ISSN 2277-3061

3-D from 2-D Using Warping Transformations

M. A. Ashabrawy[1], E. E. Elbehadi[2]
[1] Salman bin Abdulaziz University, Community College, Computer Department, KSA
ashabrawy@hotmail.com
[2] Salman bin Abdulaziz University, Faculty of Science, Mathematics Department, KSA.
h_feast@yahoo.com

ABSTRACT

Shown in this paper are methods on how to find the third dimension of a single image or from the two views of the image taking in a different angle using the method more accurate and faster to get to the third dimension in the following cases: One image of the same scene. Two views of the same scene from two different perspectives. Pictures of parts of the same scene. Set of pictures for different views of the work of the subject Panorama. This method is known Image Warping, which falls below a set of transfers such as (Affine - Bilinear - Projective - Mosaic – Similarity transformation) was compared to the work of transfers between the previous and this will be applied to more pictures. The idea is based on building code software is built on the programming language Visual C + + with the library for drawing an OpenGL program Matlab, which way can build a model of the following conversions, which fall under the so-called image warping of the conversion linear Bilinear Mapping and conversion Affine Mapping and conversion imagery Projective Mapping .

shown in this paper are methods on how to correct camera exposure changes and how to blend the stitching line between the images. We will show panorama photos generated from both still image.

Keywords

Digital image warping, Digital image processing, Bilinear mapping, Affine Transformations, Image Mosaicing, Panorama, Projective mapping.

Council for Innovative Research
Peer Review Research Publishing System
Journal: INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY
Vol 13, No. 4
editorijctonline@gmail.com
www.cirworld.com/journals

[1] Reactors Department, Nuclear Research Center , Atomic Energy Authority Egypt.
[2] Zagazig University , Faculty of Science, Mathematics Department, Egypt.

Introduction

Digital image warping is a growing branch of image processing that deals with geometric transformation techniques. Early interest in this area dates back to the mid-1960s when it was introduced for geometric correction applications in remote sensing. Since that time it has experienced vigorous growth, finding uses in such fields as medical imaging, computer vision, and computer graphics. Although image warping has traditionally been dominated by results from the remote sensing community, it has recently enjoyed a new surge of interest from the computer graphics field. This is largely due to the growing availability of advanced graphics workstations and increasingly powerful computers that make warping a viable tool for image synthesis and special effects. Work in this area has already led to successful market products such as real-time video effects generators for the television industry and cost-effective warping hardware for geometric correction. Current trends indicate that this area will have growing impact on desktop video.

Digital image warping has benefited greatly from several fields, ranging from early work in remote sensing to recent developments in computer graphics. The scope of these contributions, however, often varies widely owing to different operating conditions and assumptions morphing = $(\text{warping})^2$ + blending.

The equation above refers to the fact that morphing is a two-stage process which involves coupling image warping with color interpolation. As the morphing proceeds, the first image (source) is gradually warped towards the second image (target) while fading out. At same time the second image starts warping towards the first image and is faded in. Thus, the early images in the sequence are much like the first image. The middle image of the sequence is the average of the first image distorted halfway towards the second one and the second image distorted halfway back towards the first one. The last images in the sequence are similar to the second one. Then, the whole process consists of warping two images so that they have the same "shape" and then cross dissolving the resulting images [1].

Geometric transformations permit elimination of the geometric distortion that occurs when an image is captured. Geometric distortion may arise because of the lens or because of the irregular movement of the sensor during image capture.

Geometric transformation processing is also essential in situations where there are distortions inherent in the imaging process such as remote sensing from aircraft or spacecraft. On example is an attempt to match remotely sensed images of the same area taken after one year, when the more recent image was probably not taken from precisely the same position. To inspect changes over the year, it is necessary first to execute a geometric transformation, and then subtract one image to other. We might also need to register two or more images of the same scene, obtained from different viewpoints or acquired with different instruments. Image registration matches up the features that are common to two or more images. Registration also finds applications in medical imaging [2] and [3].

People have always been fascinated about capturing the entire view of the scenes. Before the era of digital cameras, wide-angle view is captured using special optical lens. However, these lenses are usually mounted on SLR cameras which most people do not have. Plus, lens distortion is often introduced in these pictures and even with the wide-angle lenses we are still unable to obtain the full 360 degree view. A new generation of digital cameras based on the line scanning technologies, such as the ones produced by Panoscan.com, allows us to capture incredible 360 degree views of the scenes. The pictures produced from these cameras often have very high quality. The drawback is that those cameras are very expensive and far beyond the reach of average consumers. One of the major advantages of using image processing is affordability as anyone can install a piece of software on a PC and is able to process the data to produce the panorama photos. However, since the images are taken at multiple moments while the camera is panning around the scene, they need to be registered to each other in order to obtain the final result. This registration or motion matching has proven to be a difficult problem and that is what most of the research work in this field has focused in the past. In a perfect world where we can have the camera placed horizontally and panning exactly around its focal point, if we know the tilt angle and how much the camera has panned, we can warp all the images to a sphere based on the focal field of view of the camera model. In the case when the tilt angle is zero, a cylinder is a good substitute for the sphere. Theoretically, all the images can be warped to such a common reference sphere or cylinder and we can then reproduce the entire field of view from this sphere or cylinder. This is known as spherical or cylindrical warping [1].

In reality, without knowing any camera angles and camera focal field of view, a correct estimation for this kind of warping is difficult to obtain. Instead, people have been mostly trying to use 2D planar matching techniques to obtain relative matching between two images, such as affine matching. However, without correct warping, there would always

be errors during introduced during the matching due to the perspective changes from the 3D scene to the 2D image. One interesting idea is to use only narrow center strips of video frames [4]. This approach works for high frame-rate video data. It, in fact, is mimicking the line-scan cameras mentioned earlier. The line-scan cameras scan one vertical line at a time and there is no geometric distortion. However, issues still remain. What if there are objects moving in the scene? The strips would likely cut the moving objects into parts. Plus, this approach would not work on still photo stitching.

The work in the literature has mostly focused on how to match images in the general cases of transformation, i.e. in the case when the camera pan, rotate, and tilt in any directions. We realize that no matter how well the matching is done, there will be some misalignments between the images. This could happen because the camera is drifting away from its initial focal point position, as is always the case for hand-held cameras. It could also happen because there are moving objects in the scene or because of the 3D to 2D transformation that can not be accounted by 2D image matching. So instead, this work rather focuses on the stitching side to avoid such kind of misalignments or make them less visible[5].

We do assume that people capture the data with the panorama photos in mind. This means that the camera is held roughly horizontally and the panning is done consistently along the horizontal or vertical direction, rather than the general scenario when the camera can be moved in any directions. We will present some interesting observations in motion matching assuming this panorama mode [6] and [7] and [8].

We will also show how to deal with some other practical issues in generating good panorama photos. One problem we have faced is the change of exposure in camera settings, since most cameras are in automatic mode and adjust to lights when taking pictures. As a result, one picture could be significantly brighter than another. And this needs to be corrected before the final stitching process. Another issue is on how to blend two images.

Methods and algorithms

1.0 Steps can be divided in three stages which are as follows:

(i) - Access to the images you want to work through the following:

- Scanner with a high resolution high dpi.
- Digital camera with a high resolution high dpi.

(ii) - After downloading the images on the computer begins the second phase of the programming consists of the following steps:

- Convert images from one image to a digital data file: (*. Dat ,*. Bmp or *. Jpg ...), convert to through a program that is currency programming languages (Matlab code) And this file is a digital file in two dimensions (matrix composed of two dimensions) 2-D.
- Then the candidate digital deviate that is used to address the work to the images to delete Disturbances and distortions that are in the photos and the work of purification of the images is done through the programming of this filter through the work program languages Programming the following in Fortran Language.
- Normalization of the digital file after the previous entry by the candidate in order to ensure that all the values of this file takes Limited to values between zero and one: [0,1], through the work on language Visual C++ . Language Repeat steps in (b) and (c) more than once until you improve the image and the work The opposite process is drawing digital file and return it to the image:(*. Dat Convert to *. Bmp or *. Jpg,…) data file is converted from digital two dimensions to three dimensions (2-D Convert to 3-D File *. Dat) this step is very important steps to terminate the work and to put thedecision and the end of the third stage and that through the work program of the following programming language Visual C + + .

(iii)- After the completion of the initial stages and the third stages are doing a software code of the previous transfers such as transfers (bilinear mapping, affine mapping, projective mapping and mosaicing) by Matlab program.

- After the construction of the previous transfers and applied on the image we find a fortress on the third dimension or depth of the image required.

1.1 Bilinear mapping

Bilinear mapping are most commonly defined as a mapping of a square into a quadrilateral this mapping can be computed by linearly interpolating u linearly along the top and bottom edges of the quadrilateral and then linearly interpolating v between two interpolated points to yield a destination point (x , y) [8] and [9] and [10].

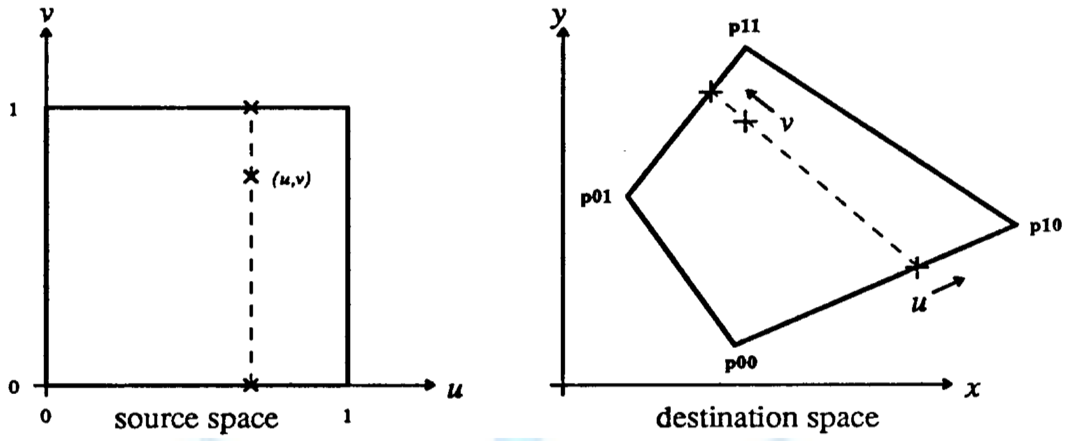

Fig1: Bilinear mapping

As:

$$[x \;\; y] = \vec{w}(x,y) = (1-u)(1-v)\,\vec{p}_{00} + u(1-v)\,\vec{p}_{10} + (1-u)v\,\vec{p}_{01} + uv\,\vec{p}_{11}$$

The general form in matrix rotation is:

$$\underset{\text{Given}}{\underset{\uparrow}{[x \;\; y]}} = \underset{\text{Given}}{\underset{\uparrow}{[uv \;\; u \;\; v \;\; 1]}} \underset{\text{to be obtained}}{\underset{\uparrow}{\begin{bmatrix} a & e \\ b & f \\ c & g \\ d & h \end{bmatrix}}}$$

$$x = u\,A$$

required to find A, the dimensional for x is (4 x 2) and for u is (4 x 4) and for A is (4 x 2) and we can find the u^{-1}:

$$x = u\,A$$

$$A = u^{-1}\,x$$

Example for finding the u^{-1}:

let

$$(u_0, v_0) = (0,0),\; (u_1, v_1) = (1,0)\;,$$
$$(u_2, v_2) = (0,1),\;\; (u_3, v_3) = (1,1)$$

For the first point (0, 0) we have A bilinear mapping is affine if a = e = 0. The matrix of 8 coefficients may be computed from the four point correspondence of Fig 1 as follows:

$$T = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \textit{ with } \begin{bmatrix} x_{00} & y_{00} \\ x_{10} & y_{10} \\ x_{01} & y_{01} \\ x_{11} & y_{11} \end{bmatrix}$$

Now it remains to find the inverse of the matrix T.

$$T^{-1} = \begin{bmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

By using Elementary row operation in Gauss- Seidl, the bilinear mapping has an unmusical wise of properties, because of its linear interpolation the forward transform from source to destination space preserves lines which are horizontal or vertical in the source space. But it dose not preserve diagonal lines, diagonal lines are actually mapped into parabola.

The inverse mapping from destination space to source space is not even single valued [11].

$$avu + cv = x - bu - d$$

$$x = avu + cv + bu + d$$ From,

$$[x \quad y] = [uv \; u \; v \; 1] \begin{bmatrix} a & e \\ \mathbf{b} & f \\ c & g \\ d & h \end{bmatrix}$$

We have the x- component:

$$x = avu + cv + bu + d$$

$$v = \frac{x - bu - d}{au + c}$$

Also have the y- component:

$$y = (e\,u + g)v + (f\,u + h)$$

Substituting for v from the above equation we get

$$y = (e\,u + g)\frac{(x - bu - d)}{au + c} + (f\,u + h)$$

Or

$$((f\,u + h) - y) = (eu + g)\frac{(x - bu - d)}{au + c}$$

Or

$$((f\,u + h) - y)(au + c) - (eu + g)(x - bu - d) = 0$$

Then we can done for v, yielding the two quadratic equations

$$(au + c)(f\,u + h - y) - (eu + g)(bu + d - x) = 0$$

$$(av + b)(gv + h - y) - (ev + f)(cv + d - x) = 0$$

So $$Au^2 + Bu + C = 0$$

$$Dv^2 + Ev + F = 0$$

We can find (u , v) in terms of (x , y) by evaluating the coefficient A, B, C above and then computing, where:

$$A = af - be \quad B = ex - ay + ah - de + cf - bg \quad C = gx - cy + ch - dg$$
$$D = ag - ce \quad E = ex - ay + ah - de - cf + bg \quad F = fx - by + bh - df$$

$$u = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

$$v = \frac{x - bu - d}{au + c}$$

The inverse transform is multi-valued and is much more difficult to compute than the forward transform.

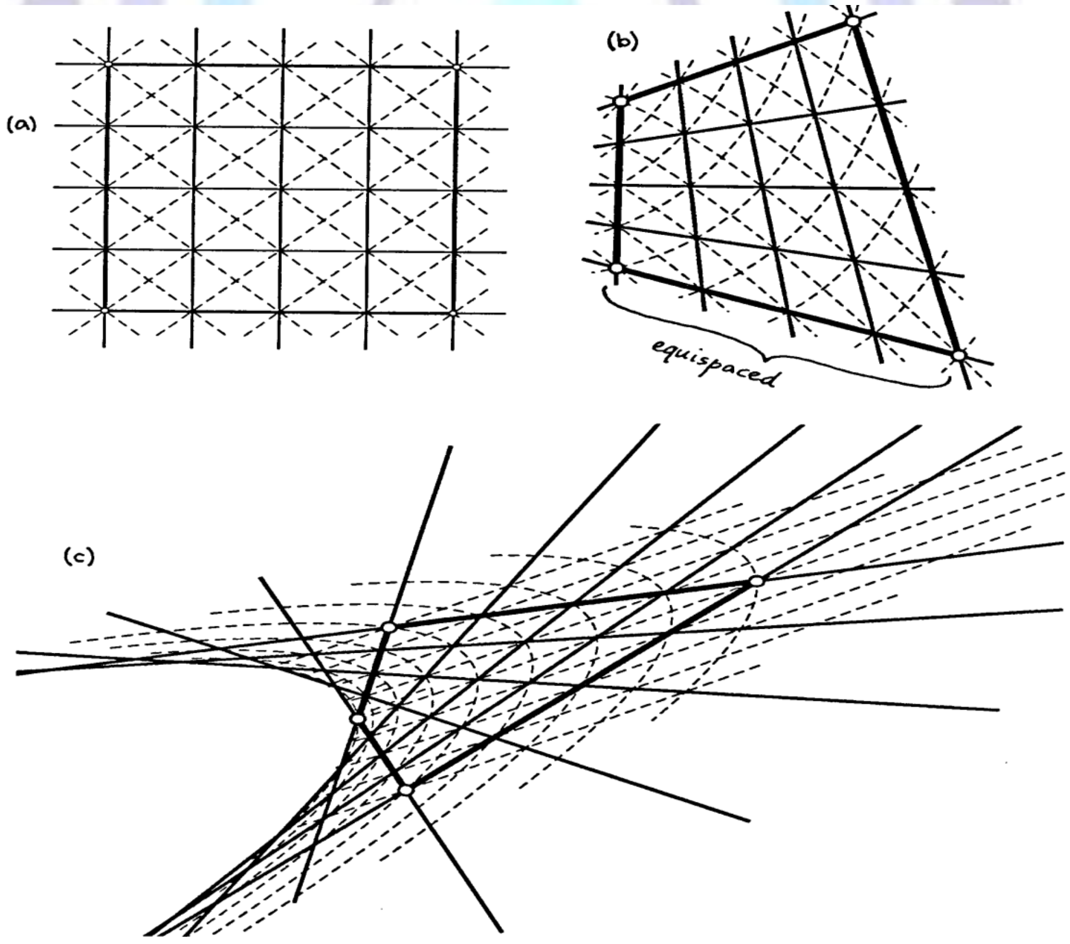

Fig 2: Bilinear warp of a test grid.

a) Grid in source space. B) Warped grid in destination space for quadrilateral 1 (vertices marked with dots) and note the curvature of diagonals.

c) Warped grid in destination space for quadrilateral 2. Diagonals map to parabolas in destination space. And note foldover of some diagonals (caused by the existence of two roots in quadratic equation).

In practice, the geometric transform is often approximated by the bilinear transformation. 4 pairs of corresponding points are sufficient to find the transformation.

$$u = a_0 + a_1 x + a_2 y + a_3 xy$$

$$v = b_0 + b_1 x + b_2 y + b_3 xy \quad \ldots\ldots\ldots\ldots\ldots\ldots \text{(1.1)}$$

Even simpler is the affine transformation for which three pairs of corresponding points are sufficient to find the coefficients

$$u = a_0 + a_1 x + a_2 y$$

$$v = b_0 + b_1 x + b_2 y \quad \ldots\ldots\ldots\ldots\ldots\ldots \text{(1.2)}$$

The only difference between bilinear and affine transformation is that the coefficients a_3 and b_3 in (1.2) are set to be zeros in (2.3). In fact, affine transformation is a particular form of bilinear transformation.

For example, a second-degree approximation requires only six coefficients to be solved. In this case, N=2 and K=6. We thus have the inverse mapping equation [12].

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \ldots \\ \ldots \\ u_M \end{bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 & x_1 y_1 & x_1^2 & y_1^2 \\ 1 & x_2 & y_2 & x_2 y_2 & x_2^2 & y_2^2 \\ 1 & x_3 & y_3 & x_3 y_3 & x_3^2 & y_3^2 \\ & & \ldots & \ldots & \ldots & \\ & & \ldots & \ldots & \ldots & \\ 1 & x_M & y_M & x_M y_M & x_M^2 & y_M^2 \end{bmatrix} \begin{bmatrix} a_{00} \\ a_{10} \\ a_{01} \\ a_{11} \\ a_{20} \\ a_{02} \end{bmatrix} \quad \ldots\ldots\ldots\ldots \text{(1.3)}$$

And forward mapping equation

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \ldots \\ \ldots \\ x_M \end{bmatrix} = \begin{bmatrix} 1 & u_1 & v_1 & u_1 v_1 & u_1^2 & v_1^2 \\ 1 & u_2 & v_2 & u_2 v_2 & u_2^2 & v_2^2 \\ 1 & u_3 & v_3 & u_3 v_3 & u_3^2 & v_3^2 \\ & & \ldots & \ldots & \ldots & \\ & & \ldots & \ldots & \ldots & \\ 1 & u_M & v_M & u_M v_M & u_M^2 & v_M^2 \end{bmatrix} \begin{bmatrix} a_{00} \\ a_{10} \\ a_{01} \\ a_{11} \\ a_{20} \\ a_{02} \end{bmatrix} \quad \ldots\ldots\ldots\ldots \text{(1.4)}$$

Where $M \geq 6$. A similar equation holds for v and b_{ij}. Both of these expressions can be written in matrix form as

$$U = WA$$

$$V = WB \quad \text{For inverse mapping} \ldots\ldots\ldots\ldots \text{(1.5)}$$

And

$$X = WA$$

$$Y = WB \quad \text{For forward mapping} \ldots\ldots\ldots\ldots \text{(1.6)}$$

This is a least-squares problem. Pseudo-inverse solution gives the following results:

$$A = (W^T W)^{-1} W^T U$$

$$B = (W^T W)^{-1} W^T V \quad \ldots\ldots\ldots\ldots\ldots\ldots \text{(1.7)}$$

1.2 Affine Transformations

Affine mappings include scales, rotation, translation, and shears; they are linear mappings plus a translation[3] and [9]. Formally, A mapping T(x) is linear if and only if t(x + y) = T(x) + T(y) and T(αx) = α T(x) for any scalar α.

A mapping T(x) is affine if and only if there exits a constant c and a linear mapping L(x) such that. T(x) = L(x) +c for all x. obviously, linear mappings are a subset of affine mappings.

A general 2-D affine mapping may be written

$$P_d = P_s M_{sd} \quad \text{.................................... (2.1)}$$

$$\begin{pmatrix} x & y & 1 \end{pmatrix} = \begin{pmatrix} u & v & 1 \end{pmatrix} \begin{bmatrix} a & d & 0 \\ b & e & 0 \\ c & f & 1 \end{bmatrix} \quad \text{..........(1.2)}$$

$$x = au + bv + c$$

$$y = au + bv + f \quad \text{...................(2.3)}$$

Transform matrix M_{ab} where a is the initial coordinate system and b is the final coordinate system has 6 degrees of freedom. The vectors (a , d) I = (1, 0) and (b , e) I = (0, 1) are basis vectors of the destination space and (c, f) is the origin.

The dimensional for P_a is (1 x 3) and for P_b is (1 x 3) and for M_{ab} is (3 x 3) Affine mapping preserve parallel lines and equispaced point are transformed as equispaced point along a line in the destination space, although the spacing in the two coordinate systems may be different.

We can invert and affine mapping to find the destination to source transformation simply by inverting the mapping matrix M_{sd} if and only if M_{sd} has an inverse. Since an affine mapping has 6 degrees of freedom, it may be defined genetically by specifying the source points (u_0, v_0), (u_1, v_1), (u_2, v_2) and the destination points (x_0, y_0), (x_1, y_1), (x_2, y_2) this is obviously done by using

$$\begin{bmatrix} x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{bmatrix} = \begin{bmatrix} u_0 & v_0 & 1 \\ u_1 & v_1 & 1 \\ u_2 & v_2 & 1 \end{bmatrix} \begin{bmatrix} a & d & 0 \\ b & e & 0 \\ c & f & 1 \end{bmatrix} \quad \text{.......................... (2.4)}$$

Affine mapping can either map any triangle in source space into any triangle in destination space, or map a source rectangle into a parallelogram but no more general destinations are possible. To warp a rectangle into general quadrilaterals, we need a bilinear projection, or some other more complex mapping.

Properties of Affine Transformations.

Affine transformations preserve affine combination of points W = a1P1 + a2P2 is affine then

T(W) = a1T(P1) + a2T(P2) is affine.. (2.5)

This follows because the transformation is represented by matrix multiplication.

Affine transformations preserve lines and planes a line is represented by two points parametrically by L (t) = (1-t) A + Bt Parallel lines remain parallel under affine transformation, two parallel lines can be expressed as:

L1(t) = A1 + bt

L2(t) = A2 + bt... (2.6)

Then the transformed lines are

T(L1(t)) = MA1 + Mbt

T(L2(t)) = MA2 + Mbt

are still parallel.

Note :

- Affine transformation cannot represent perspective because they can not generate vanishing Points.
- Any Affine Transformation is composed of elementary operations.

M = (shear)(scaling)(rotation)(translation) in 2D.

M = (scaling (rotation) (shear1) (shear2) (translation) in 3.

1.3 Projective mappings

The Projective mapping, also known as the perspective or homogeneous transformation, is a projection from one to plane through a point onto another plane[9] .Homogeneous transformations are used extensively for 3-D affine modeling transformations and for perspective camera transformations[12].

The 2-D projective mappings studied here are a subset of these familiar 3-D homogeneous transformations.

The general form of a projective mapping is a rational linear mapping:

$$x=\frac{au+bv+c}{gu+hv+i}\ ,\qquad y=\frac{du+ev+f}{gu+hv+i} \quad \ldots\ldots\ldots\ldots\ldots\ldots (3.1)$$

Manipulation of projective mappings is much easier in the homogeneous matrix notation:

$$p_d=p_d\,M_{sd}=\left(x'\ \ y'\ \ \omega\right)=(u'\ \ v'\ \ q)\begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix} \ldots\ldots\ldots\ldots\ (3.2)$$

Where

$$(x,y)=(x'/\omega\ ,\ y'/\omega\ )\ for\ \omega=\neq 0, and\ (u,v)=(u'/q,v'/q\ )\ for\ q\neq 0.$$

Although there 9 coefficients in the matrix above, these mappings are homogeneous, so any nonzero scalar multiple of these matrix gives an equivalent mapping. Hence there are only 8 degrees of freedom in a 2-D projective mapping. We can assume without loss of generality that i=1 except in the special case that source point (0, 0) maps to a point at infinity. A projective mapping is affine when g = h = 0.

Projective mappings will in general map the line at infinity to a line in the real plane. We can think of this line as the horizon line of all vanishing points (is a point in a perspective drawing to which parallel lines not perpendicular to the image plane appear to converge), by analogy to the perspective projection.

Affine mapping are the special case of projective mappings that map the line at infinity into itself. By defining the projective mapping over the projective plane and not just the real plane, projective mappings become bijections (one-to-one and onto), except when the mapping matrix is singular. For non-degenerate mappings the forward and inverse transforms are single-valued, just as for an affine mapping [11] and [12].

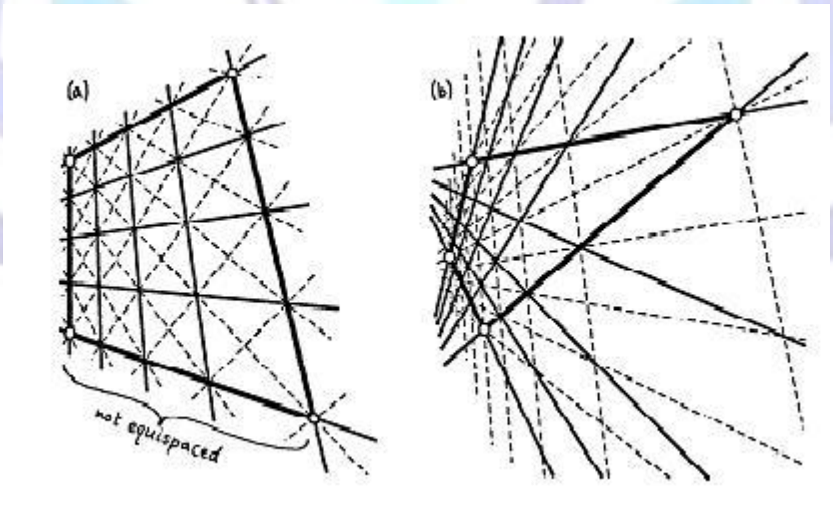

Fig 3: Projective warps of a test grid controlled

a) Warp grid in destination space for quadrilateral b) Warped grid in destination space for quadrilateral 2.

Note the vanishing points.

Projective mappings share many of the desirable properties of affine mapping. Unlike bilinear mappings, which preserve equispaced points along certain lines, the projective mappings do not in general preserve equispaced points as in Fig 3

Instead they preserve a quantity called the cross ratio of points [11]. Like affine mapping, projective mappings preserve lines at all orientations. In fact, projective mappings are the most general line- preserving projective mappings and may be concatenating their matrices. Another remarkable property is that the inverse of a projective mappings, which explained by reversing the plane-to-plane mapping by which a projective mapping is defined. The matrix for the inverse mapping is the inverse or adjoint of the forward mapping. (The adjoint of a matrix is the transpose of the matrix of cofactors [32], $M^{-1} = adj(M) / \det(M)$).

In homogeneous algebra, the adjoint matrix can be used in place of the inverse matrix whenever an inverse transform is needed, since the two are scalar multiples of each other, and the adjiont always exits, while the inverse does not if the matrix is singular. The inverse transformation:

$$p_s = p_d \, M_{ds}$$

$$=(u' \;\; v' \;\; q)=(x' \;\; y' \;\; \omega)\begin{bmatrix} A & D & G \\ B & E & H \\ C & F & I \end{bmatrix}=$$

$$=(x' \;\; y' \;\; \omega)\begin{bmatrix} ei-fh & fg-di & dh-eg \\ ch-bi & ai-cg & bg-ah \\ bf-ce & cd-af & ae-bd \end{bmatrix} \quad \text{.............. (3.4)}$$

When mapping a point by the inverse transform we compute (u, v) from (x, y). If $\omega \neq 0 \text{ and } q \neq 0$ then we can choose $\omega = 1$ and calculate:

$$u=\frac{Ax+By+c}{Gx+Hy+I} \;, \qquad v=\frac{Dx+Ey+F}{Gx+Hy+I} \quad \text{................ (3.5)}$$

In an interactive image warper one might specify the four corners of source and destination quadrilaterals with a tablet or mouse, and wish to warp one area to the other. This sort of task is an ideal application of projective mappings, but the problem is to find the mapping matrix.

A projective mapping has 8 degrees of freedom which can be determined from the source and destination coordinates of the four corners of a quadrilateral. Let correspondence map (finite).To compute the forward mapping matrix M_{sd}, assuming that i=1, we have eight equations in the eight unknowns from a to g:

$$x_k=\frac{a u_k+b v_k+c}{g u_k+h v_k+1} \Rightarrow u_k a+v_k b+c-u_k x_k g-v_k x_k h=x_k \quad \text{......(3.6)}$$

$$y_k=\frac{d u_k+e v_k+f}{g u_k+h v_k+1} \Rightarrow u_k d+v_k e+f-u_k y_k g-v_k y_k h=y_k \quad \text{...... (3.7)}$$

For k = 0, 1, 2, 3. This can be rewritten as an 8 x 8 system:

$$\begin{bmatrix} u_0 & v_0 & 1 & 0 & 0 & 0 & -u_0v_0 & -v_0x \\ u_1 & v_1 & 1 & 0 & 0 & 0 & -u_1v_1 & -v_1x_1 \\ u_2 & v_2 & 1 & 0 & 0 & 0 & -u_2v_2 & -v_2x_2 \\ u_3 & v_3 & 1 & 0 & 0 & 0 & -u_3v_3 & -v_3x_3 \\ 0 & 0 & 0 & u_0 & v_0 & 1 & -u_0v_0 & -v_0y_0 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -u_1v_1 & -v_1y_1 \\ 0 & 0 & 0 & u_2 & v_2 & 1 & -u_2v_2 & -v_2y_2 \\ 0 & 0 & 0 & u_3 & v_3 & 1 & -u_3v_3 & -v_3y_3 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \quad \text{........ (3.8)}$$

This linear system can be solved using Gaussian elimination for the forward mapping coefficients a to h. If the inverse mapping is desired instead, then either we compute the adjoint of M_{sd} or we follow the same procedure, starting from Equ. (3.4) instead of Equ. (3.1), and solve an 8 x 8 system for coefficients A to H.

There are more efficient formulas for computing the mapping matrix. The formula above handles the case where the polygon is a general quadrilateral in both source and destination spaces. We will consider three additional cases: square-to- quadrilateral, quadrilateral-to-square, and the general quadrilateral-to-quadrilateral mapping.

Case1. The system is easily solved symbolically in the special case where the uv quadrilateral is a unit square. If the vertex correspondence is as follows:

Table 1

x	y	u	v
x_0	y_0	0	0
x_1	y_1	1	0
x_2	y_2	1	1
x_3	y_3	0	1

Then the eight equations reduce to:-

$$c = x_0$$

$$a + c - g\,x_1 = x_1$$

$$a + b + c - g\,x_2 + h\,x_2 = x_2$$

$$b + c - h\,x_3 = x_3$$

$$f = y_0$$

$$d + f - g\, y_1 = y_1$$

$$d + e + f - g\, y_2 - h\, y_2 = y_2$$

$$e + f - h\, y_3 = y_3 \qquad \text{................ (3.8)}$$

If we define

$$\Delta x_1 = x_1 - x_2 \qquad \Delta x_2 = x_3 - x_2 \qquad \sum x = x_0 - x_1 + x_2 - x_3$$

$$\Delta y_1 = y_1 - y_2 \qquad \Delta y_2 = y_3 - y_2 \qquad \sum y = y_0 - y_1 + y_2 - y_3$$

Then the solution splits into two sub-cases:

$(a) \sum x = 0 \ and \ \sum y = 0.$ This implies that the xy polygon is parallelogram, so the mapping is affine, and

$a = x_1 - x_0$, $b = x_2 - x_1$, $c = x_0$, $d = y_1 - y_0$, $e = y_2 - y_1$, $f = y_0$, $g = 0$, $h = 0.$

$(b) \sum x \neq 0 \ or \ \sum y \neq 0$. Gives a projective mapping:

$$g = \frac{\begin{bmatrix} \sum x & \Delta x_2 \\ \sum y & \Delta y_2 \end{bmatrix}}{\begin{bmatrix} \Delta x_1 & \Delta x_2 \\ \Delta y_1 & \Delta y_2 \end{bmatrix}} \quad h = \frac{\begin{bmatrix} \Delta x_1 & \sum x \\ \Delta y_1 & \sum y \end{bmatrix}}{\begin{bmatrix} \Delta x_1 & \Delta x_2 \\ \Delta y_1 & \Delta y_2 \end{bmatrix}} \qquad \text{.......................... (3.9)}$$

$$\begin{aligned} a &= x_1 - x_0 - g\,x_1 \\ b &= x_3 - x_0 + h\,x_3 \\ c &= x_0 \\ d &= y_1 - y_0 - g\,y_1 \\ e &= y_3 - y_0 - h\,y_3 \\ f &= y_0 \end{aligned} \qquad \text{.. (3.10)}$$

This computation is much faster than a straightforward 8 x 8 system solver. Then mapping above is easily generalized to map a rectangle to a quadrilateral by pre-multiplying with a scale and translation matrix.

Case 2. The inverse mapping, a quadrilateral to a square, can also be optimized. It turns out that the most efficient algorithm for computing this is not purely symbolic, as in the previous case, but numerical. We use the square-to-quadrilateral formulas just described to find the inverse of the desired mapping and then take its adjoint to compute the quadrilateral –to-square mapping [11].

Case 3. Since we can compute quadrilateral-to-square and square-to- quadrilateral mappings quickly, the two mappings can easily be composed to yield a general quadrilateral-to-quadrilateral mapping as Fig 4 below. This solution method is faster than a general 8 x 8 system solver.

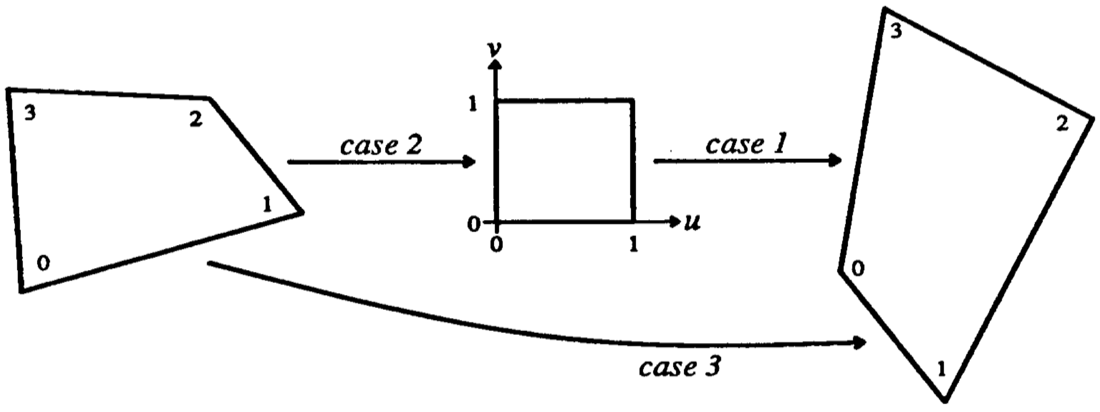

Fig 4: Quadrilateral-to-quadrilateral mapping as a composition of simpler mappings.

1.4 Comparison of simple mapping

Affine and projective mappings are closed under composition, but bilinear mappings are not. The three basic mapping classes compose as follows:

Table **2**

$\Downarrow$ MAP1 $\circ$ MAP2 $\Rightarrow$	Affine	Bilinear	Projective
Affine	Affine	Bilinear	Projective
Bilinear	Bilinear	Biquadratic	Rational Bilinear
Projective	Projective	Rational Biquadratic	Projective

Thus, the composition of two bilinear mapping is a biquadratic mapping. Since (nonsingular) affine and projective mappings are closed under composition, have an identity and inverses and obey the associative law, they each form a group under the operation of composition. Bilinear mappings do not form a group in this sense and [12].

We summarize the properties of affine, bilinear and projective mappings below:

Table 3

Property	**Affine**	**Bilinear**	**Projective**
Preserves parallel lines	Yes	No	No
Preserves lines	Yes	No	Yes
Preserves equispaced points	Yes	No	No
Maps square to	Parallelogram	Quadrilateral	Quadrilateral
Degrees of freedom	6	8	8
Closed under composition	Yes	No, biquadratic	Yes

From the above table, affine mappings are the simplest of the three classes. If more generality is needed, then projective mappings are preferable to bilinear mappings because of the predictability of line preserving

mappings. For the implementer, the group properties of affine and projective mappings make their inverse mapping as easy to compute as their forward mappings. Bilinear mappings are computationally preferable to projective mappings only when the forward mapping is used much more heavily than the inverse mapping [13].

1.5 Image Mosaicing

One application for image warping is merging of several images into a complete mosaicing to form a panoramic view. In mosaicing, the transformation between images is often not know beforehand [8]. Two images are merged and we will estimate the transformation by letting the user give points of correspondence (also called landmarks or fiducially markers) in each of the images. In order to recover the transformation, the warping equation so that warping parameters is the vector t in [13] and [14]:

$$x' = Z\,t$$ ………………………………… (5.1)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 \\ y & -x & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s\cos\alpha \\ s\sin\alpha \\ t_x \\ t_y \\ 1 \end{bmatrix}$$ ………………………….. (5.2)

The warping parameters is now obtained by solving the linear system above in $t = Z/xp$ where xp (:) and

yx (:) are column vectors containing the x' *and* y' coordinates in matlab.

Mathematical Formulation of parametric warping for video registration [5] and [16]

Let:

(i) $\vec{x} = (x, y)^T$ denote some pixel coordinates in the image plane, where $\vec{x} = \begin{bmatrix} x \\ y \end{bmatrix}$.

(ii) $I\ (\vec{x}, t)$ be the intensity value for image pixel $\vec{x}$ is in same two dimensional images I, taken at same time t.

(iii) The gray level image gradient at the image point $\vec{x}$ is denoted by $\nabla_{\vec{x}}\ (\vec{x}, t)$.

A region I with N pixels after same time T has elapsed is defined finally by:

$$\vec{I} = (\vec{x}_1\ , \vec{x}_2\ , \vec{x}_3 , ..., \vec{x}_N\ )$$

Let f be same transformation relation $f : R^2 \to R^2$. Which is parameterized by the coefficient vector μ, $f\ (\vec{x}\ ; \mu(t))$ denotes the parametric motion of each image pixel $\vec{x}$ in terms of $\mu(t)$ with m components, which $\mu(t)$ is a set of time variant parameters that need to be estimated in real time [19] and [20].

$$\mu(t) = a_0 + a_1 t + a_2 t + a_3 t + a_4 t + a_5 t + a_6 t + a_7 t + a_8 t + ... \quad .... (5.3)$$

This transformation called warping can be expressed as: $\vec{u} = f\ (\vec{x}\ ; \mu\ )$, $\vec{u} = f\ (\vec{x}\ )$ at the same time determined by μ .

The transformation called warping can be expressed as: $\vec{u} = (u, v)^T = f\ (\vec{x}\ ; \mu\ )$ with the intensity constancy constraint, this means that the pixel intensity ravines even if constant and unchanged its position changed, i.e. $I\ (f\ (\vec{x}\ ; \mu(t)), t) = I_0\ (\vec{x}\ , t_0\ )$ as delay [18].

For certain image region I, we denote its pixel intensity for each $\vec{x}_i$ and time t by:

$$\vec{I}\ (\mu, t) = \begin{bmatrix} I(\ f\ (\vec{x}_1 ; \mu(t)), t) \\ I(\ f\ (\vec{x}_2 ; \mu(t)), t) \\ . \\ . \\ . \\ I(\ f\ (\vec{x}_N ; \mu(t)), t) \end{bmatrix} \quad .... (5.4)$$

By making the constant intensity assumption, the motion of image pixels can be represented in terms of their spatial and temporal derivation as:

$$\vec{I}\ (\mu(t) + \delta\mu, t + \Delta t) = \vec{I}\ (\mu, t) + M\ (\mu, t)\ \delta\mu + \Delta t\, \vec{I}_t\ (\mu, t) + (h.o.t)\ [Higher\ order\ termes]$$

$$.. (5.5)$$

The above linearization is carried out by Taylor series of μ *and* t as:

$$f\ (x + \Delta x, y + \Delta y) = f\ (x, y) + [\ \frac{\partial f}{\partial x} \Delta x + \frac{\partial f}{\partial y} \Delta y\ ] + ... = f\ (x, y) + [\ \nabla f\ ] + ...$$

$$f\ (x + \Delta x, y + \Delta y) = f\ (x, y) + [\ \frac{\partial f}{\partial x} \Delta x + \frac{\partial f}{\partial y} \Delta y\ ]\ \frac{f}{1 * 1} + [\ \frac{\partial f}{\partial x}\ \Delta x + \frac{\partial f}{\partial y} \Delta y]^2\ \frac{f}{2 * 1} + ... =$$

$$.. (5.6)$$

$\vec{I}_t$ is the derivative of $\vec{I}$ with respect to the component of time parameter t and is written as:

$$\vec{I}_t(\mu,t)=\begin{bmatrix} I_t(f(\vec{x}_1;\mu(t)),t) \\ I_t(f(\vec{x}_2;\mu(t)),t) \\ . \\ . \\ . \\ I_t(f(\vec{x}_N;\mu(t)),t) \end{bmatrix} \qquad \text{...} (3.5)$$

M is the Jacobian matrix of $\vec{I}$ with respect to μ , which N x m matrix of partial derivatives.

$$M(\mu,t)=\left(\vec{I}_{\mu 1}(\mu(t),t) \quad \vec{I}_{\mu 2}(\mu(t),t) \quad ..... \quad \vec{I}_{\mu m}(\mu(t),t) \right)$$

$$\vec{I}_{\mu i}(\mu,t)=\begin{bmatrix} I_{\mu i}(f(\vec{x}_1;\mu(t)),t) \\ I_{\mu i}(f(\vec{x}_2;\mu(t)),t) \\ . \\ . \\ . \\ I_{\mu i}(f(\vec{x}_N;\mu(t)),t) \end{bmatrix} \qquad \text{...} (5.6)$$

From the intensity constant constraint, the motion parameter vector of the image region can be estimated at time t by minimizing the following least squares error function as[16] and [19]:

$$\vec{e}\,(\mu,t)=\sum(I(f(\vec{x};\mu(t),t),t)-I_0(\vec{x},t_0\,))^2 \qquad \text{.................................} (5.7)$$

By substituting Equ. (5.5) into Equ. (5.7) and ignoring the higher order terms, we obtain:

$$\vec{e}\,(\mu,t)=\left\| \vec{I}(\mu,t)+\vec{M}\,\delta\,\mu+\vec{I}(\mu,t)\Delta t-\vec{I}(0,t_0)\right\|^2 \qquad \text{.........................} (5.8)$$

With the additional approximation $\vec{I}(\mu,t)\,\Delta t\approx\,\vec{I}(\mu,t+\Delta t)-\vec{I}(0,t_0)$ then

$$\vec{e}\,(\mu,t)=\left\| \vec{M}\,\delta\,\mu+\vec{I}(\mu,t+\Delta t)-\vec{I}(0,t_0)\right\|^2 \qquad \text{...............................} (5.9)$$

In this equation, $\vec{I}(\mu,t+\Delta t)-\vec{I}(0,t_0)$ can be considered as the distortion errors of warping image.

Applications

- One a single image still image.

- Two views from two different perspectives.

- panorama photos generated from both still image.

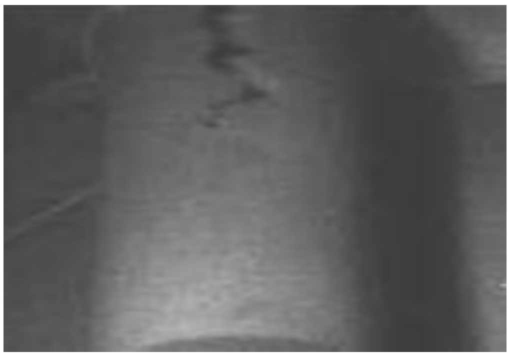

Fig 4a: One a single image still image

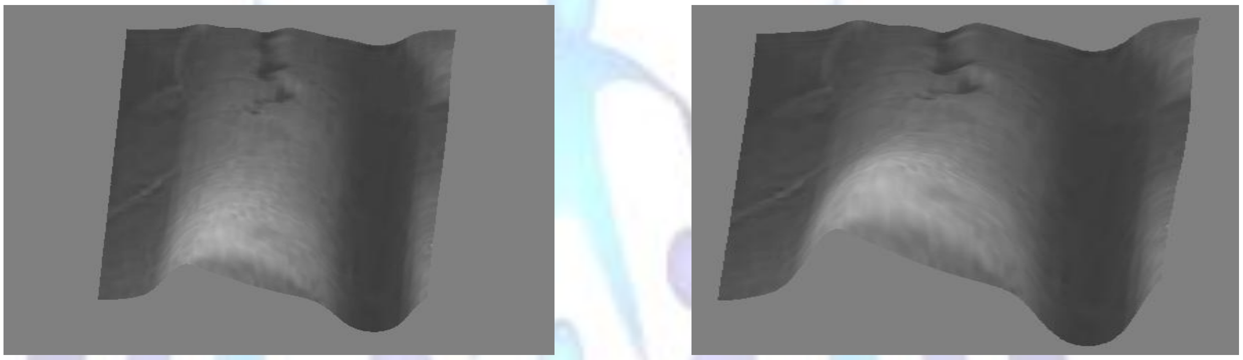

Fig 4b: Applying of OpenGL programs on one a single image still image.

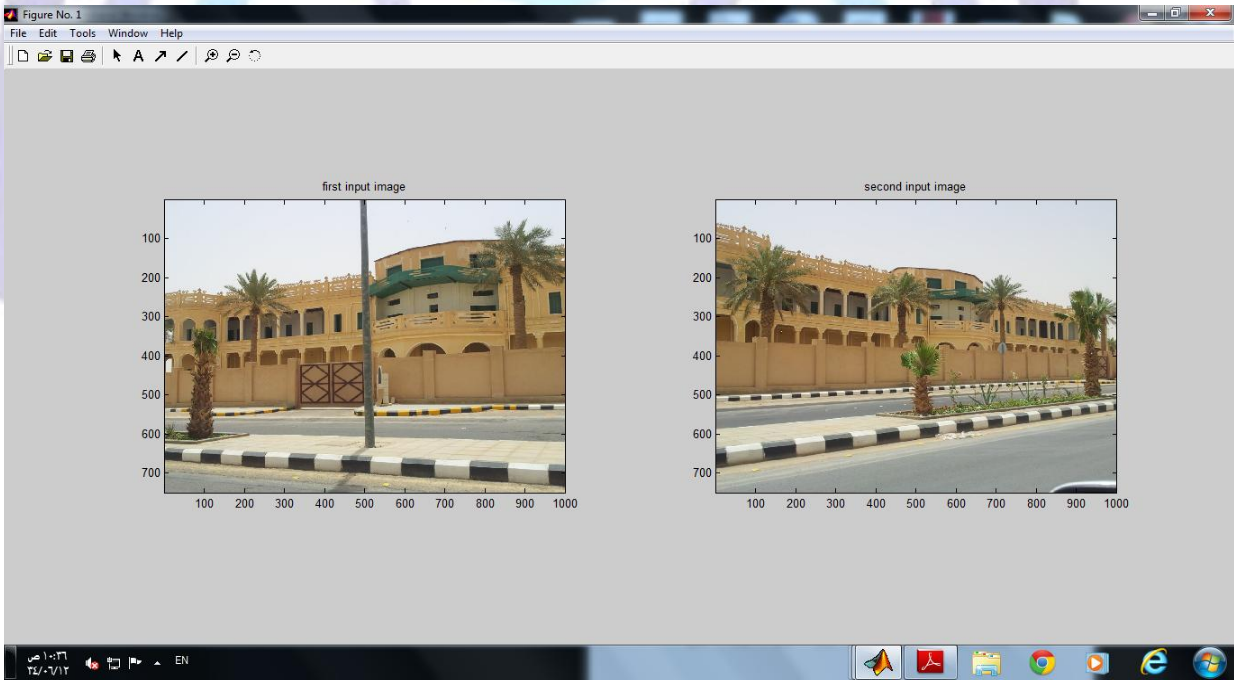

Fig 5a: Input (First image and Second image)

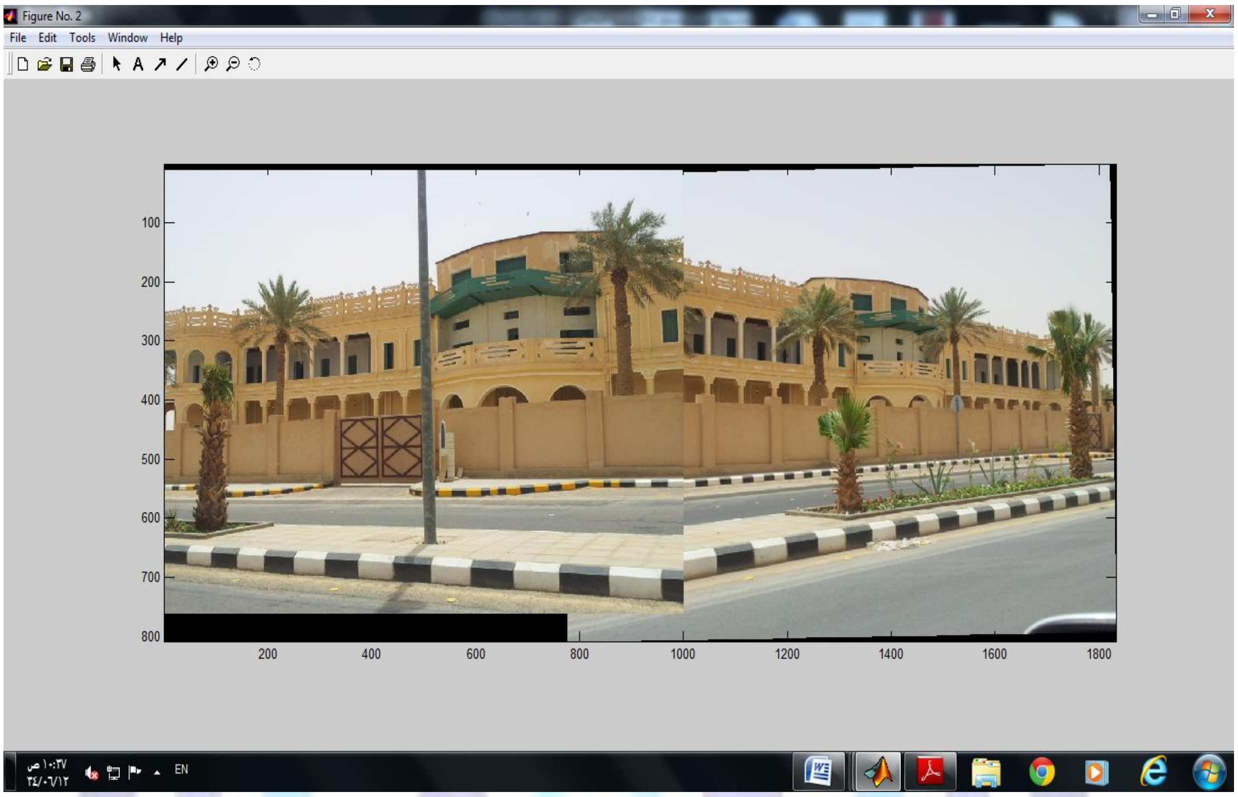

Fig 5b: Mosaic image after we apply the model of mosaicing Panorama.

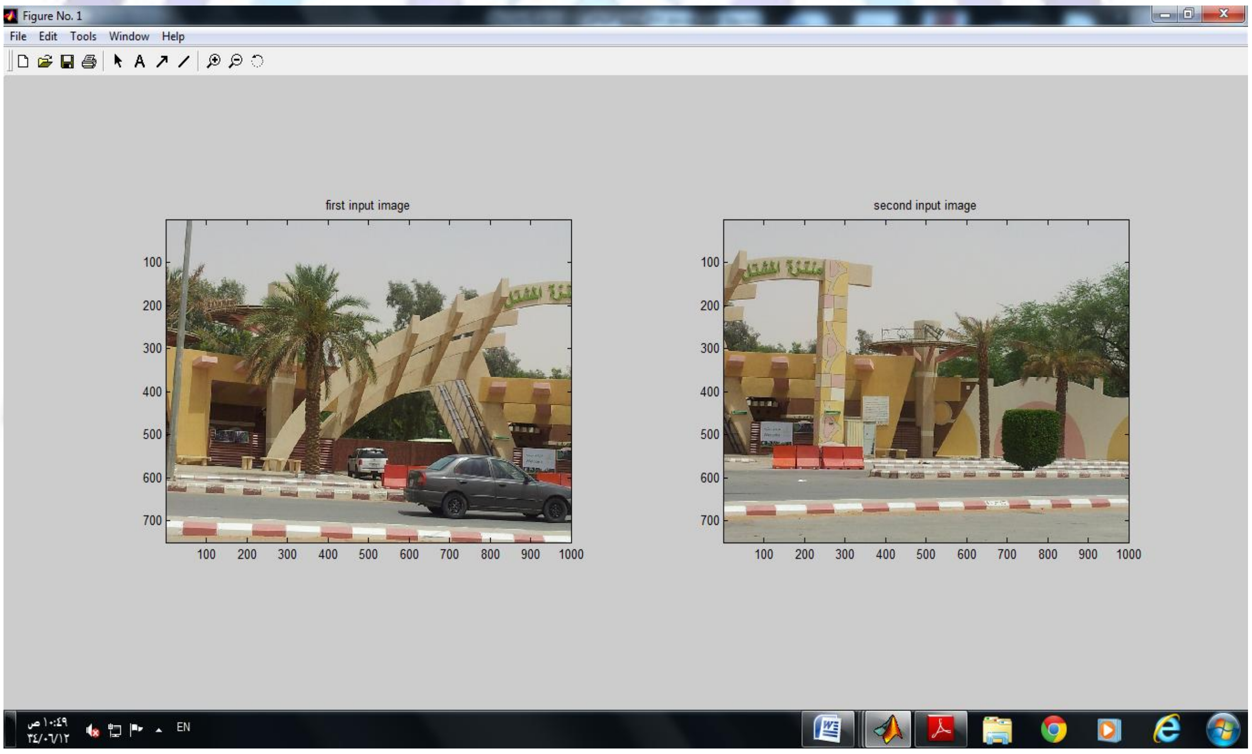

Fig 6a: Input (First image and Second image)

ISSN 2277-3061

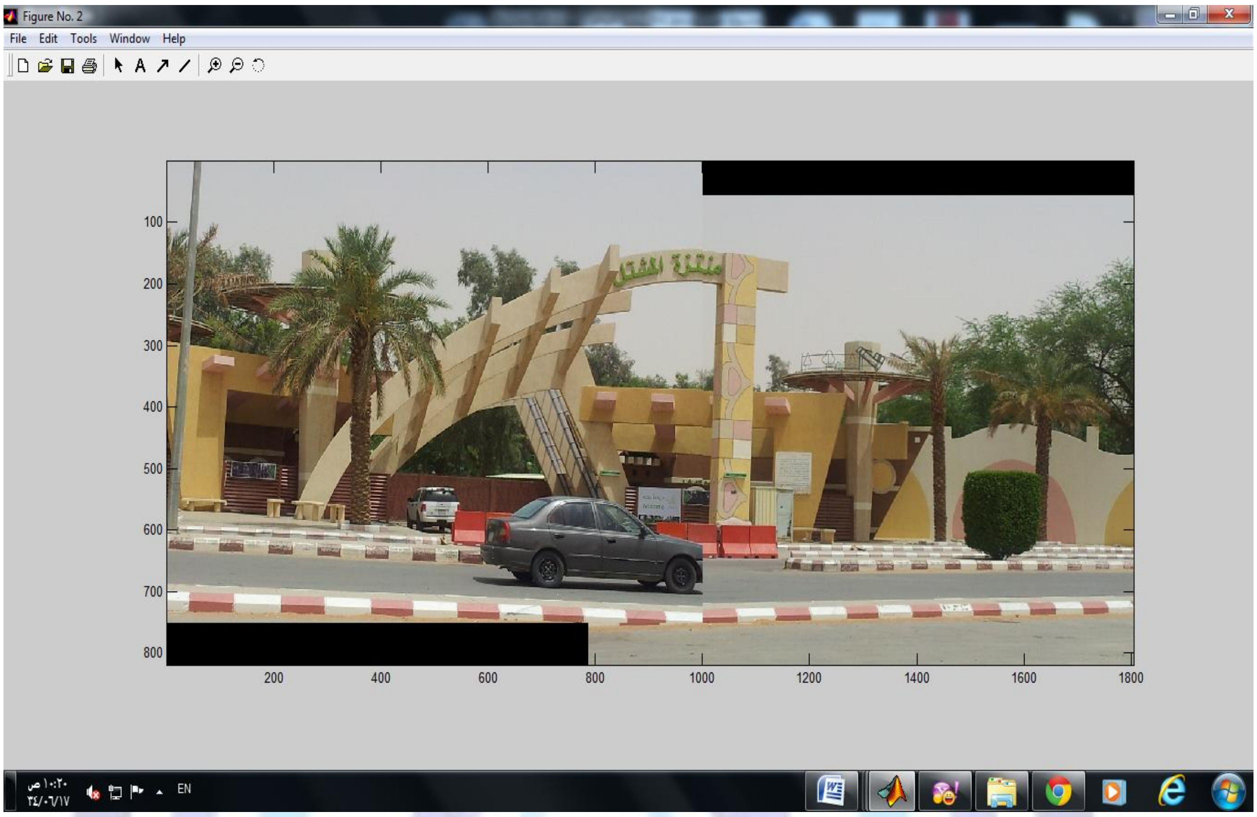

Fig 6b: Mosaic image after we apply the model of mosaicing Panorama.

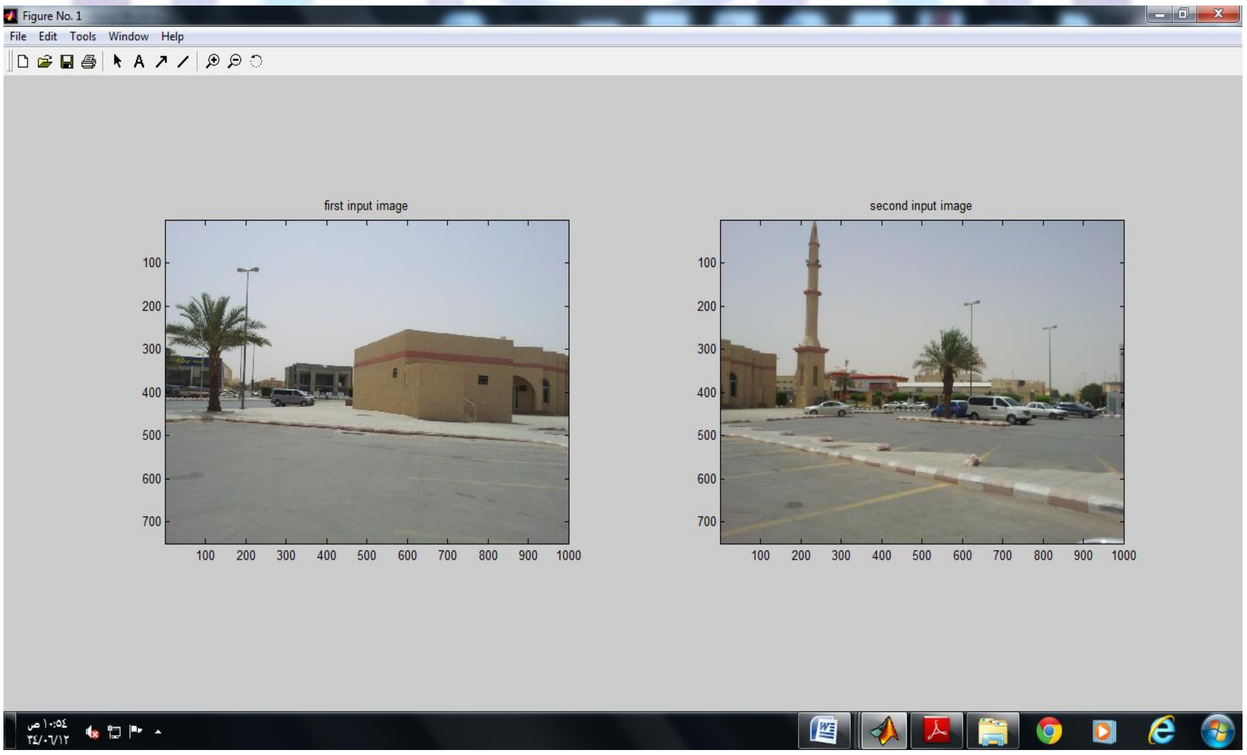

Fig 7a: Input (First image and Second image)

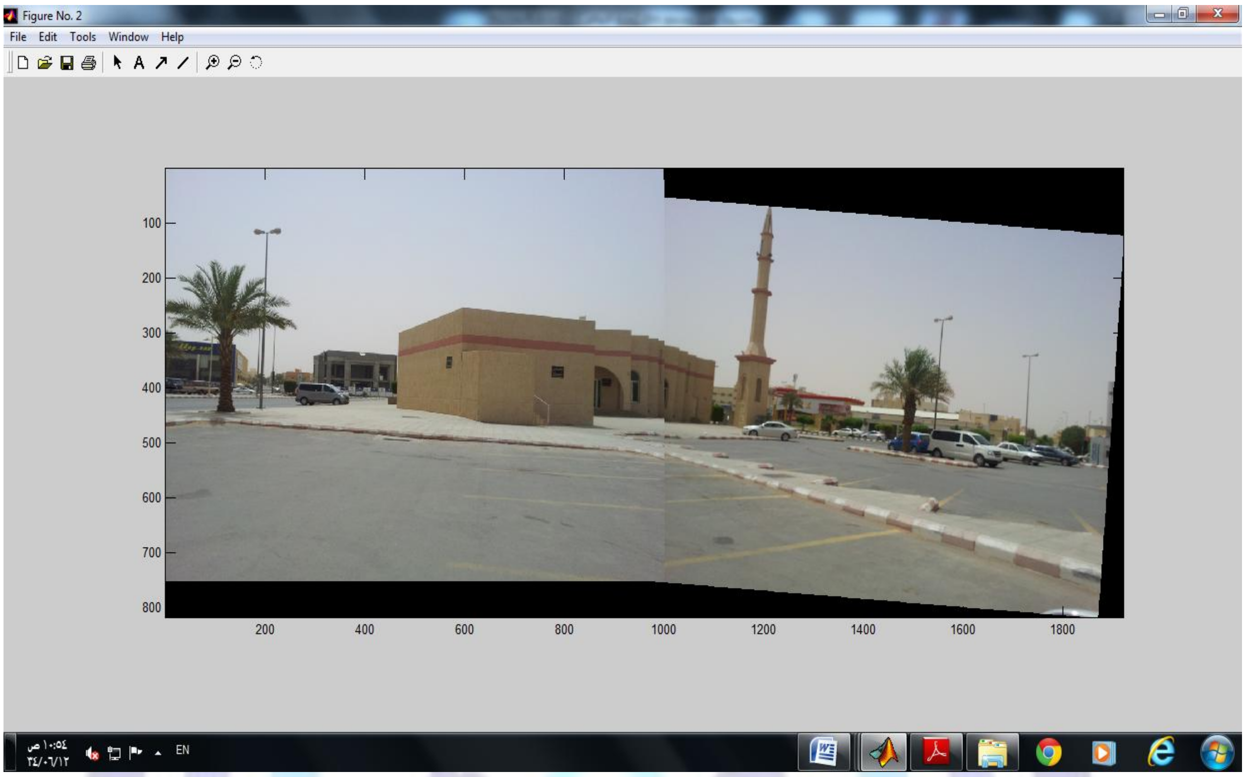

Fig 7b: Mosaic image after we apply the model of mosaicing Panorama.

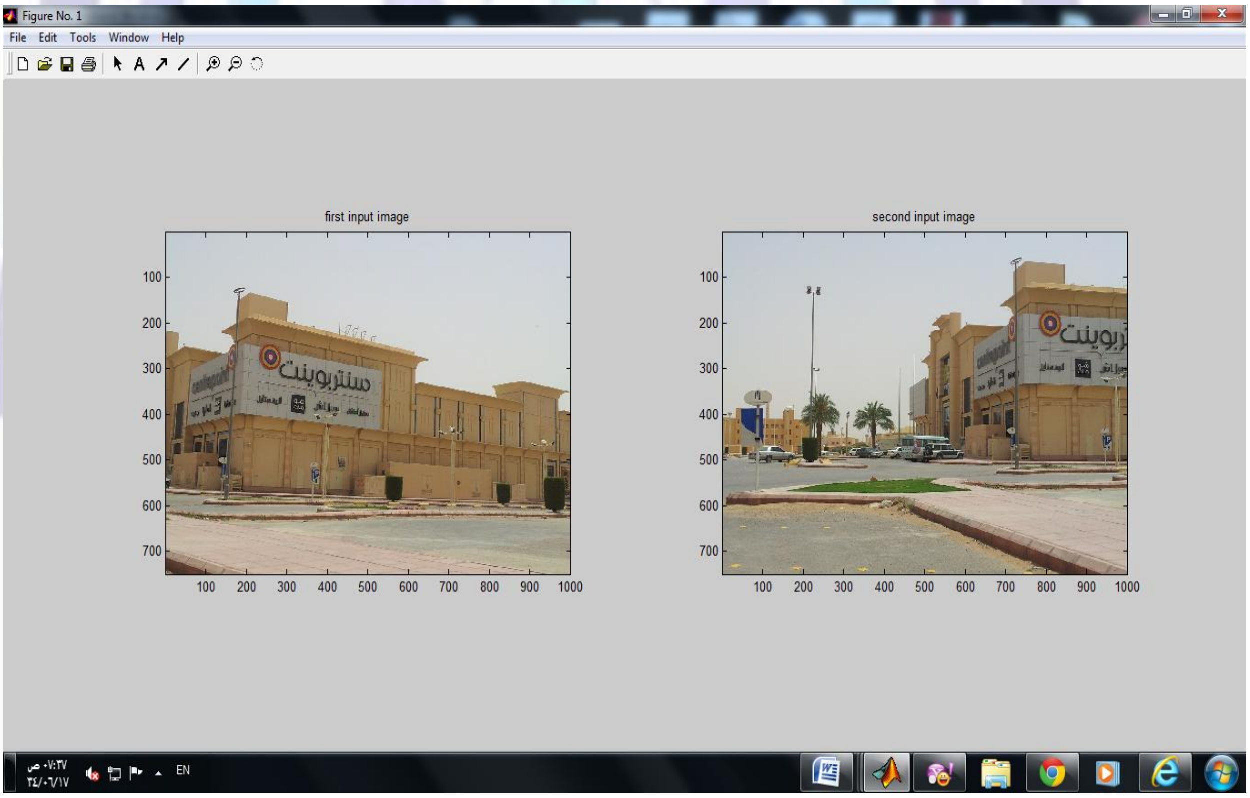

Fig 8a: Input (First image and Second image)

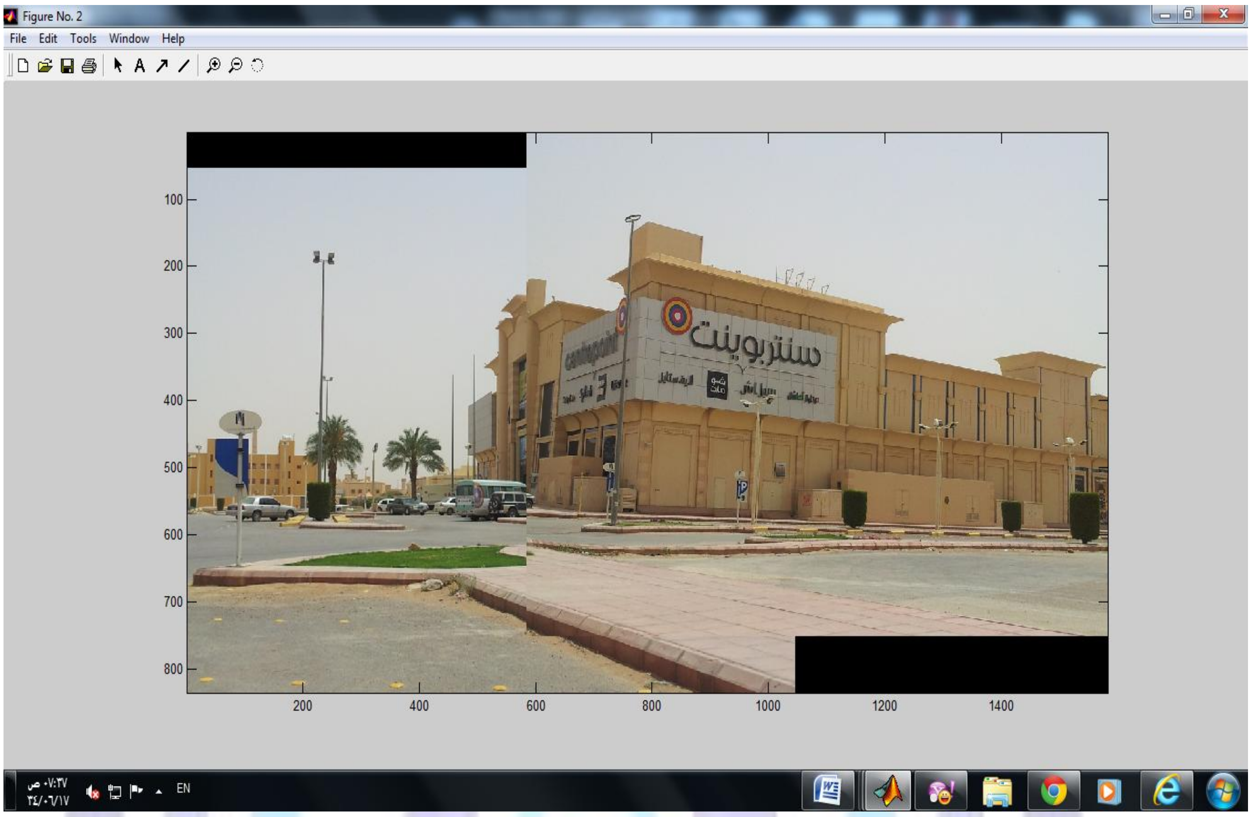

Fig 8b: Mosaic image after we apply the model of mosaicing Panorama

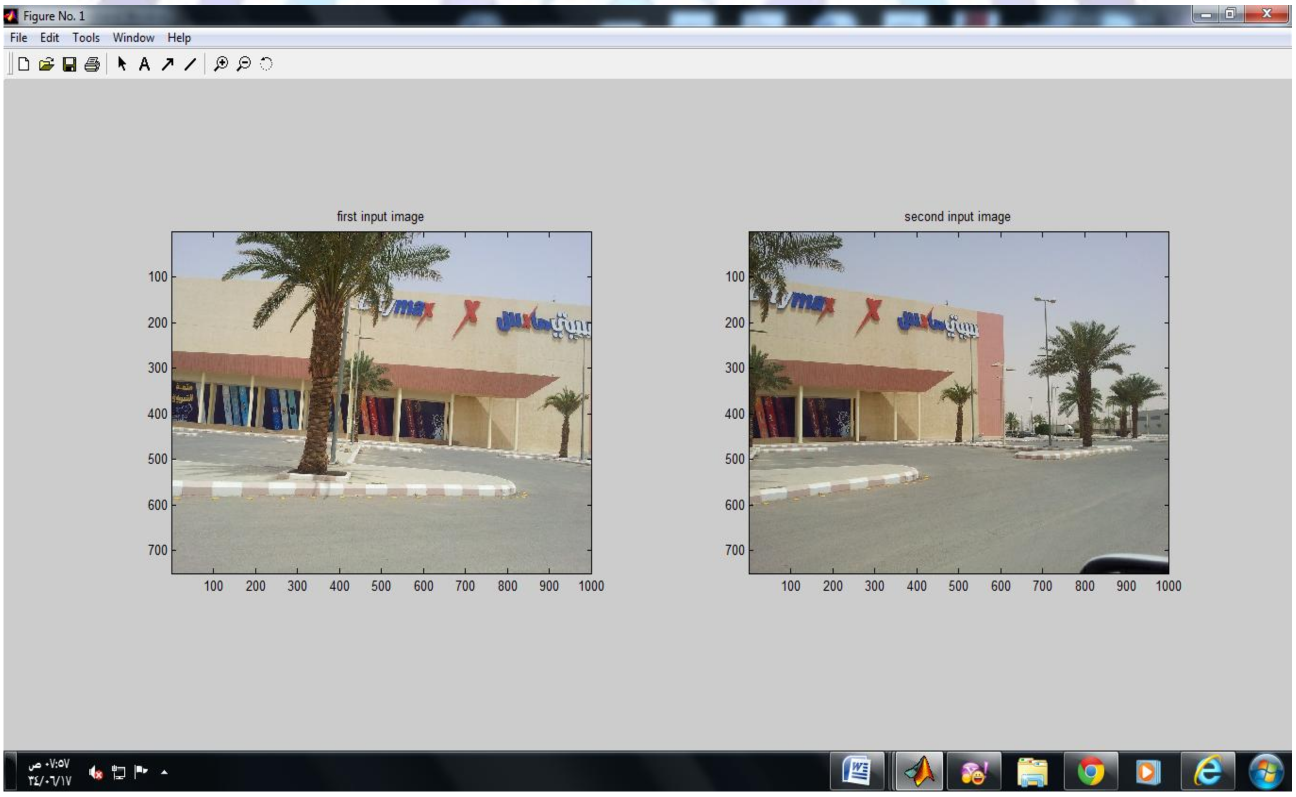

Fig 9a: Input (First image and Second image)

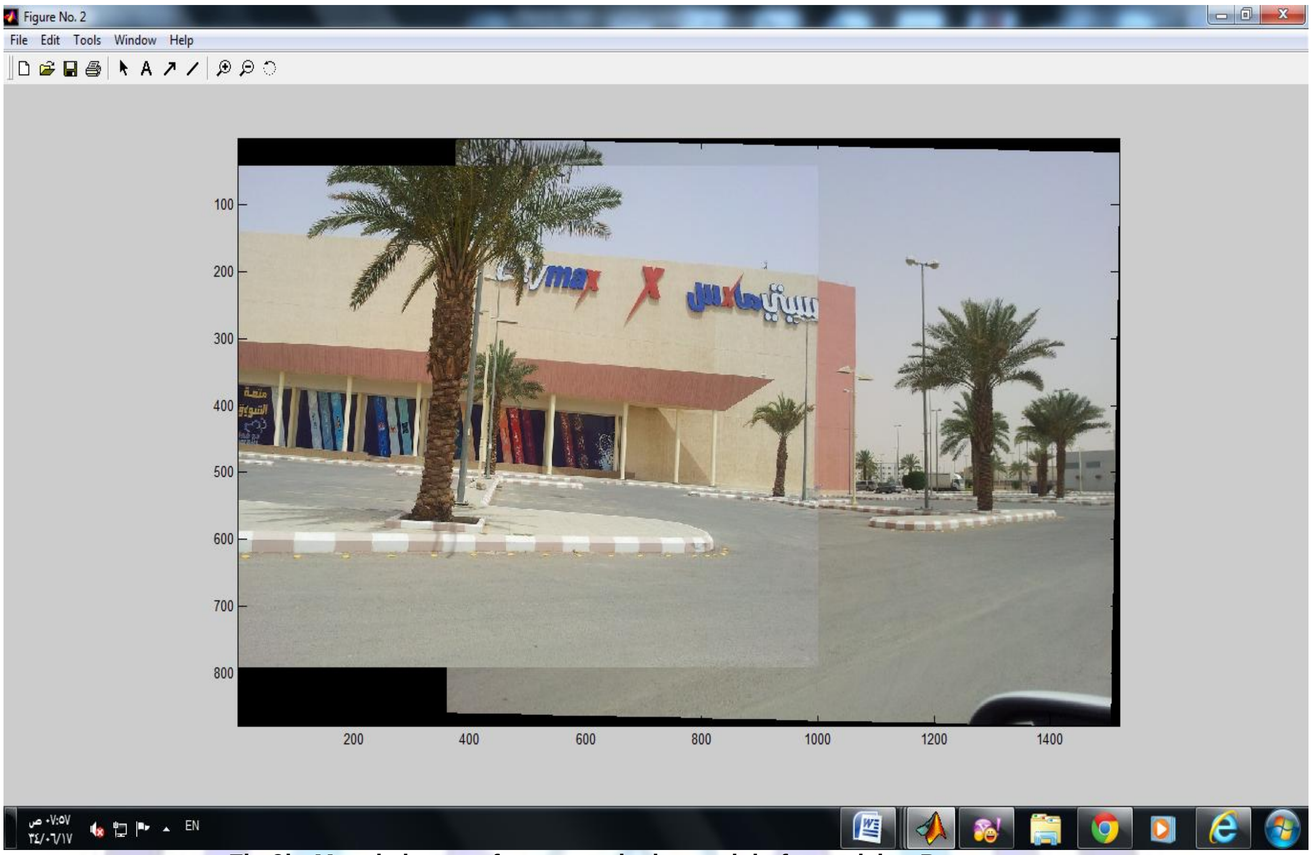

Fig 9b: Mosaic image after we apply the model of mosaicing Panorama.

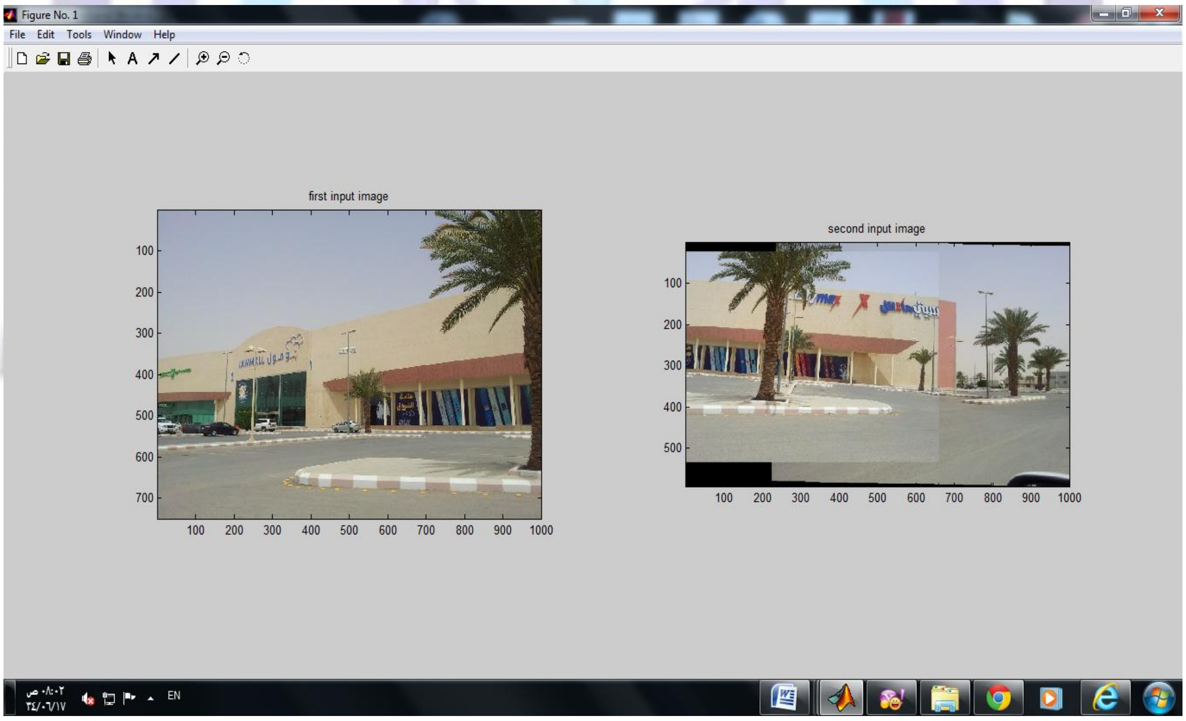

Fig 10a: Input (First image and Second image)

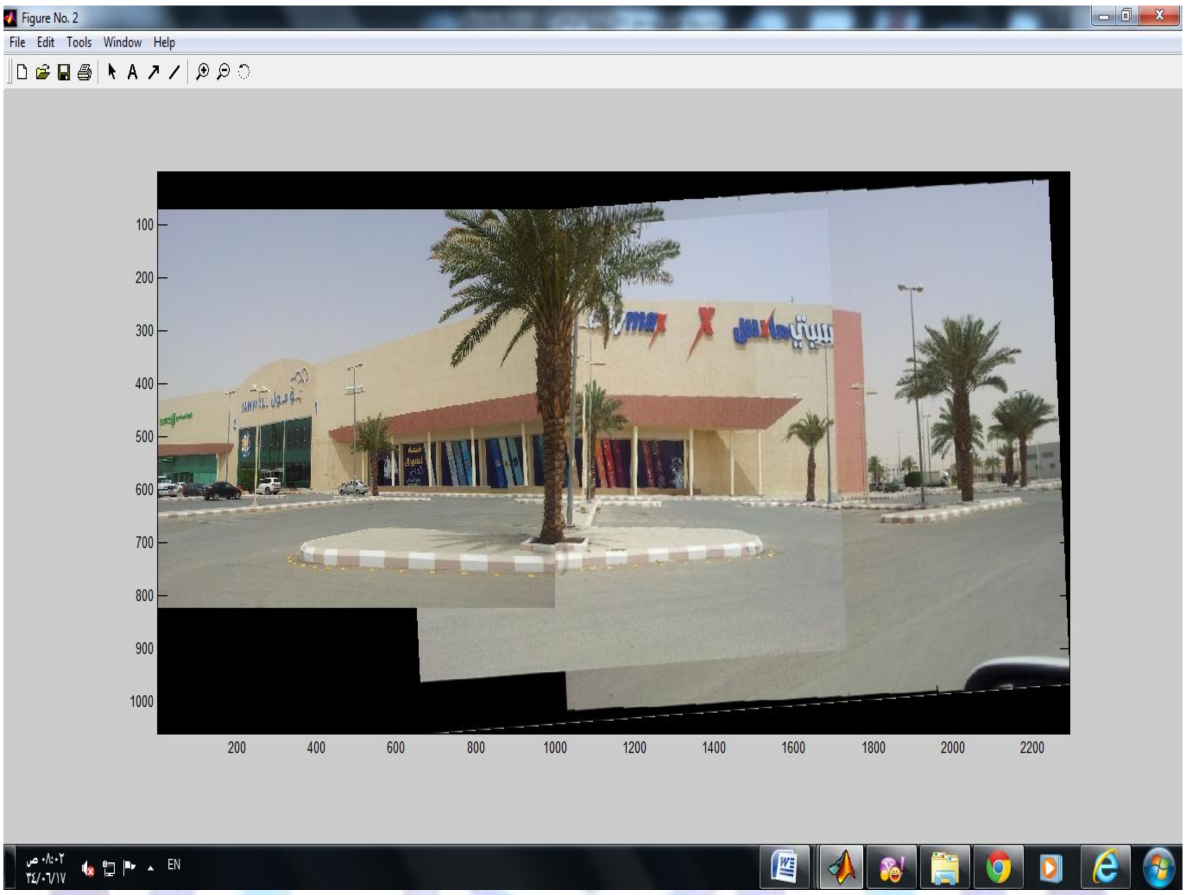

Fig 10b: Mosaic image after we apply the model of mosaicing Panorama.

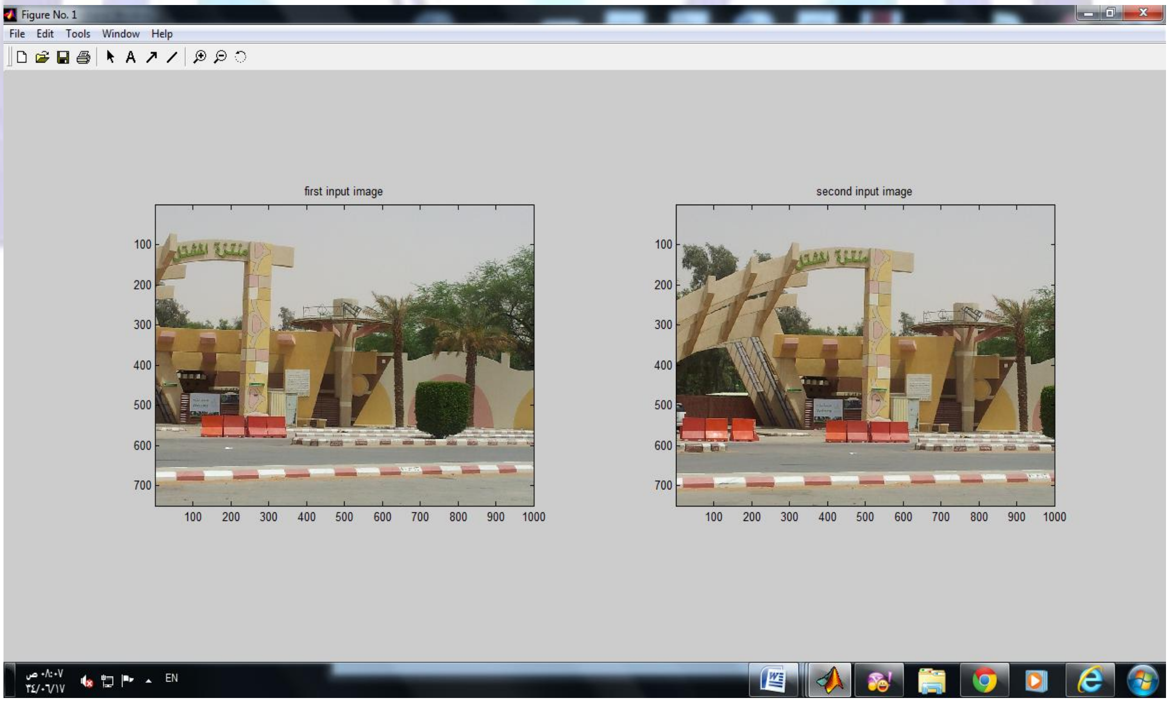

Fig 11a: Input (First image and Second image)

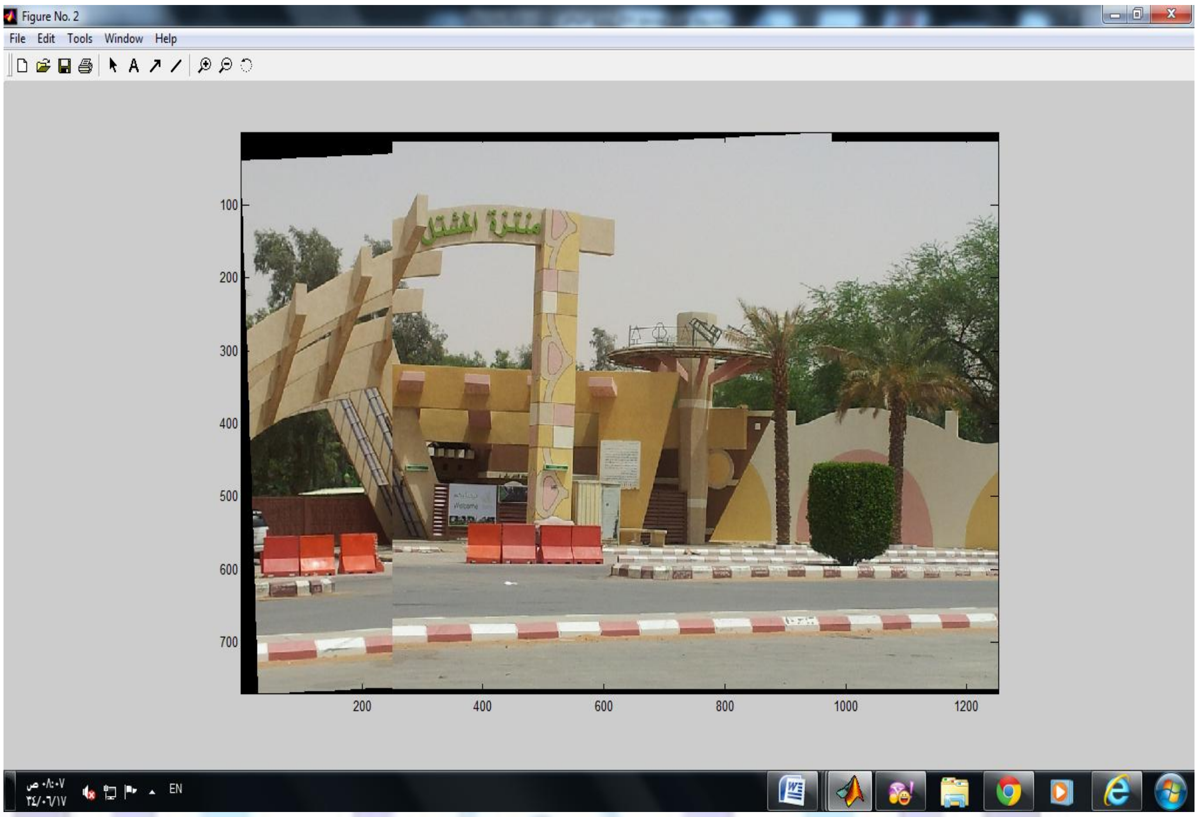

Fig 11b: Mosaic image after we apply the model of mosaicing Panorama

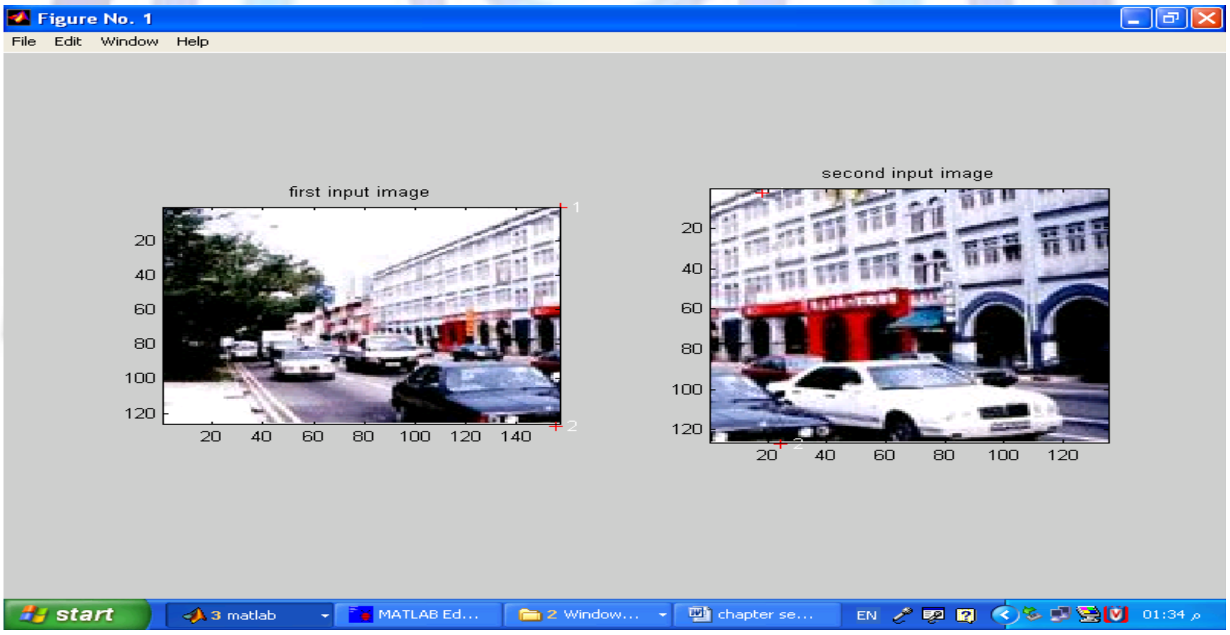

Fig 12a: Input (First image and Second image)

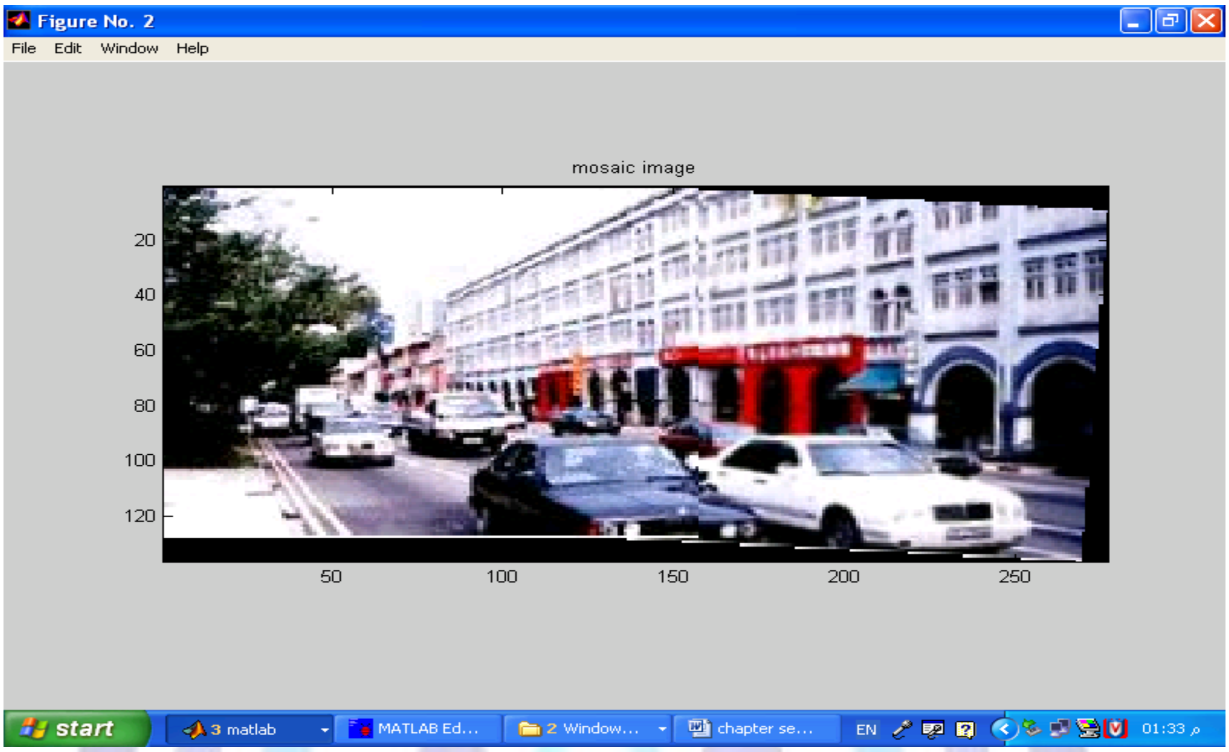

Fig 12b: Mosaic image after we apply the model of mosaicing Panorama

Results

We show here some examples of panorama photos generated from both still photos by using warping transformation method (Affine, Bilinear, Projective, Mosaic, Similarity transformation). Fig. 5 to Fig.12 are generated from both still photos and Fig. 4 is generated from the one still image by sing OpenGL codes using model. It is worth to mention that in both Fig.5 to Fig.12, overall brightness not changes from images to images, intensity correction is done and blending is added to smooth out the transition between the images.

we obtain a smooth panorama photo without any visually disturbing artifacts. This approach does not prevent a moving object from appearing more than once in the image. But then multiple appearances make the picture more dynamic and more interesting. We have good method to find third dimensional from two dimensional as panorama and more acquitted.

Conclusions

This paper presents techniques to handle some practical issues when generating panorama photos. Realizing the fact that there would always be some misalignments between two images no matter how well the matching is done, we propose a warping transformation method that finds a line of best agreement between two images, to make the misalignments less visible bye using Affine, Bilinear, Projective, Mosaic, Similarity transformation. Also shown in this paper are methods on how to find 3D from 2D in three cases: One a single image still image. Two views from two different perspectives. panorama photos generated from both still image.

In the future, we plan to add find 3D from 2D bye anther methods.

Acknowledgements

The authors acknowledge the financial support from the deanship of scientific research at Salman bin Abdul aziz university (KSA) under project number 42/H/1433.

Reference

[1] George. Wolberg, "Digital Image Warping", IEEE Computer Society Press, Los Alamitos, California,1990.

[2] Dwayne Phillips, " image processing in c , Analyzing and enhancing digital images", Second edition, R & D Publications, 1994.

[3] A. Ardeshir Goshtasby, " 2-D and 3-D image registration for medical, remote sensing and industrial applications", Wiley & Sons, 2005.

[4] Innchyn Her, "Geometric Transformtions on the Hexagonal Grid", IEEE Trans. Image Processing, pp1213-1222, vol 4, no 9, Sept. 1995.

[5] Milan. Sonka, "Image Processing, Analysis, and Machine Vision", Brooks/Cole Publishing Company, 2nd.Ed. 1999.

[6] R. Szeliski and H.-Y. Shum, "Creating full view panoramic image mosaics and texture-mapped models," SIGGRAPH, p. 251-258, 1997.

[7] Y. Li, Li, Xu, G. Morrison, C. Nightinggale, and J. Morphett, "Robust panorama from mpeg video," vol. I, p. 81-84, ICME, 2003.

[8] H.-Y. Shum and R. Szeliski, "Construction and refinement of panoramic mosaics with global and local alignment," ICCV, p. 953-58, 1998.

[9] B. Rousso, S. Peleg, I. Finci, and A. Rav-Acha, "Universal mosaicing with pipe projection," ICCV, p. 945-952, 1998.

[10] S. Peleg and J. Herman, "Panoramic mosaics with VideoBrush," DARPA Image Understanding Workshop, pp. 261-264, May 1997.

[11] E. A. Maxwell, "The methods of plane projective geometry", based on the use of general homogeneous coordinates, Cambridge U. Press, London, 1946.

[12] I. D. Faux, M. J. Pratt, "Computational Geometry for design and manufacture", Ellis Horwood Ltd.,Chichesterm England, 1979.

[13] S. Peleg and J. Herman, "Panoramic mosaics by manifold projection," IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 338-343, June 1997.

[14] J.W. Hsieh, "Fast stitching algorithm for moving object detection and mosaic construction," vol. I, p. 85-88, ICME, 2003.

[15] C. A. Glasbey and K. V. Mardia, "A review of image warping methods", Journal of Applied statistics,25(2):155-172., 1998.

[16] Ismail A. Ismail, S. A. Zaki, M. A. Ashabrawy, E. Raka " Crack detection using image processing ", Suez canal university, 2010.

[17] Richard Hartley and Andrew Zisserman, "Multiple View Geometry in computer vision", second edition,2004.

[18] C. A. Glasbey and K. V. Mardia, "A review of image warping methods", Journal of Applied statistics, 25(2):155-172., 1998.

[19] James D. Foley, Andries van Dam, "Fundamental of interactive computer graphics", Addison- Wesley,Reading, Ma, 1982.

Author' biography with Photo

Dr. Mohamed Abdel Fatah Ashabrawy obtained B.Sc. Computer Science in 1996, M.Sc. Computer Science in 2002 and Ph.D Computer Science, Image processing, Medical image processing in 2010 from the Department of Computer Science, Suez canal University, Assistant Professor in Salman bin Abdulaziz University, Community College, Computer Department, KSA ,Reactors Department, Nuclear Research Center , Atomic Energy Authority Egypt.