Chapter 4

A METHOD FOR IMAGE WARPING

Objective

1. To localize the facial landmark points (FLP) in given face images of source and target,
2. To form triangulations with FLP in source and target images,
3. To find a warping method with mapping of every pairs of triangles of source and target.

Image warping (Figure 4.1) is the geometrical mapping of an image from one spatial domain (the source shape) onto another (the target shape)[12]. In the processing of image warping source and target shapes are defined by some feature points. By the given corresponding feature points between two shapes, a relational mapping function has been derived which determines a mapping from the source to the target image. This derived function determines the complete rule that identifies how an image is warped from one image to another. In general, warping is a continuous mapping technique of two 2-D plans.

Figure 4.1: Warping of a man to a cat [81].

Though an image can be warped in various ways, in actual warping, pixels are mapped and transformed from one image (source) to another image (target) without changing the colors of source.

There are two ways to generate an image for warping, they are known as forward mapping and inverse or reverse mapping (Figure 4.2). In forward mapping, all the source pixels may not be transformed to the mapped target image, because some pixel co-ordinates of target image may not be mapped from source image. But in reverse mapping, it is guaranteed that all the pixel co-ordinates of target image will be mapped to the source image, whereas some pixel co-ordinates of source may be ignored. The differences have been shown in Figure 4.2.

Image warping has been mainly used in the area of remote sensing, medical imaging, computer vision, and computer graphics. This technique has been applied in real-time video effects for the movie and television industry. Some examples of warped image are shown in Figures 4.1, 4.3 and 4.4.

Exiting warping methods [12, 14, 26, 58, 80] are basically based on (a) affine transformations and (b) projective transformations. Where affine transformations are combination of scaling, rotation, shearing, translation and reflection (Figure 4.5). The main properties of affine transformation are as follows:

1. Origin is not necessarily transformed to origin,

2. Line remains same as line,

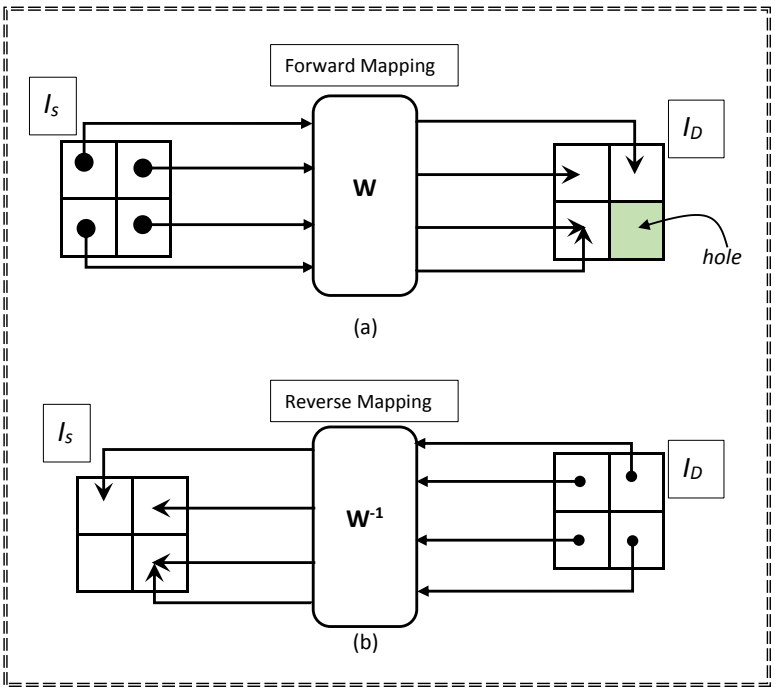


Figure 4.2: Forward versus reverse mapping [24].

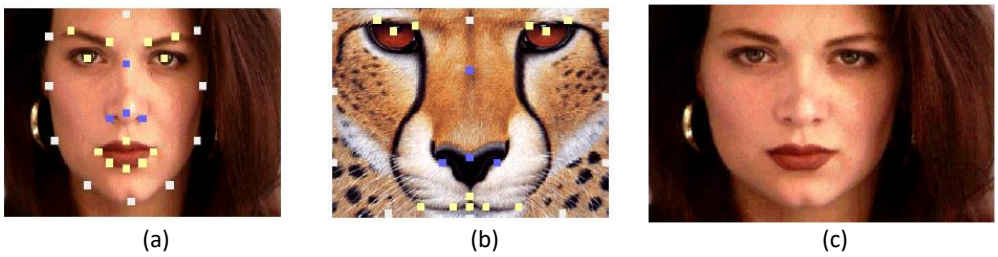


Figure 4.3: Warping of an woman to a tiger [14]. (a) Source, (b) target and (c) warped image. Corresponding landmark points of source and target images are also shown in (a) and (b).

3. Parallel line remains same as parallel line,

4. Ratios are preserved.

Again projective transformations are combinations of affine and projective transformations. The main properties of projective transformations are as follows:

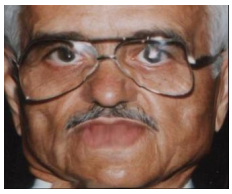
1. Origin is not necessarily transformed to origin,

2. Line remains same as line,

3. Parallel line is not necessarily transformed to parallel line,

4. Ratios are not preserved.

Source & Target



Warped

Figure 4.4: Warping of source to target image.

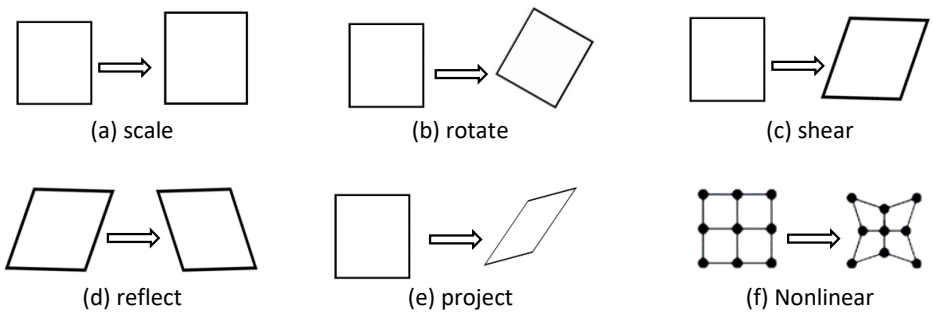


Figure 4.5: Shape Transformations [48].

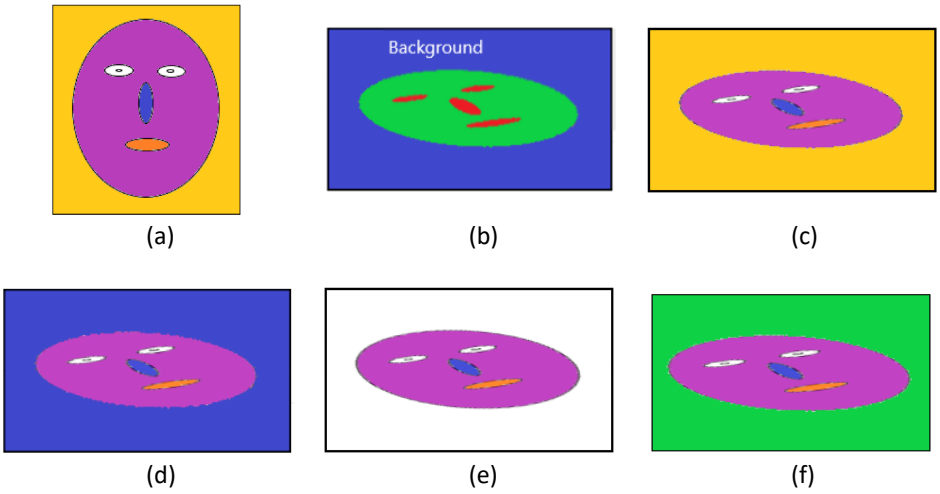


Figure 4.6: Warping images (a) source input, (b) target shape, (c) warping with existing method, complete source image has been warped, color of the source remains same, (d) - (f) warping with proposed method. (d) Warped image with same background (outer region of face) of the target, only a portion of the source image has been warped, color of the remaining portion (background) is same as target image, (e) warped image without background, only a portion of the source image has been warped, color of the remaining portion (background) is blank and (f) warped image with different background, only a portion of the source image has been warped, color of the remaining portion (background) is other than of target image.

All the major warping methods are included in the popular Matlab function *imwarp*() [33]. Existing methods maps all the points of the source to the target image. But in some particular cases some one may need to transform a selective portion of a source to target image. Also, in the model of our proposed work, we need warped images which can have background (outer region of face) of a target image or without any background. It is little bit difficult to get such a result with existing warping methods, therefore we have proposed a model which can satisfy our requirement (Figure 4.6). The proposed warping method has been described in the successive sections.

In our work we have used warping technique to get the older aged morphed synthesized images of a child image. The technique of proposed morping method using warped images has been described in next chapter.

4.1 Proposed Warping Method

We have proposed a mapping technique which maps one triangle to another and establish a relation of each pixel of one triangle to another. Then we apply this triangle based mapping to map two face images by dividing the total face image

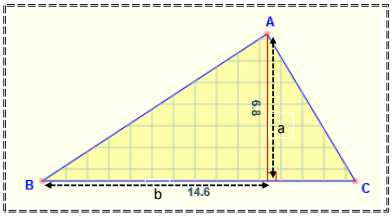


Figure 4.7: Area of a triangle, $Area = \frac{ba}{2}$.

to some specific numbers of triangles[1]. For mapping of two triangles we followed inverse/ reverse mapping technique, where mapping of every pixel of the target triangle is optimised, whereas mapping of some pixels of source triangle may be ignored. This approach will connect all the pixel positions of the target triangle with the corresponding pixel positions of the source triangle.

This method describes a triangle based algorithm to transform a source image into a target image. In this work, a digital source face image is mapped to a target face image to transform the shape of source into the target. This work is done with six steps. The initial step of the stated algorithm takes source and destination face images as input from the specified location or through the webcam. The second step deals with finding the 68 landmark or facial feature points of input images for tracking the features such as eyes, mouth, nose, lips, ears and face. The third step generates fixed 116 nos of non-overlapping triangles from 68 landmark points (which has been found in the second step) for both the input face images. In forth step one mapping link is established between each pair of corresponding 116 triangles of both the images. Then these pairs of triangles are divided on the basis of given threshold value of the in-radius of the triangle pairs. In this step a set of smallest subtriangles are found in the last level for the triangle pairs. In fifth step each pair of smallest sub-triangles are mapped with pixels from source face image to destination face image and then generate intermediate image with color interpolation. The sixth step is the process of assembling the 116 triangles to generate the resultant face image in the shape of target face image.

Incirle, Incenter, and Inradius of A Triangle

Incircle: A circle which is also known as "inscribed circle", which is the largest circle that will fit inside a triangle is called incircle of that triangle, where each of the three sides of the triangle is a tangent to the circle (Figure 4.8).

[1]The 116 triangulation of a face image is visualized in Figure 3.10 in the previous chapter.

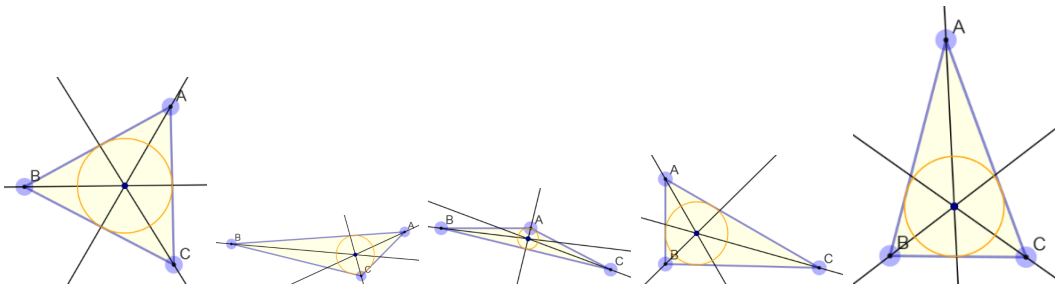


Figure 4.8: Incenter, incirle and inradius of different types of triangles.

Incenter: The position of the center of the incircle of a triangle is called incenter of that triangle, *i.e.* the point where the angle bisectors meet.

Inradius: The radius of the incircle is called inradius. The radius is given by the formula:

$$r = \frac{2a}{p} \tag{4.1}$$

where a is the area of the triangle, and p is the perimeter of the triangle, the sum of its sides.

Area of a triangle: The number of square units it takes to exactly fill the interior of a triangle is called area of the triangle, usually it is a "half of base times height", the area of a triangle is given by the formula below.

$$Area = \frac{ba}{2} \tag{4.2}$$

where b is the length of the base, and a is the length of the corresponding altitude (Figure 4.7).

http://www.mathopenref.com

4.1.1 Triangle to Triangle Mapping

In the proposed triangle based mapping technique, one triangle is mapped to another one. Here source triangle $T_1 = \Delta P_1P_2P_3$ and target triangle $T_2 = \Delta Q_1Q_2Q_3$ are taken as input, these inputs may be of different sizes, or different shapes.

Initially both the triangles T_1 and T_2 are one-one mapped by their incenters by default. If the inradius value of both the triangles are grater than a given thresh hold value ϵ, where $0 < \epsilon < 0.5$, then we parallelly divide both the triangles T_1 and T_2 into four sub triangles by the middle points of each side of them. Otherwise we map the triangles T_1 and T_2 with a technique as described in the subsection

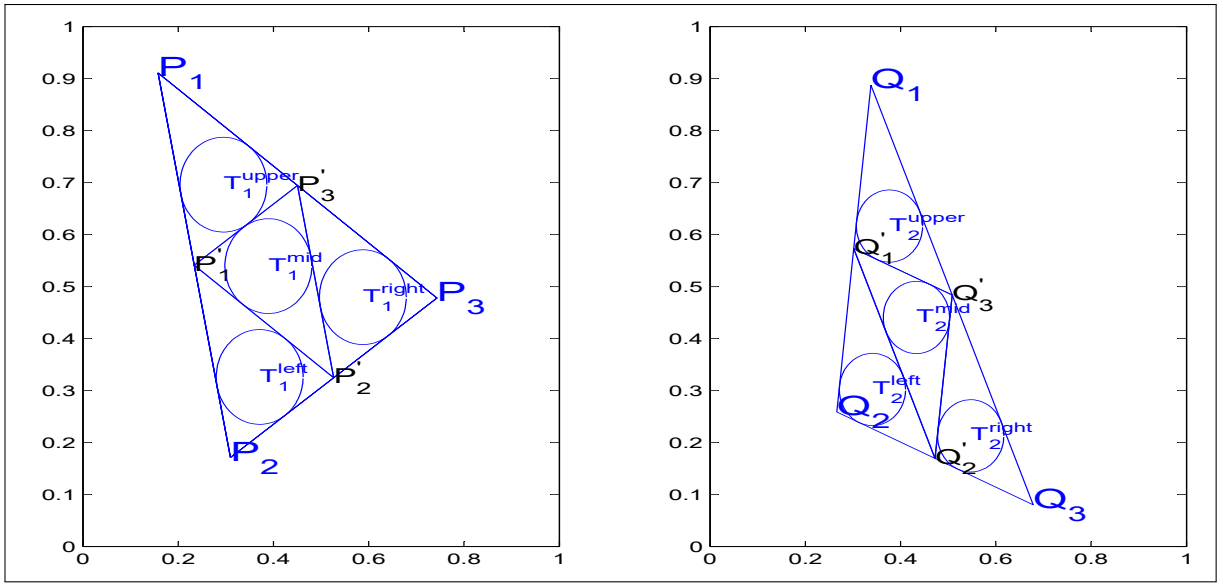


Figure 4.9: Sub-triangles of two given triangles T_1 and T_2

4.1.5. In this way we repeat the same technique for all the pairs of new subtriangles recursively and whenever one triangle is obtained with inradius value less than ϵ then mapping is computed with them.

The four sub triangles which are obtained from a given triangle are denoted by upper, left, right and middle as shown in (Figure 4.9). In every process of subtriangulations we get four sub triangles $T_2^{upper}, T_2^{left}, T_2^{right}, T_2^{middle}$ of triangle ΔT_2, and corresponding sub triangles $T_1^{upper}, T_1^{left}, T_1^{right}, T_1^{middle}$ of triangle ΔT_1.

After getting the subtriangles, we compute the in-radius of subtriangle T_2^{upper} of triangle T_2 and for corresponding subtriangle T_1^{upper} of triangle T_1 (Figure 4.9) and if either one is less than the given threshold value ϵ then we record the one-one mapping point of the in-centre of T_2^{upper} and T_1^{upper}, and map the pixel of source ant target, (pixel mapping technique is given in the subsection 4.1.5). Otherwise we consider T_2^{upper} and T_1^{upper} as the main triangles and repeat the same technique recursively. Similarly we apply this recursive technique for all other pair of sub triangles $(T_2^{left}, T_1^{left}), (T_2^{right}, T_1^{right})$ and $(T_2^{middle}, T_1^{middle})$.

And finally we get the record of mapping information of all the pixel points located in target triangle which are connected with the pixel points in the source triangle. This complete process has been explained with the master procedure $WarpObject(Obj_1, Obj_2)$ in section 4.1.2, and child procedures $Subtri(T)$ 4.1.4 and $MappTriangle(T_1, T_2)$ 4.1.3.

Notations: The following few notations are used in the following procedures.

- Object $Obj = \{T_1, T_2, \ldots, T_n\}$ is a set of n numbers of non-overlapping triangles.
- Triangle $\Delta T_1 = \{P_1, P_2, P_3\}$, where $P_i = (x_i, y_i), i = 1 \ldots 3$ are vertices of ΔT_1.
- Triangle $\Delta T_2 = \{Q_1, Q_2, Q_3\}$, where $Q_i = (x_i, y_i), i = 1 \ldots 3$ are vertices of ΔT_2.

4.1.2 Procedure $WarpObject(Obj_1, Obj_2)$

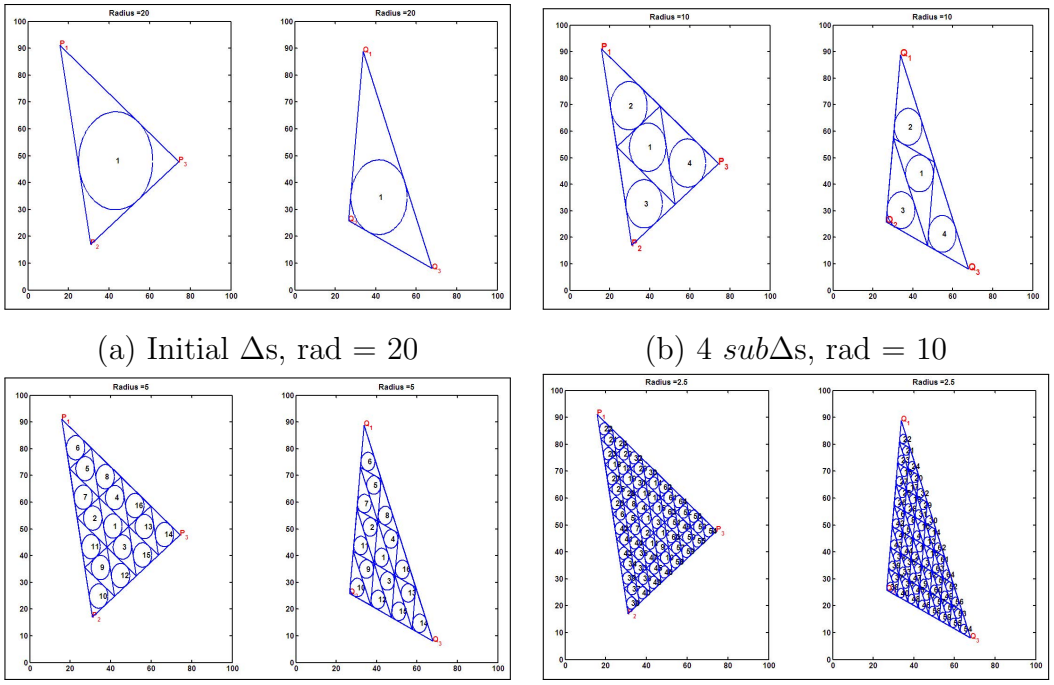
The procedure $WarpObject(Obj_1, Obj_2)$ presents the proposed warping method for two given objects Obj_1, Obj_2, where objects Obj_1 and Obj_2 are nothing but the set of finite numbers of non-overlapping triangles. This procedure is used to warp the given object Obj_1 into the another given object Obj_2, and at the end of this process Obj_1 gets the exact shape of Obj_2.

1. **Input:** Obj_1, Obj_2, for which triangle $\triangle T_1^i \in Obj_1$ and $\triangle T_2^i \in Obj_2$, $1 \leq i \leq n$, n is number of triangles
2. **Output:** Mapping of every points $Q \in Obj_2$ to $P \in Obj_1$,
3. for $i = 1 \ldots n$
   (a) find (T_1^i, T_2^i), find i^{th} pair of triangles from Obj_1 and Obj_2;
   (b) $MappTriangle(T_1^i, T_2^i)$,
4. end-for

Some steps of this process is shown in Figure 4.10, and one result is shown in Figure 4.13.

4.1.3 Procedure $MappTriangle(T_1, T_2)$

The procedure $MappTriangle(T_1, T_2)$ maps all the interior points of two given triangles T_1, T_2. Initially it finds the inradius value of both the triangle pair and if it is greater than threshold value ϵ for both the triangles then it divides them into four equal subtriangles, namely $[Upper, Left, Right, Middle]$, shown in Figure 4.9. Otherwise it goes to map the smallest triangulation process, discussed in the section



(a) Initial Δs, rad = 20 (b) 4 $sub\Delta$s, rad = 10

(c) 16 $sub\Delta$s, rad = 5 (d) 64 $sub\Delta$s, rad = 2.5

Figure 4.10: Steps in Triangle Mapping Technique, read Δ as triangle.

4.1.5. Then recursively call the same function for these four pair of subtriangles, and recursive process continues until it finds atleast one subtriangle whose inradius value is smaller than the given thresh hold value ϵ. This process is shown in Figure 4.11.

1. **Input:** $\triangle T_1$, $\triangle T_2$: vertices (P_1, P_2, P_3) and (Q_1, Q_2, Q_3) of two triangles T_1 and T_2.

2. **Output:** Corresponding points sets of all the interior points of two triangles T_1 and T_2.

3. $r_1 = inRadius(\triangle T_1)$;

4. $r_2 = inRadius(\triangle T_2)$;

5. if $r_1 > \epsilon$ & $r_2 > \epsilon$

   (a) Find a set of subtriangles $\{T_1^{Upper}, T_1^{Left}, T_1^{Right}, T_1^{Middle}\}$ of T_1 using procedure $Subtri(T_1)$

   (b) Find a set of subtriangles $\{T_2^{Upper}, T_2^{Left}, T_2^{Right}, T_2^{Middle}\}$ of T_2 using procedure $Subtri(T_2)$

(c) for $Pos = Upper, Left, Right, Middle$

- recursively use self procedure for two subtriangles T_1^{Pos}, T_2^{Pos} using $MappTriangle(T_1^{Pos}, T_2^{Pos})$.

(d) end-for

6. else

(a) Find corresponding points sets of all the interior points of two triangles T_1 and T_2 using procedure $PixelMap(T_1, T_2)$, (Section 4.1.5)

7. end-if

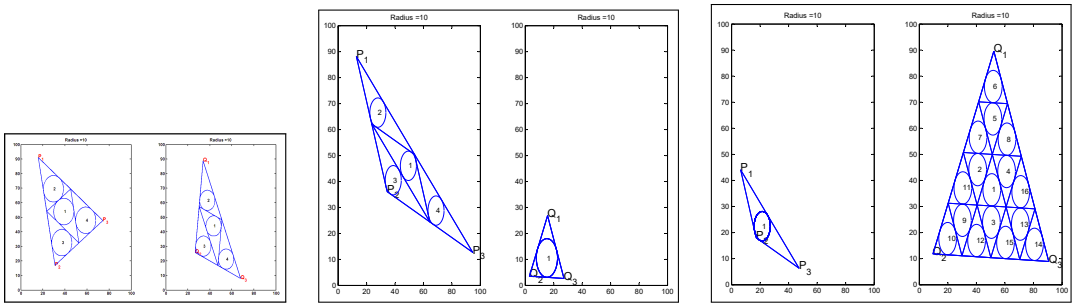
4.1.4 Procedure $Subtri(T)$

Procedure $Subtri(T)$ divides a given triangle $T = \{P_1, P_2, P_3\}$ into four subtriangles named $T^{Upper}, T^{Left}, T^{Right}, T^{Middle}$ by equally dividing each side of the triangle by 2.

1. **Input:** $T = \{P_1, P_2, P_3\}$, A triangle T with three vertices P_1, P_2, P_3
2. **Output:** A set of subtriangles $\{T^{Upper}, T^{Left}, T^{Right}, T^{Middle}\}$ of T, Figure 4.9
3. for $i = 1 \ldots 3$

o $P_i' = \frac{P_i + P_{(i+1)\%3}}{2}$, midpoints of three sides of triangle T.

4. end-for
5. $T^{Upper} = \{P_1, P_1', P_3'\}$
6. $T^{Left} = \{P_1', P_2, P_2'\}$
7. $T^{Right} = \{P_2', P_3, P_3'\}$
8. $T^{Middle} = \{P_1', P_2', P_3'\}$
9. return $\{T^{Upper}, T^{Left}, T^{Right}, T^{Middle}\}$, set of four subtriangles.



(a) (b) (c)

Figure 4.11: Three cases in Triangle Mapping Algorithm, when any inradius $< \epsilon$ for a pair of triangles of source and target. (a) bothe inradii are $< \epsilon$ for both the triangles, (b) one smallest triangle in target but more than one subtriangles in source, and (c) more than one subtriangles in target but one smallest triangle in source.

4.1.5 Pixel Mapping $PixelMap(T_1, T_2)$

In the pixel mapping procedure $PixelMap(T_1, T_2)$, two triangles source T_1 and target T_2 are taken as parameter for the mapping process. This procedure will be invoked when the inradius of any one of these triangles is less then the value of given thresh hold ϵ, which mean atleast one triangle is in the state of smallest subtriangle and no further subtriangulation is accepted.

Here, three cases will arise (A) both the triangles are smallest and equal in inradius, (B) first triangle is smallest whose inradius is $< \epsilon$ and (C) second triangle is smallest whose inradius is $< \epsilon$. These cases are shown in Figure 4.11.

Let, r_1, r_2 are inradii, and c_1, c_2 are incenters of two triangles T_1 and T_2 respectively. The coordinate points (x_1, y_1) and (x_2, y_2) represent c_1 and c_2 where values of these (x, y) coordinates are real numbers. Then the said three cases can be handled with the following steps:

1. **Case–A:** $r_1 < \epsilon$ and $r_2 < \epsilon$
   Transfer pixel from c_1 to c_2 using $PixelTarnsfer(c_1, c_2)$

2. **Case–B:** $r_1 < \epsilon$ and $r_2 > \epsilon$
   Transfer all pixels from c_1 to all the interior points $c_2^i \in T_2$ using $PixelTarnsfer(c_1, c_2^i)$, interior points of a triangle can be found from all the smallest subtriangles[2].

3. **Case–C:** $r_1 > \epsilon$ and $r_2 < \epsilon$
   Transfer pixel from c_1 to c_2 using $PixelTarnsfer(c_1, c_2)$

[2]To get smallest subtriangles of a given triangle, recursively divide the triangle until *inradius* of a subtriangle $> \epsilon$, where $0 < \epsilon < 0.5$.

In the next step, $PixelTarnsfer(c_1, c_2)$ a procedure for transfer pixel from c_1 to c_2 has been described.

4.1.6 Transfer pixel from c_1 to c_2: $PixelTarnsfer(c_1, c_2)$

This procedure describes the technique to transfer pixel from position $c_1 = (p, q)$ to $c_2 = (p', q')$, incenters of two triangles; where values of these coordinates are real numbers. Basically, this process will copy the pixel values from the position located in source point (p, q) to the position located in target point (p', q').

Let, S and $T' = T$ are source and copy of the target image respectively, and $\triangle T_1 \in S$, $\triangle T_2 \in T'$; and $S[i, j], T'[i, j]$ represent $(i, j)^{th}$ pixel value of the respective images, and i, j are integer numbers. The technique for pixel transfer from c_1 to c_2 are stated in the following steps:

1. Find $(M, N) = NearestPixels(p, q)$[3], nearest pixel coordinates of (p, q)
2. Find $(M', N') = NearestPixels(p', q')$, nearest pixel coordinates of (p', q')
3. if (M', N') represents single pixel of T' and $T'[M', N']$ is already marked as finalized, then

   - No updation is required, simply return from this process.

4. if (M', N') represents single pixel of T', then

   - $T'[M', N'] = Average(S[M, N])$,
   - mark $T'[M', N']$ as finalized, *i.e.* for this point (p', q') in target image T' no updation is required,
   - return from this process.

5. For all $(i', j') \in [M', N']$
   if $T'[i', j']$ is not updated till now, then

   - $T'[i', j'] = Average(S[M, N])$,
   - mark $T'[i', j']$ as updated,

[3]Since pixel values of an image locate in integer coordinate positions, therefore we have to find floor and ceiling values of (p, q) to locate the nearest pixel coordinates.

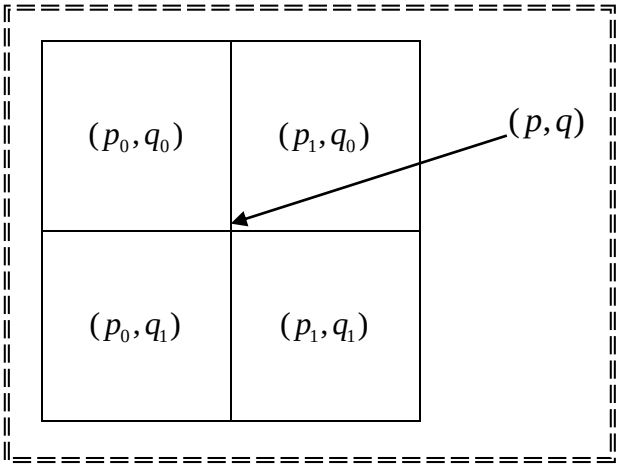


Figure 4.12: Nearest pixel's integer coordinates: (p, q) are real numbers, $p_0 = \lfloor p \rfloor, q_0 = \lfloor q \rfloor, p_1 = \lceil p \rceil, q_1 = \lceil q \rceil$

The procedure "Nearest Pixel Coordinate" (Section 4.1.7) has been used to find the nearest set of pixel's positions (X, Y) (Figure 4.12), *i.e.* coordinates in integer number of a given point (p, q), where p, q are real numbers.

4.1.7 Nearest Pixel Coordinates: $NearestPixels(p, q)$

This procedure finds the nearest integer coordinates of a given point (p, q), where p, q are real numbers (Figure 4.12). Let, $p_0 = \lfloor p \rfloor, q_0 = \lfloor q \rfloor, p_1 = \lceil p \rceil, q_1 = \lceil q \rceil$, then nearest integer coordinates can be found by the following steps:

1. if p and q both are integer numbers then return (p, q);
2. if p is integer but q is not a integer number, then (p, q_0) and (p, q_1);
3. if q is integer but p is not a integer number, then return (p_0, q) and (p_1, q);
4. if p and q both are not integer numbers, then return $(p_0, q_0), (p_0, q_1), (p_1, q_0)$ and (p_1, q_1);

4.2 Database used for Experiments of Image Warping

A. S. Georghiades et al. [25] have presented the **"Extended Yale Face Database B"** which contains 16128 images of 28 human subjects. There are 9 different poses and 64 illumination conditions are available for every subject present in the dataset

(a) Source (b) Target

(c) Original into the shape of target (d) Output into the shape of target

Figure 4.13: Warping source image into the shape of the target image

(in total 28 persons × 9 poses × 64 illumination *i.e.* 16128). The Extended Yale Face Database B images are open for any research purposes. A sample dataset has been shown in Figure 4.15.

4.3 Experimental Results of Image Warping

We have chosen source and target images from the "Extended Yale Face Database B" and warped the source image into the 8 different poses of the target image. In the dataset, in total 9 different poses are available for all the images. Some warped output images are shown in Figure 4.14. We have tested with randomly selected 150 number of experiments for proposed and existing *imwarp*() function of Matlab [33], and found 150 pairs of warped images (W_i^1, W_i^2) for the input of S_i for a particular pose (shape), where W_i^1 and W_i^2 are the warped images of proposed and existing *imwarp*() function respectively. Thus for every experiment Exp_i we have source input S_i, target pose or shape T_i, ground truth G_i and warped images (W_i^1, W_i^2) where $i = 1, 2, \ldots 150$. After warping the source into target pose, the similarity has been computed with the respective original poses of the source. To compare two images we have used a face recognition tool, which has been implemented with OpenCV and Python (the implementation details has been described by R. Raja

Figure 4.14: Result of warping of source to target. The source image has been warped into 8 different poses, and compared with the originals in the respective poses (Extended Yale Face Database B [25]).

Figure 4.15: Extended Yale Face Database B of 28 different subjects. [25]

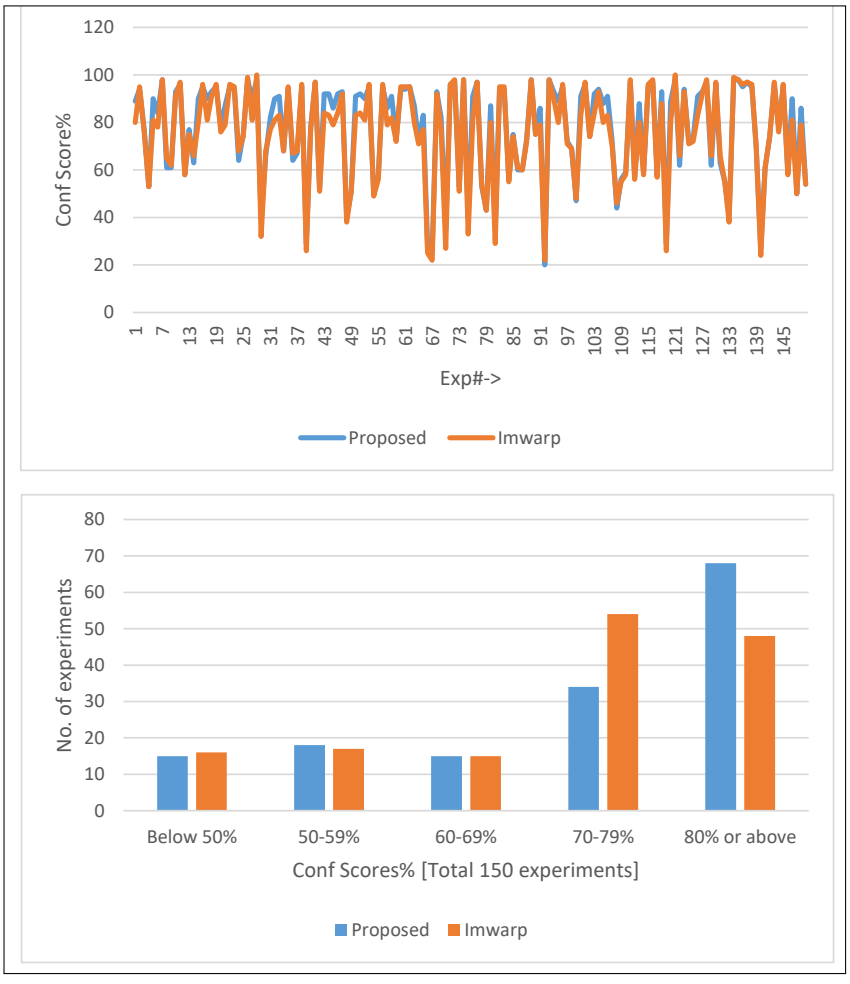


Figure 4.16: The comparative results of proposed and existing *imwarp*() Matlab method [33] based on confidence score of image recognition (L. Dinalankara [17]). In the top figure, confidence score of 150 individual experiments have been shown, and number of experiments on different confidence score have been shown in the bottom figure.

[51] and L. Dinalankara [17]) (Section 6.1.2 in Chapter 6).

Using the said face recognition tool, we have compared each pair of ground truth and warped images (G_i, W_i^1) and (G_i, W_i^2) and recorded confidence scores C_i^1 and C_i^2 of proposed and *imwarp*() Matlab function.

The comparative results of proposed and existing *imwarp*() Matlab method [33], which are based on confidence score returned by said face recognition tool are shown in Figure 4.16. We have found that the similarity (80% or above) based on confidence score of image recognition is higher than that of world-wide acceptable imwarp function, and below 80% is almost same. The comparative result analysis has shown that proposed method can produced acceptable warped images.

4.4 Conclusion

In this proposed method, we represent a pair of given face images (source and target) by some fixed number of triangles. These triangles represent the same facial parts of both the images. We find the 68 landmark points (discussed in previous chapter) for the image pair and each images are divided by 116 fixed triangle sets from this 68 landmark points. With 116 iterations, 116 pairs of corresponding triangles are chosen from both the images, and proposed triangle based mapping algorithm has been applied to get record of mapping information of each pair of triangles. Finally with this mapping information, one image is warped into another image, where source image gets the shape of target image, i.e. the shape of first image is transformed into second images.

The proposed triangle wise warping method processes divide and conquer approach to get the final output. In this technique whole source and target images are divided by smaller triangular shaped images based on the given facial feature points. Then the smaller triangular shaped images are warped with the proposed technique. Finally all the smaller shaped warped images are merged to get the complete warped image of the source.

The methodology of our proposed warping method is very easy to implement and works as per our expectations. Another advantage of this method is that it can find the mapping of a selective portions of source and target images by which a required segment of a source image can be transformed into target image. One drawback, this method includes that actual source image can not be obtained by inverse mapping.

The comparative results discussed in the previous section shows that our proposed method is acceptable. After getting the resultant warped image we apply another proposed morphing technique to get the synthesized older aged face image of the source image of younger age, which technique has been discussed in Chapter 5.