# Object Mosaicking: Reconstruction of Moving Objects Captured Through a Limited View*

Richard Caywood, Justin Edwards and Ramazan S. Aygun
Computer Science Department
University of Alabama in Huntsville
Huntsville, AL, USA
{rcc0005,jge0002,aygunr}@uah.edu

***Abstract*—There had been significant research on background mosaic or sprite generation in the past. In some application domains such as surveillance, the complete view of an object might be of interest rather than the background scene. In this paper, we present object mosaicking for reconstruction of a moving object from its partial views due to limited view or object occlusion. Object mosaicking may play important role in identification and tracking of objects and face recognition. Our algorithm has three steps: automatic rectangular boundary detection, elimination of static temporal texture, and application of mosaic generation algorithms with varying blending methods. We have applied our object mosaicking on three sets of video objects and obtained promising results for object mosaicking.**

***Keywords—mosaic generation; image registration***

## I. INTRODUCTION

The research on computer vision systems has increased more and more in the last decade as the technology for capturing and storing video data have become cheaper and easier with the proliferation of user friendly, portable, and smart hand-held devices. Today, even low-end smart phones can capture high quality video at the push of a button. This has led to production of enormous video content as people carry their smart phones with them. The videos have been accessible to public as social networks such as Facebook [1] and Twitter [2], and news sites like CNN [3] enabled their users to upload and share videos. However, indexing videos based on content is still a challenging problem due to detection and recognition of objects. Partial view or incomplete appearance of an object makes detection and recognition even harder. In this paper, we look into the construction of moving objects from partial views.

In many videos, the objects of interest are moving or the camera is moving or both. Since the camera has a limited field-of-view, as the camera moves new scenes from the world are captured that were not visible in the previous frames of a video. The frames of a video may not necessarily capture a complete scene or a complete object due to the field-of-view of the camera or object occlusions. An important area in computer vision is mosaic generation. Basically, a background mosaic or a background sprite is the complete static view of an environment that cannot be captured in a single frame of a video. Image stitching, background extraction, or video panorama generation research has also overlaps with mosaic generation research. Mosaic generation has also been studied as sprite generation in MPEG-4 standard [4] for video compression purposes

The significant majority of mosaic generation research has been performed for the background scene [5-8]. Now, suppose that instead of generating an entire panorama of a scene, a user is interested in only a certain section of the scene such as an object of importance that the user wishes to focus on. There are videos where a single object in a video can only be captured partially due to object occlusion. Object might be behind or in between other objects such as trees. It is possible that at each frame a partial view of the object might be captured. For example, consider a surveillance camera in a convenience store. The purpose of this camera is to make sure that anyone who tries to rob the store will end up on camera. So the object of importance in this example would be the robber's face. However, there might not be any single video frame that contains the robber's entire face.

In this paper, we propose a method to reconstruct the object from its partial views. We call such mosaic generation as “Object Mosaicking” since the mosaic is generated for the object not for the background. Therefore, object mosaicking is the process of recreating a single object from the parts of the object present in all the frames of the video in question. Using object mosaicking on the above convenience store example, a user would reconstruct the robber's face from the different parts of his or her face seen in the frames of the surveillance video.

Wang et al. proposed multimodal temporal panorama for moving vehicle detection and reconstruction [9]. They utilize visual, audio, and motion data to recreate the full image of cars passing by a PTZ camera. Their system requires manual setting of the boundaries of the region of interest. If the view of the camera is not parallel to the ground (i.e., the camera is viewing the road from an angle), manual input is needed to identify the slant of road with respect to the camera pose. The authors do not clearly mention about the blending technique they use. It is not clear if the vehicle needs to have a constant speed or not. We wonder if the vehicle can be elongated if the car is moving slowly.

In this paper, we propose an object mosaicking method

This material is based upon work supported by the National Science Foundation under Grant No. 0812307. REU students were funded with REU Supplement by the National Science Foundation Grant. No. 1240395.

based on our previous sprite generation technique used for background mosaic generation. Our proposed system detects the borders of the occlusion or the visible scene area. Our system does not require manual settings of the boundary and the moving object does not need to be parallel to the ground because we use affine motion model for global motion estimation. The object does not need to move at a steady speed. As long as the global motion can be estimated, the object may move at any irregular speed. In addition, the regions in the scene which may deteriorate the mosaicking are eliminated by ignoring those regions during motion estimation and blending. We apply traditional mosaic generation based on averaging and our sprite generation technique based on fusion of mosaics.

This paper is organized as follows. The following section provided information about mosaic generation. Section 3 explains our method for object mosaicking including automatic border detection and constant region elimination. Section 4 provides experimental results and discussion. The last section concludes our paper.

## II. BACKGROUND

Mosaic generation is composed of three major steps: global motion estimation (GME), warping, and blending. GME identifies how much the change is between two frames. The motion parameters that are determined by GME are used to find the proper coordinates on the mosaic where the next frame should be placed. Warping step aligns corresponding images on top of each other based on their content. Blending phase determines the output pixel value for the overlapping areas of images when they are mapped onto the mosaic. Warping and blending are used to make sure the mosaic comes out as clean and artifact-free as possible.

GME is the most computation extensive part of mosaic generation. Therefore, usually 2D motion models are utilized for the motion parameter estimation in real-time applications. 2D GME corresponds to the estimation of 8 motion parameters. Equation 1 provides how pixel coordinates of one image is mapped to the pixel coordinates of another image:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & t_x \\ a_2 & a_3 & t_y \\ a_4 & a_5 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (1)$$

where $t_x$ and $t_y$ correspond to translational parameters. Given images $I$ and $I'$, a pixel at $I(x,y)$ is mapped to $I'(x',y')$ using the 3x3 matrix provided in (1). If zoom operation is considered, $a_1=a_2=a_4=a_5=0$, and $a_0$ and $a_3$ are estimated. Parameters $a_0$, $a_1$, $a_2$, and $a_3$ are related to rotation angle of $\theta$ when $a_0=a_3=cos\theta$ and $-a_1=a_2=sin\theta$. Parameters $a_4$ and $a_5$ correspond to perspective motion. For affine motion, $a_4$ and $a_5$ are set to 0. Here, for affine motion, GME is estimated in a hierarchical fashion. At the lowest level, a GME is estimated and for the next level translational parameters are multiplied by 2 [11]. Then Levenberg-Marquardt optimization is applied at each level. The translational motion is estimated by direct motion estimation at the given resolution of the image.

After GME, the blending plays critical role on the accuracy of the mosaic. We use the sprite generator tool that is developed in [10]. This sprite generator tool may input a variety of video formats or an image sequence to generate the sprite. Let P be the motion parameter matrix to map a frame to the mosaic, M:

$$\begin{bmatrix} i' \\ j' \\ 1 \end{bmatrix} = P \times \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} \quad (2)$$

Typically, a blending algorithm, Blend, is used for determining what the final value is: $\mathbf{M}(i',j')=Blend(\mathbf{M},i',j',\mathbf{f_k},i,j)$. In this expression, $\mathbf{f_k}$ is a frame, $\mathbf{M}$ represents the mosaic, and the pixel at $\mathbf{f_k}(i,j)$ is mapped $\mathbf{M}(i',j')$ on the mosaic.

Our sprite generator tool provides basic averaging as a blending technique as well as the novel the sprite fusion [10]. Sprite fusion is performed by merging the results of two mosaics: assertive and conservative mosaic. In assertive, a pixel from a new frame overwrites the existing pixel value. In conservative mosaic, a pixel from a new frame is written to the mosaic unless there is no pixel value written to that position before. As an artifact, assertive mosaic keeps the last frame on the mosaic; on the hand, conservative mosaic keeps the first frame on the mosaic. As long as there is a frame in the video that does not intersect with the first frame in the sequence and if the moving objects do not appear at the borders of a frame, sprite fusion can eliminate moving objects from the assertive or conservative mosaic given the motion parameters. We cleaned up, optimized, and improved the original sprite generator tool and then added new modules for this project.

## III. OBJECT MOSAICKING

In this section, we propose our object mosaicking method. As mentioned in the previous section, the first step of mosaic generation is the global motion estimation. The global motion estimation may represent the camera motion if the camera is moving and the (moving) objects in the video are small. One of the problems for background mosaic generation is the presence of large objects. If the moving object is very large (e.g., if it occupies a half of a frame), the background mosaic cannot be generated, because the motion does not correspond to the camera motion anymore. In our object mosaicking, we are going to get benefit from this limitation.

Object Mosaicking is the process of recreating a single object from the parts of the object present in all frames of the video in question. To be able to generate the object mosaic, we should be able to determine the object motion. One challenge is that the size of the object might be respectively small. Another challenge is that the object might be partially visible due to occlusion by other objects. If we are able to reduce the region to be processed about the size of the object, the mosaic to be generated will be equivalent to the mosaic of the object.

The steps of our method are provided in Fig. 1. The two major steps are automatic border detection and temporal texture elimination. Automatic border detection is used if the object is being captured from a structured but textured and limited view. The automatic border detection system finds automatically which areas of the picture contain “useless” data and then ignores those areas.

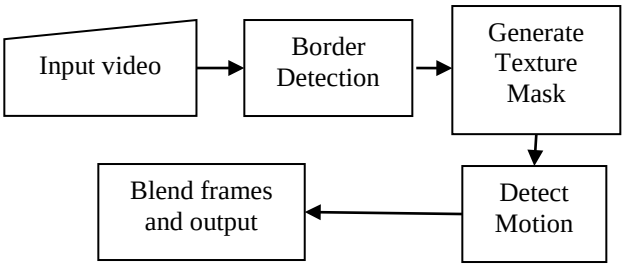



Fig. 1. Object mosaicking method

Temporal texture elimination step generates a texture map for each frame to develop a clear mosaic by eliminating areas that contain similar textures or colors which could cause inaccurate mosaic generation. Temporal texture elimination enables to focus on mostly the object area since the non-moving background is static.

Consider the following notation for this section:

| Notation | Explanation |
|---|---|
| $f_i$ | $i^{th}$ frame of the input video |
| $f_i[*,k]$ | the $k^{th}$ column of frame $f_i$ |
| $f_i[m,*]$ | the $m^{th}$ row of frame $f_i$ |
| $f_i^u[*,k]$ | the upper half of the $k^{th}$ column of frame $f_i$ |
| $f_i^b[*,k]$ | the bottom half of the $k^{th}$ column of frame $f_i$ |
| $f_i^l[m,*]$ | the left half of the $m^{th}$ row of frame $f_i$ |
| $f_i^r[m,*]$ | the right half of the $m^{th}$ row of frame $f_i$ |
| $W$ | the width of a frame |
| $H$ | the height of a frame |
| P | gap between two columns or rows |
| *Mode* | the most frequent value |
| *countIF* | returns the number of values that satisfies a condition in a set |
| $\mu$ | functioning returning the average value |

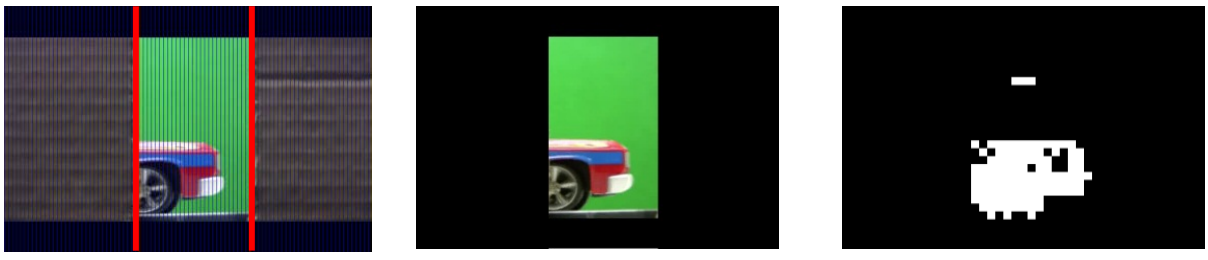
### A. *Automatic Border Detection*

Automatic Border Detection involves the determining the irrelevant object regions in the video, i.e., the parts where no movement exists, and telling our tool to ignore those parts. Algorithm 1 provides our algorithm for automatic border detection. Firstly, the left, right, top, and bottom borders are found for each frame. Rather than comparing consecutive columns or rows, rows and columns ρ pixels apart are evaluated to determine the change of average intensity. Since (top and bottom) or (left and right) sides of a frame may differ, each column and each row is divided into half. Since there could be many candidate borders, the algorithm determines the left-most (or top-most) and the right-most (or bottom-most) borders (marked red (re-edited to highlight the border) in Fig. 2 (a)). After determining the borders of a frame, the system finds the mode (or the most frequent border) of all borders from all frames. Fig. 2 (b) displays the frame after out-of-border area is eliminated.

**Algorithm 1:** Automatic Border Detection

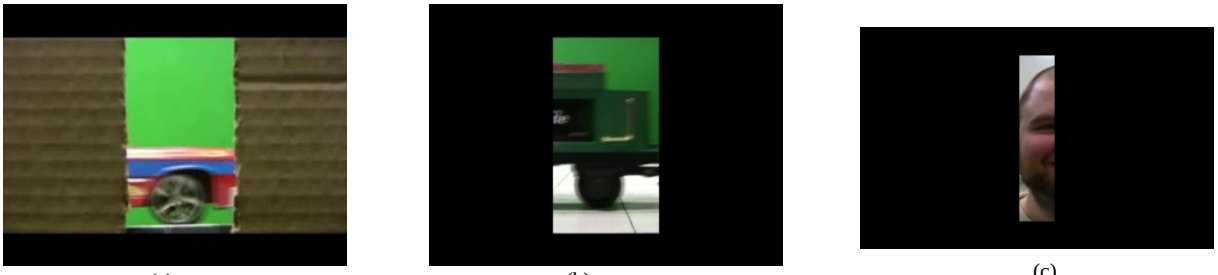
IN: A video with n frames
OUT: L (left), R (right), U (Top), B(bottom) borders
VARIABLES:
$\delta_i$: candidate vertical borders of $f_i$
$\delta_i^l$: left border of $f_i$; $\delta_i^r$: right border of $f_i$
$\lambda_i$: candidate horizontal borders of $f_i$
$\lambda_i^u$: top border of $f_i$; $\lambda_i^b$: bottom border of $f_i$

**begin**
 **for** i = 0 to n **do**
  **for** k=1 to (w/δ)-1 **do**
   **if** $|\mu(f_i^u[*,k]) - \mu(f_i^u[*,k-1])| \geq \tau$
    and $|\mu(f_i^b[*,k]) - \mu(f_i^b[*,k-1])| \geq \tau$ **then**
     $\delta_i \leftarrow \delta_i \cup \{k\}$
   **endif**
   $\delta_i^l = \min(\delta_i)+1$; // the left border
   $\delta_i^r = (\min(\delta_i)-1)$ if $(\delta_i^r+1) \notin \delta_i$ //the right border
  **endfor**
 **for** m = 1 to (h/ ρ)-1 **do**
  **if** $|\mu(f_i^l[m,*]) - \mu(f_i^l[m-1,])| \geq \tau$
   and $|\mu(f_i^u[m,*]) - \mu(f_i^u[m-1,*])| \geq \tau$ **then**
   $\lambda_i \leftarrow \lambda_i \cup \{m\}$
  **endif**
  $\lambda_i^u = \min(\lambda_i)+1$; // the top border
  $\lambda_i^b = (\min(\lambda_i)-1)$ if $(\lambda_i^b+1) \notin \lambda_i$ // the bottom border
 **endfor**
**endfor**
L=mode($\delta_i^l$); R=mode($\delta_i^r$); U=mode($\lambda_i^u$);
B=mode($\lambda_i^b$);
 where $0 \leq i < n$ // left, right, top, bottom borders
**end**



(a) (b) (c)

Fig. 2. Automatic border detection a) red borders indicate left and right borders, b) scene after border removal, and c) the texture mask for car example.



(a) (b) (c)

Fig. 3. Sample frames a) car, b) train with text, and c) face

### B. Static Temporal Texture Elimination

Temporal texture elimination involves scanning the parts of the video left after automatic border detection and generating a texture map that eliminates any homogenous parts of the remaining part of the video. The static temporal texture elimination algorithm is provided in Algorithm 2. The frame difference between consecutive frames is computed. The difference frame is split into macro-blocks. If the number of difference values that are equal to 0 is more than the half, the macro-block is considered as a texture block and not used for object mosaicking. Fig. 2 (c) displays the mask for texture mask elimination. The white regions correspond to object and the black regions correspond to the texture area.

**Algorithm 2:** Static Temporal Texture Elimination

ASSUME: macro-block size is 16x16.
IN: frames $f_i$ and $f_{i+1}$(size: h x w)
**begin**
 $\Delta f=|f_i-f_{i+1}|$ // difference of images
 Split into $\Delta f$ into macroblocks
 row= h/16; column=w/16
 **for** i = 1 to row **do**
  **for** j = 1 to column **do**
   **foreach** MB(i,j) **do**
    **if** countIF(MB(i,j),'=0')>((16*16)/2) **then**
     eliminate MB(i,j) // static texture
    **else** MB(i,j) belongs to object
    **endif**
  **endfor**
 **endfor**
**end**

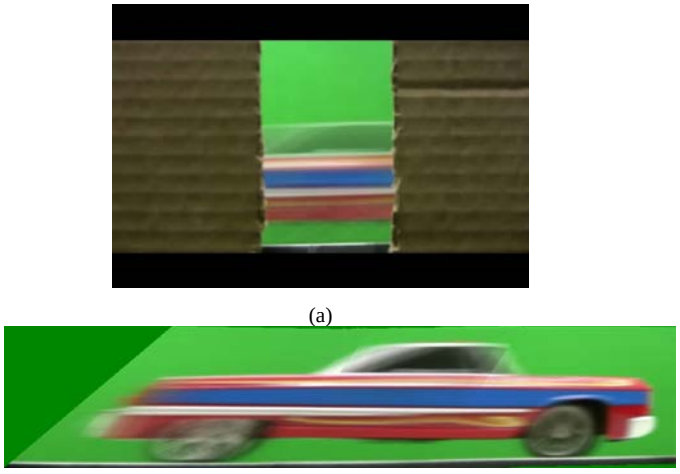
### C. Mosaic Generation for an Object

There are several factors that affect the quality of a mosaic. The GME is the first step of the mosaic generation. An error on GME will lead errors on warping and, hence, on the blending. GME is sensitive to a number of factors: the motion model that is used by the GME component; the varying depth in the scene; the resiliency of GME to static patterns; and other objects that do not conform to the global motion. The blending is also critical when GME introduces errors. Averaging technique blurs the mosaic; however, it is more resilient to errors in GME. Sprite fusion technique produces sharper mosaic; however, it is more sensitive to GME. Sprite fusion is not affected by slow moving objects or patterns in the scene or object. We use average, assertive, and conservative mosaics in our experiments.

## IV. Experiments

For the experiments we have chosen a set of 3 videos (Fig. 3). A toy car was chosen that has a variable continuous pattern along the side and complex motion with the wheels. A toy train example was chosen since it had text on the side of it. The final experiment was a face. This was chosen because of the complex nature of aligning parts of a face. In our test videos, the target object is partially visible since the target object is occluded. We have generated the videos in our lab. Although at first sight the green background may be considered as simple, it deteriorates the performance of the global motion estimation. If the majority of the background is static as in our example, GME usually yields almost no motion.

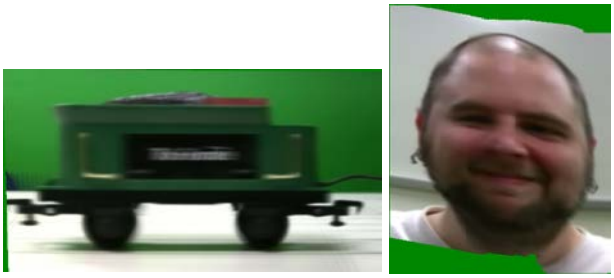
**Traditional mosaic generation.** Before providing the results of the proposed approach in this paper, we would like to present the results of traditional approaches. Fig. 4 (a) shows the results of traditional mosaics for the car example. In Fig. 4(a), we do not apply any pre-processing and get the blurred partial car with the presence of the borders. The major reason is that the original GME targets the global motion. In Fig. 4(b), we present the results after identifying the borders and just processing the area within the borders. The car image is still blurry. We are able to obtain a rough overview of the car but it is very blurry and does not contain all details of the car.



(a) (b)

Fig. 4. Traditional mosaics for the car example a) without border detection and b) with border detection.

**Experimental Results using Border Detection Without Texture Elimination.** The result for car sequence is provided in Fig. 4 (b). Fig. 5 provides the results for the train and the face examples using the areas within the border. The text on the body result of the train car is blurred and not readable. However, we were able to get somewhat acceptable mosaic for the face example. The reason is that after considering the areas within the borders, the face occupies the majority of the scene. Therefore, GME detects the motion of the face. The border detection made a significant improvement on the mosaic generation. However, the mosaics are still blurred significantly.



(a) (b)

Fig. 5. Traditional mosaics after detecting the borders and considering the area within the borders a) train example and b) face example.

**Experiments with the Border detection and Texture Elimination.**

*Car.* The three results of blending (averaging, assertive, and conservative) are close to each other with small errors in each one of them (Fig. 6). The average blending produced an image that is fairly blurry especially in the wheels where the complex motion existed (Fig. 6 (a)). The assertive blending created a more detailed object. The spokes in the wheels are easier to distinguish but the roof of the car was cut off early (Fig. 6 (b)). The conservative blending created a slightly elongated car but the wheels are very distinguishable (Fig. 6 (c)).

*Train.* For the train test the biggest thing was the readability of the text on the side. Due to the scaling of these images a lot of the details where lost. The average mosaic created a very readable text on the larger images (Fig. 7 (a)). The assertive mosaic produced text that was readable but it was slightly misaligned (Fig. 7 (b)). The conservative mosaic was very close to the assertive except it doubled the letters at the end of the last word (Figure 7 (c)).

For the face experiment the blending results for all except one method was almost perfect. Both the average (Fig. 8 (a)) and assertive (Fig. 8 (b)) blending produced nearly perfect results. The conservative (Fig. 8 (c)) blending produced misaligned results that can be attributed to an error in motion detection. For this test we have provided a graph of the PSNR (Peak Signal-to-Noise Ratio) values that represents the correctness of our mosaic (Fig. (9)). The dip in the left and right sides of the values can be due to the object not actually appearing in the frame at that moment. The average PSNR value is 28.66. The PSNR values are generated by regenerating the object regions using the estimated global motion and comparing the regenerated image to the original images used for generating the mosaic (by first computing the minimum squared error).

*Discussion.* One problem that we faced while preparing our experimental data is the automatic zooming of the camera. Since the target object was occluded by another object that was closer to the camera, automatic zooming tends to maintain focusing on the closer object not the target object. This led to blurring of the target object to be mosaicking even before any mosaicking is applied.

When we compare three blending techniques for mosaicking, the averaging technique in general provides an overall structure of the target object properly with some blurring. On the other hand, assertive and conservative mosaicking techniques provide sharp object mosaics, however, if the global motion is not estimated correctly, it causes discontinuity and line breakings. We believe that these three mosaicking techniques complete each other. If the global motion is estimated properly, assertive and conservative mosaic generation algorithms can outperform the averaging technique. If the global estimation is poor, averaging technique can be preferred to assertive and conservative mosaics.

(a) Average (b) Assertive (c) Conservative

Fig. 6. Object mosaics for car example.


(a) Average (b) Assertive (c) Assertive

Fig. 7. Object mosaics for car example.

(a) Average (b) Assertive (c) Conservative

Fig. 8. Object mosaics for face example.

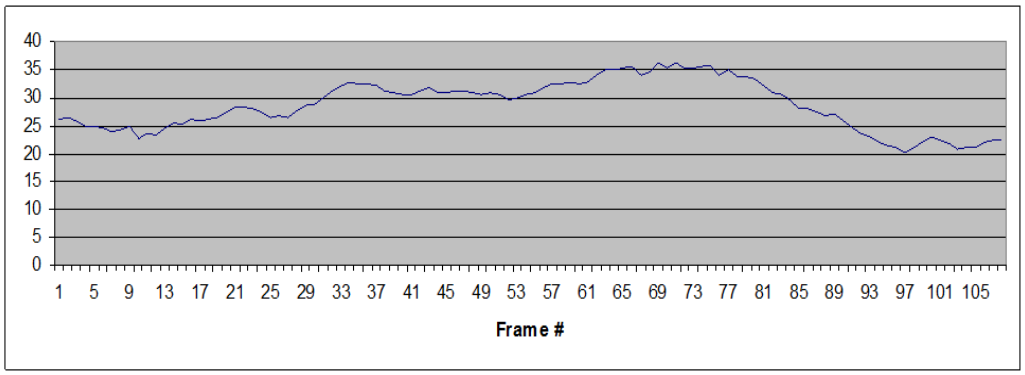



Fig. 9. PSNR values for the face experiment.

## V. CONCLUSION

In this paper, we proposed object mosaicking. Object mosaicking may help users to build a complete view of an object from partial views of an object. Two major steps are automatic border detection and temporal texture elimination. We have tested our algorithm on a different set of videos.

We were able to generate the objects at a satisfactory level. We have obtained very good results for generating the mosaic of a face. Our results for object mosaicking are promising. Especially, the mosaic for face was very good in terms of visual quality and PSNR values. While average blending produced blurred but acceptable mosaics, the assertive and conservative mosaics generated sharper but sometimes slightly misaligned mosaics.

We have obtained promising results for the future work. However, we had to conduct many experiments until we get the desired results. The parameters for our experiments are related to the threshold gap between rows/columns of the border selection, 2D motion models (affine, translational, pan-tilt-zoom), and starting and ending frames of a sequence. For future work, the most critical improvement is needed for the object motion estimation. In addition, we plan to test our system for outdoor environments and see the performance of object detection with various types of limited view or object occlusion. We plan to increase the number of videos to be tested. The view of the object may appear in any type of convex or concave region. We plan to extend our research for different types of view with extended border detection.

## REFERENCES


[1] www.facebook.com

[2] www.twitter.com

[3] www.cnn.com

[4] T. Sikora. The mpeg-4 video standard verification model, IEEE Trans. Circuits Syst. Video Technology 7 (1997) 19–31, 1997.

[5] J.-H. Lai; C.-C. Kao; S.-Y. Chien, "Super-resolution sprite with foreground removal," *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on* , vol., no., pp.1306-1309, June 28 2009-July 3 2009.

[6] M. Kunter, P. Krey, A. Krutz, and T. Sikora, "Extending H.264/AVC with a background sprite prediction mode," *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on* , vol., no., pp.2128-2131, 12-15 Oct. 2008.

[7] P. Parikh and C.V. Jawahar, , "Enhanced Video Mosaicing using Camera Motion Properties," *Motion and Video Computing, 2007. WMVC '07. IEEE Workshop on* , vol., no., pp.26, Feb. 2007

[8] H.-K. Cheung and W.-C. Siu, "Robust global motion estimation and novel updating strategy for sprite generation," *Image Processing, IET* , vol.1, no.1, pp.13-20, March 2007.

[9] T. Wang, Z. Zhu, and C.N. Taylor, "Multimodal Temporal Panorama for Moving Vehicle Detection and Reconstruction," *Multimedia (ISM), 2011 IEEE International Symposium on* , vol., no., pp.571-576, 5-7 Dec. 2011

[10] Y. Chen, A.A. Deshpande, and R.S. Aygun, “Sprite generation using sprite fusion.” ACM Trans. Multimedia Comput. Commun. Appl. 8, 2, Article 22 (May 2012), 24 pages.

[11] Y. Chen and R.S. Aygün. "Synthetic Video Generation for Evaluation of Sprite Generation." IJMDEM 1.2 (2010): 34-61. Web. 21 Sep. 2012. doi:10.4018/jmdem.2010040103