 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 keypoint locations using efficient predictive coding techniques. We compare ourselves against the independent detection of keypoints frame-by-frame and also to streaming the whole video to the augmented reality server.

Experimental results show that the proposed temporally coherent detection mechanism results in an image matching performance comparable to the oblivious detection of new keypoints every video frame. Our interframe canonical patch encoder results in a similar image matching performance at 1/15 of the bit-rate used for encoding non-coherent patches. Differential location coding results in 50% bit-rate reduction over independent coding of locations. The overall system can achieve efficient streaming mobile augmented reality at bit-rates of about 20−50 kbps, practical for today's wireless links and less than half of the bit-rate needed to stream the whole video.

References

[1] D. Lowe, Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision* **60** (2004) 91−110.

[2] H. Bay, T. Tuytelaars and L. V. Gool, SURF: Speeded up robust features, in *Proc. European Conference on Computer Vision*, Graz, Austria, May 2006.

[3] V. Chandrasekhar *et al.*, CHoG: Compressed histogram of gradients — A low bit-rate feature descriptor, in *Proc. IEEE International Conference on Computer Vision and Pattern Recognition*, Florida, USA, June 2009.

[4] V. Chandrasekhar *et al.*, Quantization schemes for low bitrate compressed histogram of gradients descriptors, in *Proc. International Workshop on Mobile Vision, Conference on Computer Vision and Pattern Recognition*, San Francisco, USA, June 2010.

[5] V. Chandrasekhar *et al.*, Compressed histogram of gradients: A low-bitrate descriptor, *International Journal on Computer Vision* **94**(5) (2011).

[6] B. Girod *et al.*, Mobile visual search, *IEEE Signal Processing Magazine* **28**(4) (2011) 61–76.

[7] V. Chandrasekhar *et al.*, Survey of SIFT compression schemes, in *Proc. Second International Workshop on Mobile Multimedia Processing*, Istanbul, Turkey, Aug. 2010.

[8] B. Girod, V. Chandrasekhar, R. Grzeszczuk and Y. Reznik, Mobile visual search: Architectures, technologies, and the emerging MPEG standard, *IEEE Multimedia* **18**(3) (2011) 86–94.

[9] G. Francini, S. Lepsoy and M. Balestri, Description of test model under consideration for CDVS, *ISO/IEC JTC1/SC29/WG11/N12367*, Dec. 2011.

[10] V. Chandrasekhar *et al.*, Improvements to the test model under consideration with low memory descriptors, *ISO/IEC JTC1/SC29/WG11/M23580*, Feb. 2012.

[11] S. Paschalakis *et al.*, CDVS CE2: Local descriptor compression proposal, *ISO/IEC JTC1/SC29/WG11/M25929*, July 2012.

[12] Y. Reznik, G. Cordara and M. Bober, Evaluation framework for compact descriptors for visual search, *ISO/IEC JTC1/SC29/WG11/N12202*, July 2011.

[13] M. Makar, C.-L. Chang, D. Chen, S. Tsai and B. Girod, Compression of image patches for local feature extraction, in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, Taipei, Taiwan, April 2009.

[14] M. Makar, H. Lakshman, V. Chandrasekhar and B. Girod, Gradient preserving quantization, in *Proc. IEEE International Conference on Image Processing*, Florida, USA, Sept. 2012.

[15] M. Makar, S. Tsai, V. Chandrasekhar, D. Chen and B. Girod, Interframe coding of canonical patches for mobile augmented reality, in *Proc. IEEE International Symposium on Multimedia*, Irvine, USA, Dec. 2012.

[16] J. Shi and C. Tomasi, Good features to track, in *Proc. IEEE International Conference on Computer Vision and Pattern Recognition*, Seattle, USA, June 1994.

[17] H. Tao, H. S. Sawhney and R. Kumar, Object tracking with Bayesian estimation of dynamic layer representations, *IEEE Trans. Pattern Analysis and Machine Intelligence* **24**(1) (2002).

[18] D. Comaniciu, V. Ramesh and P. Meer, Kernel-based object tracking, *IEEE Trans. Pattern Analysis and Machine Intelligence* **25**(5) (2003).

[19] A. Yilmaz, O. Javed and M. Shah, Object tracking: A survey, *ACM Computing Surveys* **38**(4) (2006).

[20] P. Yin, H. C. Tourapis, A. M. Tourapis and J. Boyce, Fast mode decision and motion estimation for JVT/H.264, in *Proc. IEEE International Conference on Image Processing*, Barcelona, Spain, Sept. 2003.

[21] H. Lakshman, H. Schwarz and T. Wiegand, Adaptive motion model selection using a cubic spline based estimation framework, in *Proc. IEEE International Conference on Image Processing*, Hong Kong, Sept. 2010.

[22] T. Wiegand, E. Steinbach and B. Girod, Affine multipicture motion-compensated prediction, *IEEE Trans. Circuits Syst. Video Technol.* **15**(2) (2005) 197–209.

[23] ITU-T and ISO/IEC JTC1, Digital compression and coding of continuous-tone still images, *ISO/IEC 10918-1 - ITU-T Recommendation T.81*, Sept. 1992.

[24] V. Chandrasekhar, Low bitrate image retrieval with compressed histogram of gradients descriptors, Ph.D. dissertation, EE Dept., Stanford University, Stanford, USA. (to be published).

[25] S. Tsai *et al.*, Location coding for mobile image retrieval, in *Proc. International Mobile Multimedia Communications Conference*, London, UK, Sept. 2009.

[26] S. Tsai *et al.*, Improved coding for image feature location information, in *Proc. SPIE Workshop on Applications of Digital Image Processing*, San Diego, USA, August 2012.

[27] M. Fischler and R. C. Bolles, Random sample consensus: A paradigm for model fitting, *Commun. ACM* **24**(6) (1981) 381–384.

[28] *Stanford Streaming Mobile Augmented Reality Dataset*, 2012, http://blackhole3.stanford.edu/mar/

[29] 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.* **13**(7) (2003) 560–576.