Chapter – 5

5.1 Introduction

This chapter explains the image processing techniques used in the model. The techniques used are: automatic rotation of image, resizing of image, converting color image into black and white image, and template matching for verification of images.

5.2 Automatic Rotation of Horizontally scanned palm

On the home page of the system the guidelines are given to use this prototype. The users have to scan their left and right palms from front side as well as from back side. The important thing is that, users should scan their palm vertically. That is, flat bed scanners are rectangular in shape. User should keep the palm parallel to the long axis of the scanner. As shown in figure 6.8, system is capable to manage some deviations in the orientation of palms. This means that if user cannot keep the palm exactly vertical then also, system can work efficiently. But, if the palm is scanned totally horizontal, then algorithm may not work properly. Therefore, before moving ahead in processing, the model checks the orientation of palm and if it is found horizontal, it rotates it vertically.

Before rotating the image, it should be checked whether it is horizontal or vertical. To accomplish this, a test is performed by the model. To understand the logic of rotation of palm image, consider the figure 5.1.

In figure 5.1, the four points named topmost, bottommost, rightmost, and leftmost are calculated first, using the cropping algorithm. The cropping algorithm is explained in
chapter 6, at section 6.2.2. The distance between topmost and bottommost point gives height of palm and the distance between rightmost and leftmost point gives width of palm. If the height of palm area is less than the width, then the image of palm is horizontal. If the image is horizontal then the model rotates the image using the rotation filter provided by AForge.Net. Following code explains rotation of the image.

```csharp
// --> Rotating the image, if it is horizontal
{
    IntPoint temp = miny;
    IntPoint temp2 = maxy;
    // create filter - rotate for 270 degrees keeping original image size
    RotateNearestNeighbor filter = new RotateNearestNeighbor(270);
    // apply the filter
    oimg = filter.Apply(oimg);
    miny.Y = minx.X;
    maxy.Y = maxx.X;
    minx.X = temp.Y;
    maxx.X = temp2.Y;
}
```

The code shows the rotation of horizontal image at the angle of 270 degrees. After rotating the image, the coordinate values are also changed appropriately. These new coordinate values are considered for further processing.

The above code is for the image which is scanned horizontally from left to right side; that is, the base of palm is at left side, and top of palm is at right side. If the palm is scanned horizontally, from right to left, the angle of rotation will be 90 degrees.

Figure 5.2 shows result of above code. The image is scanned from right to left. First the model applies algorithm 6.1: Extraction of Palm from Background. Algorithm 6.1 is explained in chapter 6, at section 6.2.2. Once the palm is separated from background, the code rotates the image at the 90 degrees. The output image is then processed by Algorithm 6.2: Cropping Algorithm, and then further processing takes place.
Chapter – 5 Image Processing Techniques used in the Model

<table>
<thead>
<tr>
<th>Input Image</th>
<th>Output Image</th>
</tr>
</thead>
</table>

Figure 5.2: Result of Rotation of Palm

5.3 Resizing the Palm Image

The user uploads the palm images for processing. Before the model starts processing, it must be verified whether the given images are of palm or not. For verification of image, pattern matching techniques are used in the model. Traditional pattern matching techniques compare two images pixel by pixel and give the result. The major problem with this approach is that it is time consuming. To overcome this problem the model has adopted a modified approach of pattern matching. The two images which are to be compared are in black and white format to save the memory space and to increase the speed of processing. That is, the sample image of palm is stored as templates in black and white and the uploaded image is converted into black and white. Before converting image into black and white, the image is resized. The size of image is reduced by reducing the height and width of image. This process may reduce the quality of image, but the model is not concerned with the quality of image. The resizing process is carried out after executing algorithm 6.1 and algorithm 6.2 on the uploaded image.

Following code gives idea about resizing of the image.

```java
//--- Resizing the cropped image
// create filter
ResizeNearestNeighbor filterrez = new ResizeNearestNeighbor(124, 160);
// apply the filter
Bitmap newImage = filterrez.Apply(image);
newImage.Save(folder + "\" + lbluid.Text + "newImage.jpg");
```
In the above code, a filter called “ResizeNearestNeighbor” is used to resize the image in given height and width. The input image is the cropped image by the algorithm 6.2, mentioned in section 6.2.2. 124 and 160 are new height and width respectively, which are given to the image. The resized image is stored separately and used for verification process.

The result of above mentioned code is shown in figure 5.3.

As shown in figure 5.3, the resized image is reduced in height and width. The dimension of input image is 1781 X 2764, and the size is 2.98 MB. The output image has dimensions 124 X 60, and the size is 23 KB. This reduces the memory requirements as well as increases the processing speed.

5.4 Converting Image into black and white

As shown in section 5.3, the image is reduced in size for further computation. Before comparing the resized palm image with sample palm image, the resized palm image is converted into black and white image. This step further reduces the size of image by having just one bit per pixel. Therefore the template matching process executes faster with less amount of data.

There is another important reason behind converting the palm image into black and white image. Template matching algorithms compare each pixel of input image and sample image. This comparison includes color value of pixels. Different users have different skin colors. If the sample palm has a particular RGB component value for each pixel, and the
user palm is not of the same color, the template matching algorithm may not give proper result, i.e. though, the uploaded image is of palm, because of color variations of skin, the template matching functions may give wrong results. To avoid this skin color variations, all the palms are first converted into black and white. The sample palm images will also be in black and white. This increases the accuracy of results tremendously.

Following code shows the conversion of the resized image into black and white image.

```csharp
//getting image for processing
System.Drawing.Bitmap simg = new System.Drawing.Bitmap("D:\Ph.D Project\DSS\SampleImages\HNBSample.jpg");

//converting into black and white
AForge.Imaging.Filters.FiltersSequence seq = new AForge.Imaging.Filters.FiltersSequence();
seq.Add(new AForge.Imaging.Filters.OtsuThreshold());
Bitmap plimg = seq.Apply(cimg);
plimg.Save(folder + "\" + lbldis.Text + "plimg.jpg");
```

In above code, the “newImage.jpg” is the resized image. Two filters are used in this code segment. First filter uses BT709 algorithm to convert the image from color image to grayscale image and second filter uses “OtsuThreshold()” method to convert the grayscale image into black and white image. The resulting black and white image is stored as “userID + p1img.jpg”, which is compared with sample image. Figure 5.4 shows result of above mentioned code.

<table>
<thead>
<tr>
<th>Input Image</th>
<th>Output Image</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Input Image" /></td>
<td><img src="image2.png" alt="Output Image" /></td>
</tr>
</tbody>
</table>

Figure: 5.4 Result of Converting Image into Black and White

As shown in figure 5.4, the resulting image is black and white. This keeps the dimensions of image same but since the color information is reduced, the size of image is also reduced drastically. The size of input image is 23 KB, but the size of resulting image is just 1.8 KB. Because of this procedure, the template matching algorithm has to match the images whose size is around 2 KB. This increases the speed of processing.

### 5.5 Template Matching to Verify the Palm Images

Template matching is the processes to match two images of objects in the images. Here, template matching is used to compare two images. The first image is the palm image of user, which is converted to black and white, as mentioned in section 5.4. The second
Chapter – 5 Image Processing Techniques used in the Model

image is the sample image which is stored to compare each input image. Since the user uploads four images, there are four sample images stored in the folder. With each image of palm, appropriate sample image is matched to check whether the proper image is uploaded or not. The following code is implemented for template matching.

```csharp
// Comparing image with sample palm
// getting image for processing
System.Drawing.Bitmap osimg = new System.Drawing.Bitmap("D:\Ph.D Project\DSS\SampleImages\LHsample.jpg");
// converting into black and white
AForge.Imaging.Filters.FiltersSequence seq = new AForge.Imaging.Filters.FiltersSequence();
    seq.Add(new AForge.Imaging.Filters.OtsuThreshold());
Bitmap pimg = seq.Apply(oimg);
pimg.Save(folder + "\\" + lblfiles.Text + "pimg.jpg");
// create template matching algorithm's instance
// use zero similarity to make sure algorithm will provide anything
ExhaustiveTemplateMatching tm = new ExhaustiveTemplateMatching(0);
// compare two images
TemplateMatch[] matchings = tm.ProcessImage(pimg, osimg);
// check similarity level
if (matchings[0].Similarity <= 0.7)
{
    Response.Write("The Image is not a Palm Image!! Reload the correct Image");
}
```

In above code, “ExhaustiveTemplateMatching” class is used for template matching. The similarity value is between 0 and 1. If the similarity is 0, means two images are totally different. If the similarity value is 1, means two images are identical. In this model, if the similarity between two images is greater than 0.7, then only the image is considered as a palm image. Otherwise, the image is not considered by the prototype and the appropriate message is displayed to the user.

5.6 Summary

This chapter explains the image processing techniques like automatic rotation of horizontally scanned palm, resizing the image, converting a color image into black and white image, and comparison of two images using template matching for verification of uploaded palm images. These techniques are supportive techniques to the main model and have greater importance to run the prototype successfully.