Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2016) 24: 4254 – 4266

doi:10.3906/elk-1502-230

Keyframe-based video mosaicing for historical Ottoman documents

Ediz ŞAYKOL*
Department of Computer Engineering, Faculty of Engineering and Architecture, Beykent University, İstanbul, Turkey

Received: 26.02.2015 • **Accepted/Published Online:** 29.07.2015 • **Final Version:** 20.06.2016

Abstract: Image mosaicing is a trendy research area in computer vision in recent years. Instead of images, video frames can be mosaiced to form a larger aggregate. In both cases, merging computations generally depend on the common parts in consecutive images, which are called overlapping regions. The presence of mobile devices with a camera provides higher image and video acquisition; hence video mosaicing techniques can be utilized to digitize and to create high resolution historical document images for later use in browsing and retrieval systems. Here, we present a technique to create high resolution images of historical Ottoman documents from video. The video captures the frames while tracing the document in row major ordering. In the preprocessing steps, the frames are oriented automatically by the help of tracking the optical flow, and the keyframes are selected such that a sufficient overlapping region in between is guaranteed. Then the scanning phase traces these keyframes in horizontal and vertical manner to create the final video mosaic image in higher resolution. Both horizontal and vertical scan operations utilize a distance vector to compute similarity between two adjacent images. The effectiveness of the video mosaicing approach for historical Ottoman documents is evaluated, and the results of the experiments show that the approach is expressive yet effective to create high resolution Ottoman images.

Key words: Video mosaicing, distance vector, keyframe selection, image mosaicing, historical document images

1. Introduction

Image mosaicing is a trendy topic in computer vision. It can be simply defined as the automatic alignment (or registration) of multiple images into larger aggregates [1]. In many cases, it may not be easy to capture a larger image as a whole; hence a partial capturing scheme to cover the entire image is required. Document image mosaicing is a typical application domain, since there could be various sizes of document images. The process of document image analysis and processing then requires mosaicing of split document images to obtain a complete single image. The existence of overlapping regions among consecutive (sub)images is an important clue to merge them in a mosaic image. The entire process generally involves feature point extraction, image registration, homography computation, warping, and blending steps [2, 3]. However, enhanced techniques to track the original time of events for individual objects acting in scenes might be required (e.g., dynamic mosaicing [4], a sequential graph structure [5]).

Video frames can also be input such that the frames having sufficient overlapping regions in between need to be selected to achieve better performance. This process is called video mosaicing. The presence of mobile devices that are equipped with a camera has clearly facilitated image acquisition, especially capturing document

*Correspondence: ediz.saykol@beykent.edu.tr

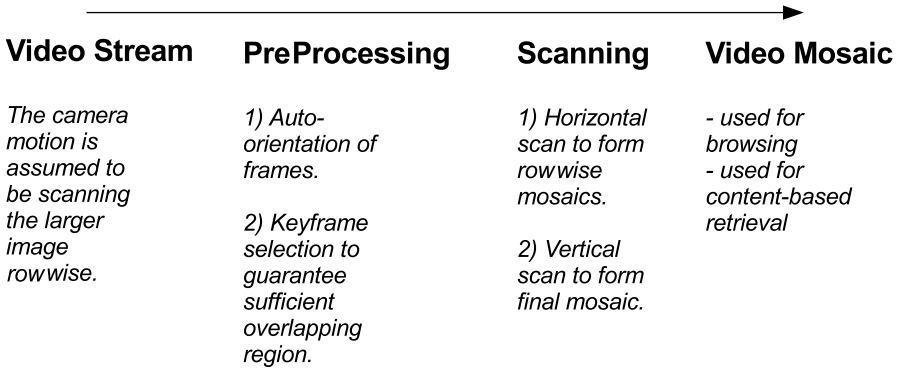


Figure 1. Overall dataflow during video mosaicing process. The processing has two phases: the first phase includes auto-orienting frames and generating keyframes. The second phase includes horizontal and vertical scanning to produce video mosaic.

images of various types including thick books or historical monuments. Since these devices are also capable of capturing video, video mosaicing techniques can be utilized to create high resolution historical document images for later use in browsing and retrieval systems. There exist browsing and retrieval systems in the literature for various historical scripts; for example in [6] a browsing system for Ottoman script is presented, whereas in [7] a content-based retrieval module is proposed.

Historical document images generally include writings and drawings together, possibly including texture, and the sides are generally tortured. This makes it harder to capture the whole historical document with a single shot having enough resolution. One of our motivations is this fact, and we present a video mosaicing approach for historical Ottoman documents in which the resultant high resolution video mosaic image could be used for browsing and content-based retrieval purposes. The videos are assumed to be captured from a hand-held camera with constant zoom during camera motion to guarantee that the sizes/resolution of the captured frames are equal, as in [8, 9]. Our video mosaicing technique processes historical Ottoman documents in two phases. The first phase is the preprocessing phase and includes auto-orienting frames and generating keyframes during camera motion. The second phase is devoted to horizontal and vertical scan processes. Figure 1 shows the data flow during our historical video mosaicing process. In the first phase, auto-orientation of frames is based on the computation of the optical flow in video by the well-known Kanade–Lucas–Tomasi (KLT) feature tracking algorithm [10, 11]. In the second phase, the similarity of the rows in vertical scanning and the similarity of columns in horizontal scanning are computed by using an adaptation of the distance histogram of the histogram based approach (HBA) [12], which was shown to be expressive in processing connected Ottoman script [7].

The main contributions of this study can be listed as follows:

- The video mosaicing approach produces a higher resolution image for the historical Ottoman document when compared to a single image capturing the document as a whole. This high-resolution output can be directly used in content-based retrieval and intelligent browsing systems (e.g., [6, 7]). Instead of a single video, capturing an adequate number of single images that have sufficient overlapping regions can be used as an alternative; however, it seems quite hard and erroneous for a user to capture a set of these images. Using a single video facilitates the tasks of the user during input acquisition. To the best of our knowledge, this work is the first video mosaicing approach targeting historical Ottoman documents to provide high quality images for retrieval and browsing purposes.

- The preprocessing phase auto-orients the captured frames and produces keyframes with sufficient overlapping regions in between. This utilizes KLT-based optical flow analysis and provides keyframe-based image mosaicing. Hence, the processing deals only with a smaller set of frames (i.e. keyframes), which leads to an efficient video mosaicing approach in terms of time and space. Moreover, these tasks help the technique capture an adequate number of single images that have sufficient overlapping regions.

- During the scanning phase, the row and column similarities are computed based on the distance vector, which is an adaptation of the distance histogram in the HBA [12]. It was shown to be expressive in encoding the shape information based on the distribution of the pixels, and was invariant under scale, rotation, and translation. Here, the distance vector is utilized to detect overlapping rows/columns efficiently between two keyframes during the scanning phase.

The rest of the paper is organized as follows: in Section 2, the related studies are presented along with comparisons to our technique. Section 3 presents the historical video mosaicing algorithm to form a unified mosaic image from individual frames captured by a mobile camera. The video mosaicing approach on a sample historical Ottoman document is also provided. The results of the performance experiments to evaluate the motivations and achievements of the proposed technique are given in Section 4. Finally, Section 5 concludes the paper and states the future work.

2. Related studies

There are effective techniques in the literature that are applicable to document images. Each of these techniques has advantageous points as well as limitations. Along with the increase in digital cameras, Doermann et al. [2] provide a comprehensive survey on especially textual document image analysis and mosaicing approaches. A recent review of various image mosaicing techniques is also presented in [3] for document images. Whichello and Yan [9] propose an automatic mosaicing process for a set of captured document images, as an offline montage. The shifts of the captured images are computed by correlating them using an image pyramid technique. The technique is verified on binarized document images. Zappala et al. [13] propose a feature-based document mosaicing approach by using an estimation scheme for the motion from point correspondence. An exhaustive search is used and the method demands a 50% overlapping region between images to produce a mosaiced image. Since the approach is limited to only text documents, it is unable to handle images containing both text and pictures.

Shivakumara et al. [14] present a more powerful approach that works for various types of document images. The main idea is to find the overlapping region efficiently, and the approach is based on comparing and matching sum of the pixel values within a specific window of the images to identify the overlapping region. This is achieved by comparing the sum of values of pixels of each window in the split images. Marzotto et al. [15] propose a high-resolution mosaicing technique for a video stream where it is necessary to use global alignment/registration. Their technique achieves the global registration by combining a graph-based technique. They also propose an error function to reduce the number of iterations. Ligang and Yongjuan present a high resolution image mosaicing method for (small) document images that are captured by a camera [16]. They employ nearest-neighbor based skew rectification to locate the horizontal vanishing point of the text plane. Then multiple overlapping blocks are processed to compute the local orientation of a vertical character stroke. The technique uses local alignment constraints as the global alignment model to eliminate the error accumulation effectively.

```
Input:  V, the video stream;
Output:  M, the video mosaic;
Locals:  M_H(t), the horizontal mosaic image at frame t
         M_V(r), the vertical mosaic image at row r
         F, the set of oriented frames
         I, the set of keyframes
         r, the row count of the mosaic image

1.   F = autoOrientFrames(V)
2.   I = computeKeyframes(F)
3.   r = 0
4.   M_H(0) = I(0)
5.   for each keyframe t in I from 1 to end
6.       I(t) = pickNextKeyframe()
7.       M_H(t) = computeHorizontalMosaic(M_H(t-1), I(t))
8.       if M_H(t) = M_H(t-1)
         /* if empty horizontal merge */
9.          M_V(r) = M_H(t)
10.         r = r + 1
11.         M_H(t) = I(t)
12.      endif
13.  endfor
14.  for each row k from 1 to r
     /* skip first row image and up to last one */
15.      M_V(k) = computeVerticalMosaic(M_V(k), M_V(k-1))
16.  endfor
     /* copy last vertical merge to final mosaic image */
17.  M = M_V(r)
```

Figure 2. The pseudo-code of the video mosaicing algorithm that we employ for historical document images. Lines 1 and 2 correspond to the preprocessing phase during camera motion, and lines 3–17 correspond to the scanning phase to produce the final video mosaic.

Liang et al. [17] propose an effective image restoration process for the mosaicing approach without restricting the camera position. The computed projectivity of two consecutive document images with an overlapping area of 10% is used to estimate the perspective distortions. The authors also propose a two-fold document mosaic composition technique including rectification and registration phases [18]. First, perspective distortion and rotation are removed from images using the texture flow information. Next, the translation and scaling are resolved by a Hough transform-like voting method. In [19], a connected component based document image mosaicing method is proposed, in which each connected component is described using the angular radial transform. The use of connected components guarantees a stable localization, and hence a successful correspondence matching even in the presence of multiple similar regions is achieved. The performance is evaluated against viewpoint, illumination, and scale variations.

Nakao et al. [8] present an input unit for personal computers, where the user normally operates on a camera and s/he can takes partial images by moving the mouse over the photograph or document. Then, these (sub)images are merged by mosaicing scheme based on a block matching method to fasten image registration. Hannuksela et al. [20] propose an interactive document image scanning scheme with mobile phones. Their mosaicing technique is two fold: first online camera motion is estimated and applied to mobile camera. Then these patches are used for automatic image stitching.

3. Video mosaicing process

The proposed video mosaicing technique includes two phases. The preprocessing phase includes auto-orienting frames and generating keyframes operations while capturing motion through camera. The scanning phase first

merges two adjacent, and sufficiently overlapping, keyframes horizontally to form row mosaics, and then merges these row mosaics to create the final video mosaic image. The pseudo-code of the video mosaicing algorithm is listed in Figure 2. The first two steps, i.e. lines 1 and 2, correspond to the preprocessing phase in which the captured frames are oriented automatically by the help of the optical flow and keyframes are generated to guarantee a sufficient overlapping region in between. The scanning phase includes the operations listed between lines 3 and 17. $M_H(t-1)$, $I(t)$, and $M_H(t)$ have the same height during the horizontal scan as a direct consequence of $computeHorizontalMosaic(I_L, I_R)$ function at line 7. During the horizontal scan, if an empty merge is encountered, as checked in line 8, this means that $I(t)$ is in fact the first frame of the next row; hence the row mosaic is formed in $M_H(t)$. The mosaic of each row is then merged to form the final video mosaic.

3.1. Preprocessing phase

Auto-orientation of frames in the preprocessing phase is based on the computation of the optical flow in video. The optical flow can be seen as the velocity of the scene at each pixel. We employed the well-known KLT feature tracking algorithm [10, 11]. In general, the major drawback of using optical flow is that the projected motion of an object depends on its distance to the camera. In our case, since the zoom level and the distance of the camera can be safely assumed to be the same throughout the video, that drawback does not affect auto-orienting the frames (see line 1 in Figure 2, $F = autoOrientFrames(V)$).

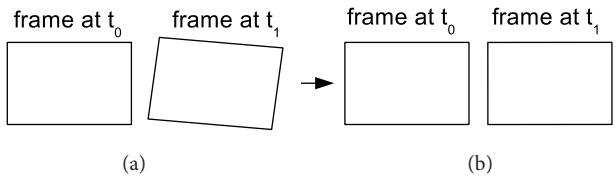


Figure 3. Auto-orientation scheme. (a) Based on the optical flow calculations at time t_0, the orientation of the frame at time t_1 is checked; if the threshold is exceeded, (b) then its orientation is corrected.

Figure 3 shows the auto-orientation process. Using the optical flow calculations in the frame at time t_0, the angle α between the flow vector and $\overrightarrow{e_x}$, where $\overrightarrow{e_x}$ is the unit vector on the x-axis, is calculated based on the features/points in KLT. A majority voting scheme is used to estimate the flow vector of the frame. If $\alpha \geq T_O$ for the frame at time t_1, where the threshold T_O is experimentally set to 5%, a new keyframe is created not to lose data at the boundaries. Then the frame at time t_1 is rotated back α degrees. By the help of this tracking, the camera motion from the end of a row to the beginning of a new row can also be detected, and no keyframes are generated during this motion.

One of the most important steps in our video mosaicing scheme is selecting an appropriate keyframe to guarantee a sufficient overlapping region (see line 2 in Figure 2, $I = computeKeyframes(F)$). Having placed appropriate keyframes in a list in this phase, the $pickNextKeyFrame()$ function (at line 6 in Figure 2) then processes each keyframe in this list. Let w_f denote the width of the frames captured when the camera is moving at velocity $\overrightarrow{v_c}$ on the x-axis. Due to the auto-orientation scheme, it can be assumed that $\overrightarrow{v_c}$ is constant and $\overrightarrow{v_c} \parallel \overrightarrow{e_x}$. Let $\overrightarrow{d_x}$ denote the pixel distance on the x-axis; it can be written such that $\overrightarrow{d_x} = \overrightarrow{v_c} \cdot \Delta t$. Since the frames-per-second value is known and letting Δf denote the number of frames between the previous keyframe and the current frame, it can be written as $\overrightarrow{d_x} = \overrightarrow{v_c} \cdot \Delta f$.

The scanning phase relies on an overlapping region between two consecutive keyframes, and the percentage

of the overlapping region is required to be at least 20%. The keyframe selection scheme guarantees 20% overlap in terms of w_f during the horizontal scan. Hence, this condition can be expressed as $\overrightarrow{d_x} \geq w_f \cdot 20\%$, and similarly $\overrightarrow{v_c} \cdot \Delta f \geq \frac{w_f}{5}$. This condition is checked at each frame based on the estimated value of $\overrightarrow{v_c}$ by the optical flow, which is tracked in the auto-orientation scheme a priori. When the boundary value is reached in this inequality, a new keyframe is generated having at least 20% overlapping region.

3.2. Scanning phase

The processing in the scanning phase of our video mosaicing technique has two main steps: horizontal scan and vertical scan. The former step is used to process the keyframes horizontally to form the mosaic image at each row. Each consecutive keyframe pair is merged by processing the overlapping region between them. The latter step is employed to merge the rowwise mosaic images to create the final video mosaic image. An image is defined as a function from R^2 to R, such that $f : [a, b] \times [c, d] \rightarrow [0, 1]$. Hence, an image pixel $I(i, j)$ denotes the intensity value at the location (i, j) in image I. No binarization is required since the pixel existence (i.e. $I(i, j) \neq 0$) is important for the similarity computations in merging.

Our horizontal scanning step $computeHorizontalMosaic(I_L, I_R)$ (at line 7 in Figure 2) merges two consecutive images having a sufficient overlapping region. Let I_L and I_R be two consecutive (left and right, respectively) images, and assume the image I_L has width m and height n; hence the first pixel is at $I_L(0, 0)$ and the last pixel is at $I_L(m - 1, n - 1)$. The same assumption holds for I_R. The width of I_L and I_R might be different but since the same zoom level is assumed during camera motion and the frames are aligned automatically, the height of them are equal.

The algorithm starts from $I_L(m - 1, n - 1)$ and $I_R(0, n - 1)$, and then compares the similarity at each corresponding column on the left and on the right. While computing column similarity the distance vector is used. I_L is advanced to the left and I_R is advanced to the right at each iteration of scanning to discover the overlapping region between them. At each time when an overlapping column is found between I_L and I_R, the overlapping region between them is extended with this new overlapping column. The lack of an overlapping region is the stopping case of the horizontal scanning, since this situation signifies the end of the keyframes in the same row. At the end, the current overlapping region O_R^H is used to merge I_L and I_R. The merged image $M_H(t)$ at time t is created as $M_H(t) = \bigcup((I_L - O_R^H), I_R)$.

Our vertical scanning step $computeVerticalMosaic(I_U, I_D)$ (at line 15 in Figure 2) merges two consecutive row images that are computed as the output of the horizontal mosaicing algorithm. This vertical merge operation between two rowwise mosaic images is similar to the horizontal mosaic computation with the difference that the dimensions of I_U and I_D can be different. Since the distance vector that we use for similarity computation is scale-invariant, these different sizes would not be a problem as long as the vector is normalized with respect to the maximum distance value. The same steps are followed to discover the overlapping region between I_U and I_D, and at each time when an overlapping row is found, the current overlapping region O_R^V is extended. When the scanning ends, the merged image $M_V(t)$ at time t is computed as $M_V(t) = \bigcup((I_U - O_R^V), I_D)$.

3.3. Similarity computation with distance vector

During horizontal and vertical scanning steps, the row similarity and column similarity values are computed with respect to the distance vector, respectively. The distance vector is a 1-dimensional histogram whose entries are the number of pixels existing in between the concentric circles that are centered at the center of mass c_m

with radius r, $2r$, ... The entries are normalized according to the distance of the farthest two pixels to achieve scale-invariance. Hence, the distance vector of a row/column requires the computation of the center of mass $c_m = (x_{c_m}, y_{c_m})$ a priori. It is computed as $x_{c_m} = \frac{\sum_i^n x_i}{n}$, $y_{c_m} = \frac{\sum_i^n y_i}{n}$, where n is the total number of pixels in a row/column such that $I(i,j) \neq 0$. (x_i, y_i) represents the coordinates of the pixel (i,j) in the 2D pixel matrix of the image I.

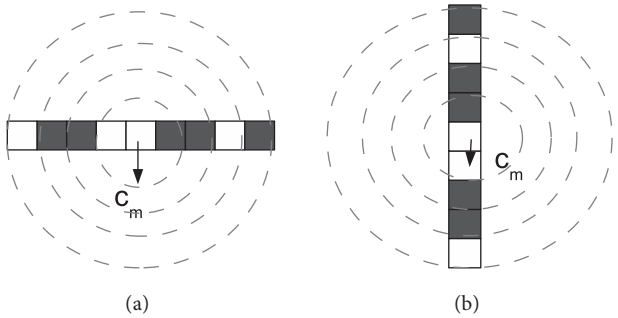


Figure 4. Distance vector computations (a) for a row during vertical scanning, (b) for a column during horizontal scanning. The center of mass c_m is shown with respect to the pixels having $I(i,j) \neq 0$ as the basis for the computation of the distance vector.

Figure 4 is shown to elaborate on distance vector computation for a sample row/column drawing with black pixels only. Based on the existence of the black pixels in the concentric circles, the distance vector D is $D = \{0, 1, 2, 1, 1\}$. After normalizing with respect to the maximum distance, which is 2 here, to guarantee scale-invariance, it can be written as $D = \{0, 0.5, 1, 0.5, 0.5\}$.

The $computeHorizontalMosaic(I_L, I_R)$ and $computeVerticalMosaic(I_U, I_D)$ (at lines 7 and 15 in Figure 2, respectively) operations employ distance vectors to compute similarity in discovering overlapping regions between two images. The following sets of operations are followed to conclude whether two rows/columns A and B are overlapping or not. First, c_m^A and c_m^B are computed as the center of mass of rows/columns A and B, respectively. Second, D_A and D_B are computed as the distance vector of rows/columns A and B based on c_m^A and c_m^B, respectively. In the third step, S_{D_A,D_B} is computed. If $S_{D_A,D_B} \geq T_D$, then it is found that the rows/columns A and B are overlapping; otherwise they are nonoverlapping. The similarity threshold T_D is experimentally set to 90%.

3.4. Video mosaicing: an example

A sample historical Ottoman document, namely a *kushan*, an Ottoman land-deed, is shown in Figure 5 to elaborate on the scanning phase of our video mosaicing process. In the preprocessing phase, the camera motion is tracked and based on the optical flow computations the captured frames are oriented. Then the keyframes are extracted with sufficient overlapping regions. Rowwise mosaic images are also shown to demonstrate the intermediate steps. Three keyframes for the 1st row and the two overlapping regions among these three frames are given with dashed lines, as an example. The result of the intermediate horizontal scan to form the rowwise mosaic image of the 1st row is shown. Respectively, the intermediate steps for the 2nd, 3rd, and 4th rows are presented. The final mosaic image is given in Figure 6 to illustrate the rowwise vertical scan step. Four consecutive rowwise mosaic images and the overlapping regions between them are shown as the result of the

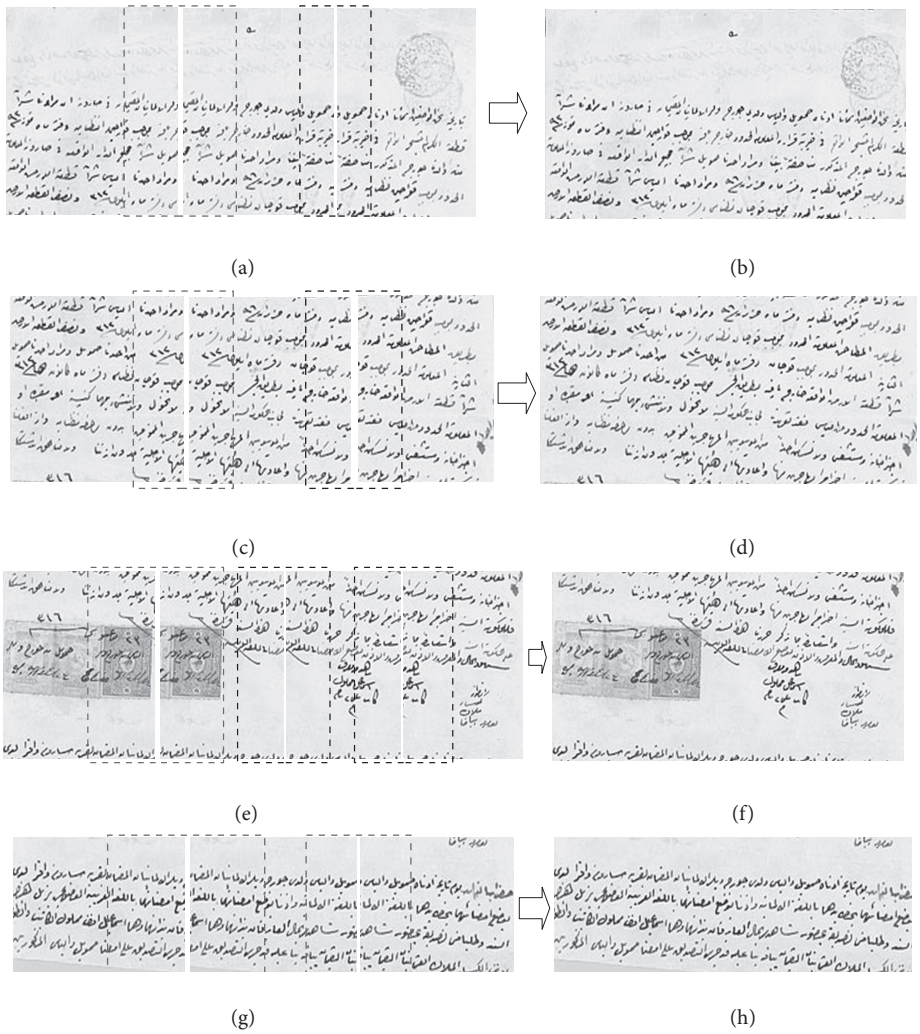
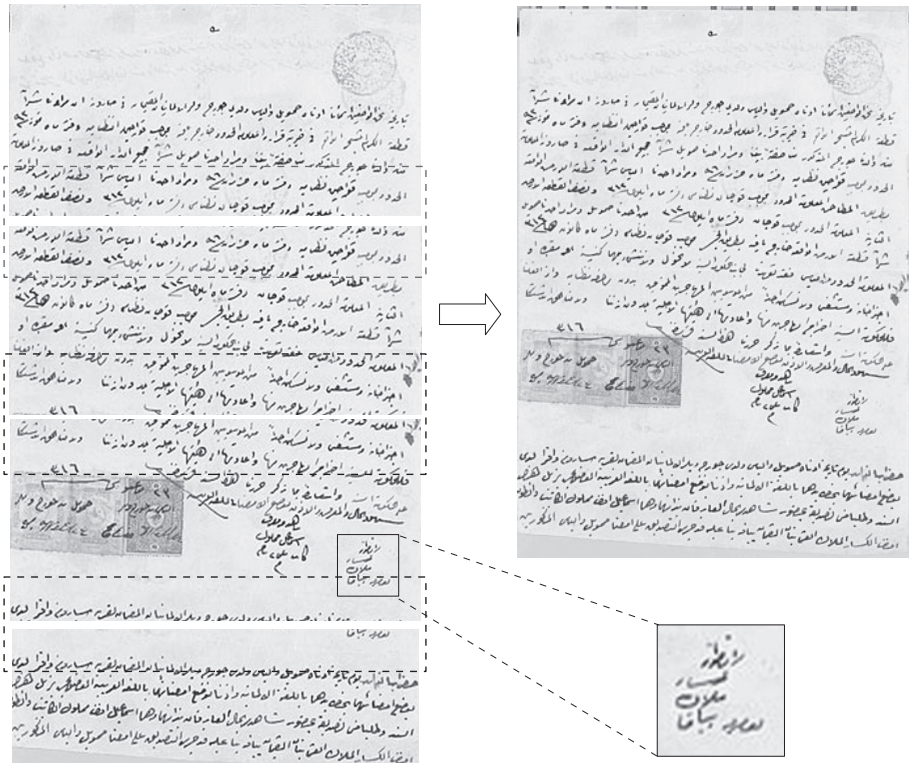


Figure 5. A sample historical document, namely a kushan, to illustrate the horizontal scanning phase of the video mosaicing algorithm with intermediate steps. (a) Three frames of the 1st row and two overlapping regions with dashed lines, (b) rowwise mosaic image of the 1st row. (c) Three frames of the 2nd row and two overlapping regions with dashed lines, (d) rowwise mosaic image of the 2nd row. (e) Four frames of the 3rd row and three overlapping regions with dashed lines, (f) rowwise mosaic image of the 3rd row. (g) Three frames of the 4th row and two overlapping regions with dashed lines, (h) rowwise mosaic image of the 4th row. (historical document source: http://www.telaviv-fever.com/index.php/2009/05/a-kushan-an-ottoman-land-deed/)

vertical scan process. A rectangular region is enlarged 2 times and shown to clarify the visualization of the final mosaic image.

4. Performance experiments

We have conducted an experimental study to evaluate the performance of our video mosaicing approach on historical Ottoman documents. It is worth noting here that the Ottoman script is a connected script based on the Arabic alphabet. A typical word in an Ottoman document consists of compounded letters similar



(a) (b)

Figure 6. A sample historical document, namely a kushan, to illustrate the video mosaicing algorithm at rowwise vertical scanning step. (a) 4 rowwise mosaic images and the overlapping regions with dashed lines, (b) full video mosaic image, and a rectangular region enlarged 2 times to ease visualization of the mosaic quality. (historical document source: http://www.telaviv-fever.com/index.php/2009/05/a-kushan-an-ottoman-land-deed/)

to handwritten text. There are also figures, drawings, and handwritten notes in various orientations in a typical document. Therefore, an ordinary textual image compression scheme for documents containing isolated characters cannot encode Ottoman documents.

We formed an Ottoman document image dataset having various scripts and drawings to evaluate the expressiveness of the proposed approach on documents of various types. Each Ottoman document image is captured by hand-held cameras as a video by a group of students in the Turkish literature department. The video capture operation traces the document in row major order, the zoom level is constant, and the frames-per-second value is 24. Each captured frame has 1280×960 resolution. The average number of frames is calculated with respect to these video data for each document. The average number of keyframes is based on the preprocessing phase of the video mosaicing approach, and similarly the video mosaic resolution is given in pixels on average. The OpenCV library is used for the preliminaries in the technique such as KLT-based optical flow and pixel level operations. The Table provides detailed information on these Ottoman documents.

The use of keyframes for video mosaic generation yields 12% memory space requirements as opposed to processing frames, as shown in Figure 7. As the basis, we considered processing only frames without any preprocessing steps. For processing keyframes in video mosaicing, the memory requirements of the preprocessing steps are included. The main reason in this gain is the fact that the average number of keyframes is very small

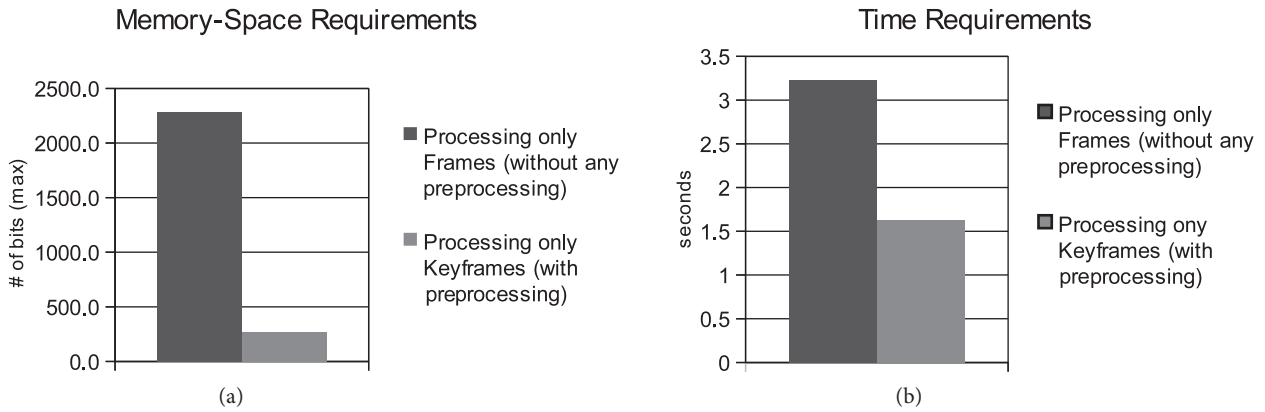


Figure 7. Performance experiments to evaluate the (a) memory-space, and (b) time requirements of processing only frames without any preprocessing versus processing only keyframes with preprocessing.

Table . Detailed information on the Ottoman document images in our dataset. The values are given on average.

Ottoman image dataset details	
Video duration (s)	3.21
# of Frames	77.6
# of Keyframes	9.2
Video mosaic resolution (pixels)	4444.9 x 5753.4

when compared to the average number of frames in the captured videos. Even though a preprocessing step is included in our video mosaicing approach, this has a limited effect in terms of space. Similarly, the total processing time is also reduced 49% in keyframe processing. The numbers are taken from a typical PC with an i5 processor.

A pixel-level comparison is also performed to see the effect of the video mosaicing approach. As the basis, we considered processing only frames without any preprocessing steps, and assumed no pixel loss during merging. Figure 8 shows the evaluation on the percentage of the overlapping region. The results show that the most appropriate value, i.e. the percentage having minimum pixel loss, is 20% among the values that we compared. The pixel loss in processing with keyframes is basically due to the preprocessing phase, or the similarity computations during the scanning process. The optical flow auto-orientation threshold T_O is set to 5%, and the distance vector row/column similarity threshold T_D is set to 90%.

Another evaluation is performed to clarify the effect of the optical flow auto-orientation threshold T_O. Similarly, the basis is processing only frames without any preprocessing steps, and we manually orient the frames to produce the video mosaic. Figure 9 shows the evaluation on this threshold. The evaluation is two-fold. First, the pixel loss is considered. $T_O = 3$ has slightly less pixel loss than $T_O = 5$. Then the average number of keyframes is compared, and here $T_O = 3$ produces more keyframes, as expected, which will increase both time and memory-space requirements. Based on these evaluations, the most appropriate value is 5%. In these evaluations, the percentage of the overlapping region is set to 20%, and the distance vector row/column similarity threshold T_D is set to 90%.

The threshold T_D for row/column similarity computations based on the distance vector is also evaluated. Here two boundary cases can be considered. One is requiring exact match for the rows/column during

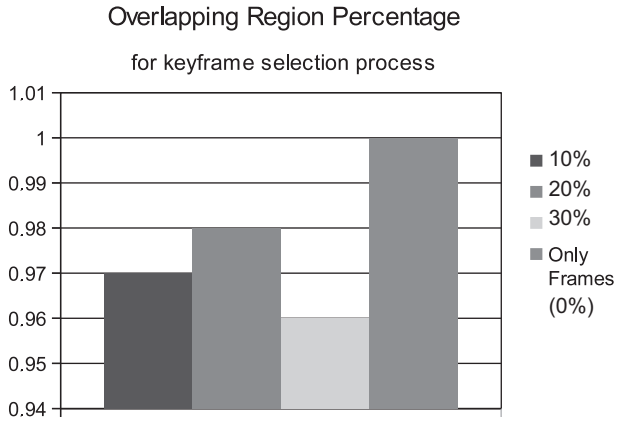


Figure 8. Performance experiments to evaluate the percentage of the overlapping region.

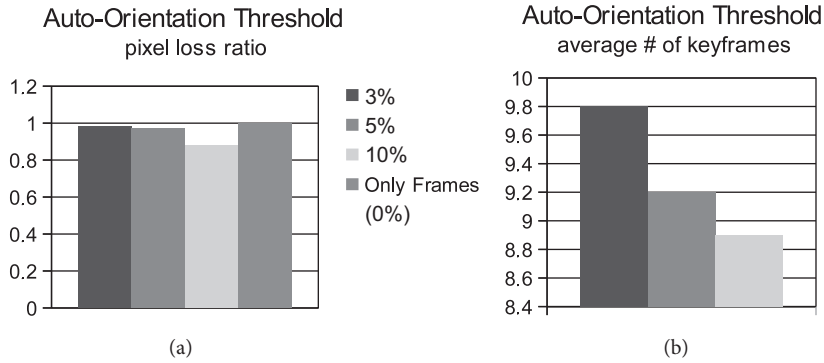


Figure 9. Performance experiments to evaluate the optical flow auto-orientation threshold T_O with respect to (a) pixel loss, (b) average number of keyframes.

overlapping region processing. Theoretically this will produce zero pixel loss; however, it will clearly exclude using keyframes. Moreover, a limited amount of salt-and-pepper type noise is also skipped, which might cause errors in merging frames. Another boundary situation is setting lower values to this similarity threshold (e.g., 75%). This has the possibility of merging nonoverlapping parts of the keyframes. Hence we set a typical value to T_D, which is 90%, based on manually evaluating the resulting video mosaics. In these evaluations, the optical flow auto-orientation threshold T_O is set to 5%, and the percentage of the overlapping region is set to 20%.

A comparison between using a single video and using a set of single images is valuable since the latter will reduce the effort of preprocessing steps. In fact, in that case the eliminated preprocessing effort is transferred to the user during input acquisition, since capturing an adequate number of single images that have sufficient overlapping regions is more tedious and erroneous when compared to auto-extracting keyframes from a single video capture. Having captured a set of appropriate single images, even though the quality of the final mosaic would be higher in terms of image resolution and reliability, the performance would be equivalent since the set of keyframes is also a set of images. The most important gain in using videos instead of a set of single images is to facilitate the user's tasks during input acquisition.

5. Conclusion

A video mosaicing technique for historical Ottoman documents is presented to produce a high resolution video mosaic image from a captured video for later use in browsing and/or content-based retrieval systems. The video mosaicing technique scans the captured frames first horizontally to create row mosaics and then vertically to create the final mosaic. This scanning phase relies on the auto-oriented keyframes that are generated in the preprocessing phase based on mainly tracking the optical flow. The assumptions include constant zoom during camera motion. Even though the average number of keyframes in the experimental study is small to evaluate the scalability of the technique, we plan to analyze the error accumulation when the number of keyframes is higher in future work.

One of the main features of the technique is adapting the distance vector for row and column similarity computations. The results show that this distance vector is reasonable in historical Ottoman documents in terms of similarity computations. Other mosaicing techniques can also be used with browsing and/or content-based retrieval systems; however, it has been shown through performance experiments that we provide an efficient and effective approach for historical Ottoman documents by using keyframes during processing and distance vector for similarity computations.

There are a couple of parameters that are set to appropriate values experimentally in our approach. In future work, we are planning to process the steps in the video mosaicing approach automatically by designing a solution to enable the system to learn these parameters while capturing the video.

References

[1] Szielski R. Video mosaics for virtual environments. IEEE Comput Graph 1996; 16: 22-30.

[2] Doermann D, Liang J, Li H. Progress in camera-based document image analysis. In: 2003 International Conference on Document Analysis and Recognition; 3–6 August 2003; Edinburgh, Scotland, UK: IEEE. pp. 606-617.

[3] Abraham R, Simon P. Review on mosaicing techniques in image processing. In: 2013 International Conference on Advanced Computing and Communication Technologies; 6–7 April 2013; Rohtak, India: IEEE. pp. 63-68.

[4] Rav-Acha A, Pritch Y, Lischinski D, Peleg S. Dynamosaicing: mosaicing of dynamic scenes. IEEE T Pattern Anal 2007; 29: 1789-1801.

[5] Kim DW, Hong KS. Real-time mosaic using sequential graph. J Electron Imaging 2006; 15: 1-13.

[6] Yalnız İZ, Altıngövde İS, Güdükbay U, Ulusoy Ö. Ottoman archives explorer: A retrieval system for digital ottoman archives. ACM J Comput Cult Herit 2009; 2: 1-20.

[7] Şaykol E, Sinop AK, Güdükbay U, Ulusoy Ö, Çetin E. Content-based retrieval of historical Ottoman documents stored as textual images. IEEE T Image Process 2004; 13: 314-325.

[8] Nakao T, Kashitani A, Kaneyoshi A. Scanning a document with a small camera attached to a mouse. In: IEEE 1998 Workshop on Applications of Computer Vision; 19–21 October 1998; New Jersey, USA: IEEE. pp. 63-68.

[9] Whichello AP, Yan H. Document image mosaicing. In: 1998 International Conference on Pattern Recognition; 16–20 August 1998; Brisbane, Australia: IEEE. pp. 1081-1083.

[10] Lucas BD, Kanade T. An iterative image registration technique with an application to stereo vision. In: 1981 International Joint Conference on Artificial Intelligence; Vancouver, Canada: William Kaufmann. pp. 674-679.

[11] Shi J, Tomasi C. Good features to track. In: IEEE 1994 Conference on Computer Vision and Pattern Recognition; 21–23 June 1994; Seattle, WA, USA: IEEE. pp. 593-600.

[12] Şaykol E, Güdükbay U, Ulusoy Ö. A histogram-based approach for object-based query-by-shape-and-color in multimedia databases. Image Vision Comput 2005; 23: 1170-1180.

[13] Zappala AR, Gee AH, Taylor MJ. Document mosaicing. Image Vision Comput 1999; 17: 585-595.

[14] Shivakumara P, Kumar GH, Guru DS, Nagabhushan P. Sliding window based approach for document image mosaicing. Image Vision Comput 2006; 24: 94-100.

[15] Marzotto R, Fusiello A, Murino V. High resolution video mosaicing with global alignment. In: IEEE 2004 Conference on Computer Vision and Pattern Recognition; 27 June–2 July 2004; Washington, DC, USA: IEEE. pp. 692-698.

[16] Ligang M, Yongjuan Y. Automatic document image mosaicing algorithm with hand-held camera. In: 2011 International Conference on Intelligent Control and Information Processing, 25–28 July 2011; Harbin, China: IEEE. pp. 1094-1097.

[17] Liang J, DeMenthon D, Doermann D. Camera-based document image mosaicing. In: 2006 International Conference on Pattern Recognition; 20–24 August 2006; Hong Kong: IEEE. pp. 476-479.

[18] Liang J, DeMenthon D, Doermann D. Mosaicing of camera-captured document images. Comput Vis Image Und 2009; 113: 572-579.

[19] Kasar T, Ramakrishnan AG. CCD: Connected component descriptor for robust mosaicing of camera-captured document images. In: IAPR 2008 International Workshop on Document Analysis Systems; 16–19 September 2008; Nara, Japan: IAPR. pp. 480-486.

[20] Hannuksela J, Sangi P, Heikkila J, Liu X, Doermann D. Document image mosaicing with mobile phones. In: 2007 International Conference on Image Analysis and Processing; 10–14 September 2007; Modena, Italy: IEEE. pp. 575-582.