بسمه تعالی

دانشگاه بیرجند

دانشکده مهندسی

گروه مهندسی کامپیوتر و IT

عنوان پروژه:

تشخیص اثر انگشت

تهیه کنندگان:

محمدرضا مشعل – *mrmashal@gmail.com*

رشید باقری – Rashid1368@yahoo.com

استاد راهنما:

دکتر جواد صدری

دیماه ۱۳۸۹

فهرست مطالب

پروژه تشخیص اثر انگشت

شکل ۱- نمونه اثر انگشت

۱. مقدمه

۱.۱ تعریف پروژه

در این پروژه، تعدادی اثر انگشت، به برنامه داده می‌شود. برنامه، ویژگی‌های این اثرانگشت ها را پیدا کرده و در یک ماتریس ذخیره می کند. سپس یک اثر انگشت جدید به برنامه داده می شود. برنامه باید مشخص کند که آیا این تصویر، جز اثرانگشت های موجود هست یا نه.

۱.۲ مشکلات پیش رو

ممکن است تصاویر اسکن شده از اثرانگشت ها، در اندازه های مختلف باشد. در این حالت، باید همه تصاویر به اندازه ثابت، نرمال سازی شوند.

ممکن است در تصاویر، نویزهایی وجود داشته باشد که باعث کم کردن کارایی برنامه شوند. برنامه باید، این نویزها را شناسایی کرده و از بین ببرد.

ممکن است اثر انگشتی، به صورت کج اسکن شده باشد و همان تصویر در ماتریس کل، ذخیره شده باشد. در این حالت برنامه باید این توانایی را داشته باشد که در این حالت هم قادر به تشخیص ویژگی ها باشد.

۱.۳ نرم افزار پیاده سازی

برای پیاده سازی و برنامه نویسی برنامه تشخیص اثر انگشت، از نرم افزار متلب نسخه ۲۰۱۰ استفاده شده است.

۱.٤ اصطلاحات

نقطه پایانی(Termination point): جایی که یک برآمدگی در اثرانگشت به آخر می‌رسد.(نقطه آبی مشخص شده در شکل ۲)

نقطه دوشاخه‌ای(Bifurcation point):نقطه ای که یک برآمدگی در اثرانگشت به دو برآمدگی دیگر تقسیم می-شود.(نقاط قرمز مشخص شده در شکل ۲)

شکل ۲- نقاط پایانی و دوشاخه‌ای در اثر انگشت

دوبعدی کردن تصویر(Binarization): پروسه تبدیل تصویر اصلی Gray Scale به یک تصویر سیاه و سفید.

شکل ۳- تبدیل تصویر Gray Scale به سیاه سفید

نازک سازی(Thinning): پروسه کاهش ضخامت هر کدام از برآمدگی‌ها به یک پیکسل.

شکل ٤- عمل نازک سازی تصویر

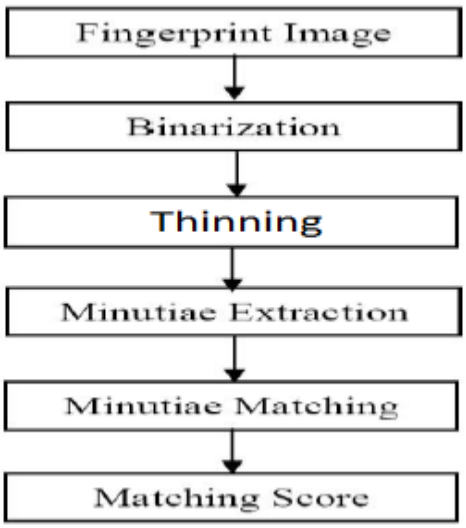
زاویه نقطه پایانی(**Termination angle**): زاویه بین خط افق و جهت برآمدگی.

زاویه نقطه دو شاخه ای(Bifurcation angle): زاویه بین خط افق و جهت حرکت شاخه بوجود آمده.

استخراج ویژگی(Minutiae extraction): استخراج ویژگی‌های منحصر به فرد هر اثرانگشت که شامل نقاط پایانی و دوشاخه ای و زاویه آنها باشد.

۲. نحوه انجام کار

۲.۱ طرح کلی



شکل ۵- بلوک دیاگرام سیستم تشخیص اثر انگشت

۲.۲ الگوریتم انجام کار

ورودی: تصویر Gray Scale از اثرانگشت

خروجی: امتیاز اثرانگشت با توجه به دیتابیس موجود از اثرانگشت‌های افراد مختلف

۱- تبدیل تصویر اثرانگشت به حالت باینری

۲- انجام عمل نازک سازی بر روی تصویر باینری

۳- استخراج ویژگی‌های منحصر به فرد اثرانگشت داده شده و ایجاد ماتریس ویژگی‌ها که شامل نقاط پایانی و دوشاخه ای و زاویه انها می باشد.

۴- مقایسه اثرانگشت ورودی با اثرانگشت‌های موجود در دیتابیس

۵- امتیاز اثرانگشت ورودی با توجه با ویژگی‌های ذکرشده آن محاسبه می شود. اگر این امتیاز از سطح آستانه تعریف شده بیشتر باشد،اثر انگشت داده شده با یکی از تصاویر دیتابیس همخوانی دارد.

۲.۲.۱ تبدیل تصویر Gray Scale به Black white

همانطور که در شکل ۳ مشخص است، تصویر ورودی، به صورت Gray Scale می باشد. برای راحت تر شدن پردازش تصویر و بدست آوردن ویژگی های اثر انگشت مذکور، باید هر تصویر ورودی را به تصویر باینری(سیاه و سفید) تبدیل کرد.

در زبان برنامه نویسی متلب، برای تبدیل تصویر به حالت دوبعدی، از دستور im2bw استفاده می شود.

$$original\_im = im2bw(original\_im);$$

۲.۲.۲ انجام عمل نازک سازی برروی تصویر

همانطور که قبلا نیز گفته شد،نازک سازی باعث تغییر ضخامت برآمدگی ها، به یک پیکسل می‌شود.نازک سازی تصویر،مکان نقاط پایانی و دوشاخه‌ای و همچنین زاویه آنها را تغییر نمی‌دهد.مهم ترین فایده نازک سازی در تشخیص اثر انگشت، کاهش پروسه زمانی استخراج ویژگی‌ها(Feature Extraction) می‌شود. در این مورد، در ادامه توضیح داده خواهد شد.

برای انجام نازک سازی در متلب، از دستور زیر استفاده می‌شود.

$$im = bwmorph(im, 'thin', Inf);$$

آرگومان سوم دستور Bwmorph که با Inf مشخص شده است، به این معنی است که عمل نازک سازی به صورت نامحدود انجام می‌شود.

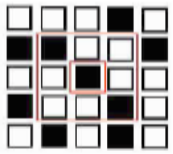
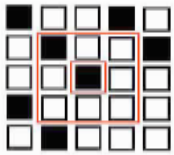
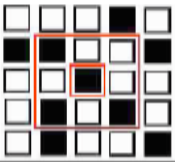
۲.۲.۳ استخراج ویژگی‌های منحصربه فرد اثرانگشت

در این مرحله، محل نقاط پایانی(Termination) و دوشاخه‌ای(Bifurcation) و همچنین زاویه آنها بدست می آید.

ویژگی‌های بدست آمده در یک ماتریس به نام InputMatrix ذخیره می‌شود.

به منظور تشخیص نقاط پایانی و دوشاخه‌ای به صورت زیر عمل می شود.

یک ماتریس ۳*۳ در نظر می گیریم و برروی تصویر نازک سازی شده حرکت می دهیم. در هر مرحله، ممکن است یکی از ۳ حالت زیراتفاق بیفتد.

	Crossing Number =2. Normal ridge pixel.
	Crossing Number =1. Termination point.
	Crossing Number =3. Bifurcation point.

شکل ٦- حالت‌های مختلف قرارگیری ماتریس ۳*۳ برروی تصویر

برای بدست آوردن Cross Number از فرمول زیر استفاده می شود.

$$CN = 0.5\sum_{i=1}^{8}|P_i - P_{i+1}|, \qquad P_9 = P_1$$

P_4	P_3	P_2
P_5	P	P_1
P_6	P_7	P_8

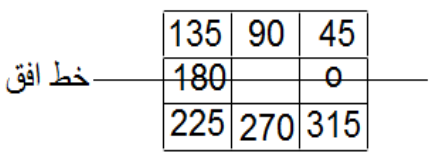
شکل ۷- فرمول بدست آوردن cross Number

نکته: باید توجه شود که اگر نقطه مرکزی ماتریس(نقطه P در ماتریس شکل۷) به رنگ سفید باشد، هیچ‌یک از حالت‌های پایانی و دوشاخه‌ای وجود نخواهد داشت.

همانطور که گفته شد، ویژگی‌های منحصربه فرد هر اثرانگشت، نقاط پایانی و دوشاخه ای و زاویه آنها خواهد بود.در این پروژه،برای بدست آوردن ویژگی‌های ذکرشده، تنها وضعیتی درنظرگرفته شده است که $CN = 1$ باشد، یعنی تنها نقاط پایانی درنظر گرفته شده است. به عبارت دیگر، نقاط دوشاخه ای با complement[1] نمودن تصویر اولیه، تبدیل به نقاط پایانی می‌شوند.

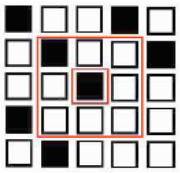
برای این کار ابتدا تصویر را نازک‌سازی می‌نمائیم و نقاط پایانی($CN = 1$) را پیدا می نمائیم. سپس این نقطه را به ماتریس ویژگی‌ها اضافه می نمائیم.

یکی دیگر از ویژگی‌هایی که باید برای این نقطه بدست آورد، زاویه آن با توجه به خط افق می‌باشد.



شکل ۸- بدست آوردن زاویه نقطه پایانی به کمک خط افق

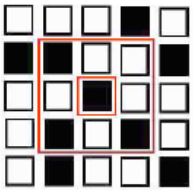
به عنوان مثال در شکل زیر، زاویه نقطه پایانی، ۱۳۵درجه می باشد.



شکل ۹- نمونه یک نقطه پایانی با زاویه ۱۳۵ درجه

حال، برای اثرانگشت داده شده، یک ویژگی استخراج شده است که شامل مکان نقطه پایانی و زاویه آن می باشد.همانطور که گفته شد ماتریس ۳*۳ را بر روی کل تصویر حرکت می دهیم و این عمل را انجام می دهیم.

به غیر از نقاط پایانی، باید نقاط دوشاخه ای و زاویه آنها را هم بدست آوریم.



شکل ۱۰- نمونه نقطه دوشاخه ای با زاویه ۲۷۰ درجه

برای این کار، در این پروژه، تصویر اثرانگشت را complement و سپس نازک سازی کرده و نقاط پایانی را بر روی آن جستجو می کنیم. توجه داشته باشید که تفاوت این قسمت با قسمت قبل که باز هم نقاط پایانی را جستجو می کردیم، در complement کردن تصویر می‌باشد. همانطور که در مرحله قبل ذکر شد،برای یافتن مشخصات نقاط پایانی، ابتدا تصویر را complement می‌نمائیم و سپس ویژگی‌های نقاط پایانی را استخراج می نمائیم، ولی در این مرحله تصویر نازک سازی شده اصلی را برای استخراج ویژگی‌ها استفاده می‌نمائیم. در اینصورت مراحل انجام کار مانند مرحله قبل خواهد بود. در نتیجه یکسری ویژگی جدید برای اثرانگشت برای نقاط دوشاخه‌ای بدست می‌آید.

۲.۲.٤ مقایسه اثرانگشت ورودی با دیتابیس

دیتابیس ما شامل ماتریس ویژگی‌های ۸۰ اثرانگشت می‌باشد که به شیوه گفته شده در مرحله ۳-۲-۲ بدست آمده است. در پروژه تهیه شده، نام این ماتریس، TemplateMatrix می‌باشد و نام ماتریس اثرانگشت ورودی، InputMatrix می باشد.

برای مقایسه این ماتریس ویژگی ورودی(InputMatrix) با هر یک از ماتریسهای ویژگی موجود در دیتابیس(TemplateMatrix) به شیوه زیر عمل می شود:

همانطور که گفته شد، ماتریس های ویژگی استخراج شده شامل مختصات نقاط ویژه به همراه زاویه و نوع آنها می باشد به این صورت که به ازای هر نقطه ویژه، یک سطر در ماتریس وجود دارد. بنابراین برای مقایسه دو اثر انگشت باید نقاط متناظر را در ماتریس های ویژگی شناسایی نماییم. برای این منظور با توجه به اینکه امکان دارد دو تصویر اثر انگشت داده شده، نسبت به هم دوران داشته باشند، نیاز به یک نقطه مرجع در هر یک از دو ماتریس خواهیم داشت که می دانیم با یکدیگر متناظر هستند تا با توجه به زاویه هر یک از این دو نقطه مرجع، سایر نقاط را در تصویر مربوطه دوران دهیم تا دوران نسبی دو تصویر (یا در واقع دو ماتریس ویژگی آنها) از بین برود. اما ما هیچگونه اطلاعی از دو نقطه ویژگی متناظر در دو تصویر برای انتخاب به عنوان نقاط مرجع نداریم. بنابراین به ازای هر حالت ممکن از انتخاب یک نقطه از ماتریس اول و یک نقطه از ماتریس دوم

(فقط نقاط هم نوع) به عنوان نقاط مرجع، عمل تطبیق را انجام می دهیم. اگر در حداقل یکی از حالات، نمره تطبیق به حداقل نمره قابل قبول رسید، دو اثر انگشت را یکسان اعلام می کنیم.

عمل تطبیق به این صورت انجام می شود که ابتدا کلیه نقاط هر ماتریس را با مرکز دوران نقطه مرجع آن ماتریس و با زاویه ای به اندازه منفی زاویه آن نقطه مرجع، دوران می دهیم. یعنی مختصات و زاویه جدید نقاط، از فرمول زیر محاسبه می شوند:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \cos\theta_{ref} & \sin\theta_{ref} & 0 \\ -\sin\theta_{ref} & \cos\theta_{ref} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - x_{ref} \\ y - y_{ref} \\ \theta - \theta_{ref} \end{bmatrix}$$

که در این فرمول، $\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix}$ مختصات و زاویه جدید نقاط، $\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$ مختصات و زاویه قبلی نقاط و $\begin{bmatrix} x_{ref} \\ y_{ref} \\ \theta_{ref} \end{bmatrix}$ مختصات و زاویه نقطه مرجع ماتریس مربوطه را نشان می دهند.

حال که دوران نسبی نقاط دو ماتریس از بین رفت، باید هر نقطه از ماتریس ویژگی اثر انگشت ورودی را با کلیه نقاط ماتریس ویژگی اثر انگشت دیتابیس از نظر مختصات و زاویه مقایسه کنیم. اگر تفاوت مقادیر به ازای یکی از حالات، از حد آستانه مورد نظر ما کمتر بود، نمره را یک واحد می افزاییم. پس از اینکه این کار را برای تمامی نقاط ماتریس ویژگی ورودی انجام دادیم، مقدار نمره به دست آمده را بر تعداد نقاط ویژگی تقسیم می کنیم تا نرمال (بین صفر و یک) شود. اگر تعداد نقاط ویژگی دو ماتریس (تعداد سطرهای آنها) با هم متفاوت بود، مقدار بزرگتر را برای مخرج کسر برمی گزینیم. به عبارت دیگر:

$$Score = \frac{number\ of\ matched\ points}{\max\{number\ of\ input\ features,\ \ number\ of\ template\ features\}}$$

۳. توضیح کد برنامه

کد MATLAB پروژه را در ادامه می‌آوریم (فایل m.fingerprint):

```
clc
warning off all
[FileName, PathName] = uigetfile('*.jpg', 'Select the Input Image');
im = imread([PathName FileName]);
inputFeatures = TotalFeatureExt(im);
load templates;
score_thresh = 0.9;
max_score = 0;
for i = 1:4
  fprintf('\n\n\n\n----------\n');
  fprintf('Library Item: %d\n', i);
  fprintf('----------\n');

  rows = (templateDatabase(:, 5) == i);
  score = CompareWithTemplate(inputFeatures, templateDatabase(rows, 1:4));
  max_score = max(double(score), double(max_score));
  if(score >= score_thresh)
    fprintf('\nINPUT MATCHED - Score: %f\n', max_score);
    disp(['Picture: library/' int2str(i) '.jpg']);
    showMinutiae(im, inputFeatures, ...
      ['MATCHED - Picture: library/' int2str(i) '.jpg']);
```

```
    return;
  end;
  fprintf('NOT MATCHED  -  Maximum Score: %f\n', max_score);
  max_score = 0;
end;
fprintf('\nINPUT NOT MATCHED');
showMinutiae(im, inputFeatures, 'NOT MATCHED');
```

دو خط اول که برای تنظیمات ابتدایی محیط MATLAB می‌باشند.

در سه خط بعد با نمایش پنجره Open Dialog فایل تصویر اثر انگشت ورودی را از کاربر دریافت نموده و پس از خواندن در متغیر im ذخیره می‌نماییم و سپس با استفاده از تابع TotalFeatureExt که بعدا توضیح داده خواهد شد ویژگیهای منحصر به فرد این اثر انگشت را طبق مطالب گذشته استخراج نموده و در ماتریس inputFeatures ذخیره می‌نماییم. چند تصویر نمونه قابل تطبیق در شاخه ACCEPT و چند تصویر نمونه غیر قابل تطبیق در شاخه REJECT قرار داده شده که می‌توانید از آنها به عنوان ورودی استفاده کنید.

در خط بعد ماتریس دیتابیس اثرهای انگشت خود را که در فایل mat.templates ذخیره شده است بارگزاری می‌نماییم. این ماتریس از الحاق ماتریسهای ویژگی اثرهای انگشت دیتابیس تشکیل شده که شامل همان چهار ستون گفته شده بعلاوه یک ستون پنجم که نام فایل اثر انگشت مربوطه را مشخص می‌کند می‌باشد.

در خط بعد متغیر score_thresh که با مقدار ۰/۹ مقداردهی شده است حدنصاب نمره لازم برای قبول اثر انگشت را مشخص می‌کند.

حلقه for بعدی برای جستجو روی تمامی اثرهای انگشت موجود در دیتابیس می‌باشد.

در این حلقه ابتدا سطور مربوط به اثر انگشت فعلی به منظور مقایسه را جدا نموده و سپس با استفاده از تابع CompareWithTemplate، آنرا با ماتریس ویژگی اثر انگشت ورودی (inputFeatures) مقایسه می‌نماییم و اگر نمره داده شده در خروجی تابع از حدنصاب بیشتر بود، اثر انگشت را می‌پذیریم.

در نهایت اگر هیچ یک از اثرهای انگشت دیتابیس با ورودی تطبیق داده نشد، آنرا رد می‌نماییم.

۳.۱ تابع TotalFeatureExt

```
function [ FeatureMat ] = TotalFeatureExt( original_im )
  original_im = im2bw(original_im);


  im = imcomplement(original_im);
  im = bwmorph(im, 'thin', Inf);
  FeatureMatrix = FeatureExt(im, 1);


  imc = original_im;
  imc = bwmorph(imc, 'thin', Inf);
  FeatureMatrix1 = FeatureExt(imc, 3);


  FeatureMat = [FeatureMatrix; FeatureMatrix1];
end
```

این تابع ویژگیهای منحصر به فرد اثر انگشت را استخراج می‌کند. همانطور که بیان شد پس از تبدیل تصویر Gray Scale به سیاه و سفید، باید یک بار روی خود تصویر و بار دیگر روی تصویر complement شده نقاط پایانی را بیابیم تا به ترتیب نقاط پایانی و دوشاخه‌ای بدست بیایند. هر بار ابتدا تصویر را با استفاده از bwmorph نازک سازی نموده و سپس آنرا به تابع FeatureExt که در زیر توضیح داده خواهد شد می‌دهیم تا نقاط پایانی را استخراج کند. در نهایت ماتریس های حاصله از دو مرحله فوق را با یکدیگر ادغام می‌کنیم.

دقت کنید که چون در تصاویر، نقاط سیاه مقدار صفر و نقاط سفید مقدار یک دارند، برای یافتن نقاط پایانی، از تصویر complement شده و برای پردازش یافتن نقاط دوشاخه ای از تصویر اصلی استفاده نموده‌ایم.

۳.۲ تابع FeatureExt

```
function [ FeatureMatrix ] = FeatureExt( im, value )
  [row, col] = size(im);
  FeatureMatrix = zeros(1, 4);
  count = 0;
  for i = 2: row − 1
    for j = 2: col − 1
      if(im(i, j) == 1)
        crossNumber = CrossNumber(im, i, j);


        if(crossNumber == 1) && (NonMarginalPoint(j, i, im) == true)
          count = count + 1;
          FeatureMatrix(count, 1) = j; % x coordinate
          FeatureMatrix(count, 2) = i; % y coordinate
          FeatureMatrix(count, 3) = ...
            orientationAngle(im, i, j, 1); % orientation angle
          FeatureMatrix(count, 4) = value; % type of minutiae
        end;
      end;
    end;
  end;
end
```

این تابع نقاط پایانی را در تصویر نازک سازی شده ورودی یافته و آنها را در قالب FeatureMatrix ذخیره می‌کند. در این ماتریس ستون اول و دوم طول و عرض نقطه، ستون سوم زاویه نقطه که قبلا تعریف شد و ستون چهارم که بیانگر نوع نقطه است به عنوان پارامتر ورودی دوم تابع گرفته شده و در همه سطرها همین مقدار قرار داده می‌شود.

پس از مقداردهی های اولیه، وارد دو حلقه می‌شویم که کل تصویر را درقالب مربع های ۳*۳ توضیح داده شده پیمایش می‌کند. در حلقه ابتدا بررسی می‌شود که نقطه وسط مربع مقدار ۱ داشته باشد. سپس مقدار Cross Number مربع با استفاده از تابع CrossNumber که در زیر خواهد آمد محاسبه می‌شود. سپس اگر این مقدار برابر یک بود و نقطه مربوطه یک نقطه حاشیه ای نبود (با بررسی توسط تابع NonMarginalPoint که در ادامه خواهد آمد) یک سطر با مقادیر لازم، به ماتریس خروجی افزوده می‌گردد. برای محاسبه زاویه از تابع orientationAngle استفاده شده که در ادامه می‌آید.

۳.۳ تابع CrossNumber

```
function [ crossNum ] = CrossNumber( im, i, j )
  crossNum = 0;
  a = zeros(1, 8);
  a(1) = im(i, j + 1);
  a(2) = im(i − 1, j + 1);
  a(3) = im(i − 1, j);
  a(4) = im(i − 1, j − 1);
  a(5) = im(i, j − 1);
  a(6) = im(i + 1, j − 1);
  a(7) = im(i + 1, j);
  a(8) = im(i + 1, j + 1);
  a(9) = a(1);
  for i = 1: 8
    crossNum = crossNum + abs(a(i) − a(i + 1));
  end;
  crossNum = crossNum/2;
end
```

این تابع مقدار Cross Number مربوط به مربع ۳*۳ به مرکز i و j را در تصویر im محاسبه می‌کند.

۳.٤ تابع NonMarginalPoint

```
function [ bool ] = NonMarginalPoint( x, y, im )
  [h, w] = size(im);
  %increase x − −>
  bool = false;
  i = x + 1;
  while(i <= w)
    if (im(y, i) == 1)
      bool = true;
      break;
    end;
    i = i + 1;
  end;
  if (bool == false)
    return;
  end;

  %decrease x < − −
  bool = false;
  i = x − 1;
  while(i >= 1)
    if (im(y, i) == 1)
      bool = true;
      break;
    end;
    i = i − 1;
  end;
  if (bool == false)
    return;
  end;

  %decrease y ^
```

```
  bool = false;
  i = y - 1;
  while(i >= 1)
    if (im(i,x) == 1)
      bool = true;
      break;
    end;
    i = i - 1;
  end;
  if (bool == false)
    return;
  end;

  %increase y (down)
  bool = false;
  i = y + 1;
  while(i <= h)
    if (im(i,x) == 1)
      bool = true;
      break;
    end;
    i = i + 1;
  end;
end
```

این تابع مشخص می‌کند که آیا نقاط بدست آمده به عنوان ویژگی، نقاط حاشیه‌ای تصویر اثر انگشت هستند که باید حذف شوند یا خیر. برای این منظور، از نقطه داده شده در تصویر، به سمت راست، چپ، بالا و پایین حرکت می‌کنیم. اگر در یکی از این جهات به انتهای تصویر رسیدیم بدون اینکه به نقطه‌ای با مقدار یک برخورد کنیم، نقطه حاشیه‌ای بوده است.

۳.۵ تابع orientationAngle

```
function [ Angle ] = orientationAngle( im, i, j, value )
  Angle = 0;
  if(im(i + 1, j) == value)
    Angle = 225;
  elseif(im(i + 1, j - 1) == value)
    Angle = 135;
  elseif(im(i, j - 1) == value)
    Angle = 180;
  elseif(im(i - 1, j - 1) == value)
    Angle = 135;
  elseif(im(i - 1, j) == value)
    Angle = 90;
  elseif(im(i - 1, j + 1) == value)
    Angle = 45;
  elseif(im(i, j + 1) == value)
    Angle = 0;
  elseif(im(i + 1, j + 1) == value)
    Angle = 315;
  end;
end
```

این تابع برای محاسبه زاویه نقطه ویژگی داده شده مطابق مباحث بیان شده استفاده می‌شود.

۳.۶ تابع CompareWithTemplate

```
function [ score ] = CompareWithTemplate( inputFeatures, templateFeatures )
  score = 0;
  ct = size(templateFeatures, 1);
  ci = size(inputFeatures, 1);
  for Ref_T = 1: ct
    for Ref_I = 1: ci
      if templateFeatures(Ref_T, 4)~ = inputFeatures(Ref_I, 4)
        continue;
      end;
      TTF = Transformation(templateFeatures, Ref_T);
      TIF = Transformation(inputFeatures, Ref_I);
      s = Score(TIF, TTF);
      score = max(double(s), double(score));
    end;
  end;
end
```

این تابع، دو ماتریس ویژگی داده شده را با یکدیگر مقایسه نموده و نمره مقایسه را طبق مباحث بیان شده محاسبه می‌کند. پس از مقداردهی های اولیه، دو حلقه for برای انتخاب نقاط مرجع ممکن گذاشته شده است. شرط ابتدایی به منظور اجتناب از انتخاب نقاط از نوع متفاوت به عنوان نقاط مرجع گذاشته شده است. پس از آن با تابع Transformation که در ادامه خواهد آمد، کلیه نقاط هر یک از دو ماتریس را نسبت به نقطه مرجع خودش تبدیل می‌کنیم. سپس تابع Score که در ادامه آنرا نیز توضیح خواهیم داد نقاط دو ماتریس تبدیل شده را با یکدیگر مقایسه کرده و نمره مربوطه را محاسبه می‌کند. خط آخر درون حلقه به منظور محاسبه ماکسیمم نمره ها به عنوان نمره تطبیق دو تصویر که خروجی تابع خواهد شد نوشته شده است.

۳.۷ تابع Transformation

```
function [ NewFeatureMat ] = Transformation( ...
  FeatureMat, RefIndex )


  xRef = FeatureMat(RefIndex, 1);
  yRef = FeatureMat(RefIndex, 2);
  thRef = FeatureMat(RefIndex, 3);


  NewFeatureMat = zeros(1, 4);
  R = [cosd(thRef) sind(thRef) 0; ...
    −sind(thRef) cosd(thRef) 0; ...
    0 0 1];

  [col, ~] = size(FeatureMat);
  for i = 1: col
    Temp = R ∗ [FeatureMat(i, 1) − xRef; ...
        FeatureMat(i, 2) − yRef; ...
        FeatureMat(i, 3) − thRef];

    NewFeatureMat(i, 1) = Temp(1);
```

```
    NewFeatureMat(i,2) = Temp(2);
    NewFeatureMat(i,3) = Temp(3);
    NewFeatureMat(i,4) = FeatureMat(i,4);
  end;
end
```

این تابع، عمل تبدیل نقاط ماتریس ویژگی داده شده را نسبت به نقطه مرجع موجود در سطر RefIndex با ماتریس تبدیل بیان شده انجام می‌دهد.

۳.۸ تابع Score

```
function [ score ] = Score( inputF, TemplateF )
  [colT, ~] = size(TemplateF);
  [colI, ~] = size(inputF);
  thresh = 30;
  th_thresh = 45;
  score = 0;

  for i = 1:colT
    for j = 1:colI
      if(abs(TemplateF(i,1) - inputF(j,1)) <= thresh) ...
       &&(abs(TemplateF(i,2) - inputF(j,2)) <= thresh) ...
       &&(abs(TemplateF(i,3) - inputF(j,3)) <= th_thresh)
        score = score + 1;
        break;
      end;
    end;
  end;
  score = score/max(colT, colI);
end
```

این تابع طبق فرمول بیان شده، هر نقطه ماتریس ویژگی اول را با نقاط ماتریس ویژگی دوم مقایسه نموده و تعداد تطبیق ها را تقسیم بر ماکسیمم تعداد نقاط در دو ماتریس می‌کند. محدوده قبول اختلاف بین مختصات نقاط با thresh و محدوده قبول اختلاف بین زوایا با th_thresh مشخص شده است.

۳.۹ تابع showMinutiae

```
function [] = showMinutiae( im, FeatureMat, msg )
  figure, imshow(im), title(msg);
  hold on;
  [col, ~] = size(FeatureMat);
  for i = 1:col
    if FeatureMat(i,4) == 1
      plot(FeatureMat(i,1), FeatureMat(i,2) ...
        , 'Marker', 'o', 'MarkerEdgecolor', 'r' ...
        , 'MarkerSize', 3);
    else
      plot(FeatureMat(i,1), FeatureMat(i,2) ...
        , 'Marker', 's', 'MarkerEdgecolor', 'b' ...
        , 'MarkerSize', 3);
```

```
    end;
  end;
  hold off;
end
```

این تابع را به منظور نمایش نقاط ویژگی یافته شده در ماتریس FeatureMat روی تصویر im به منظور استفاده در رفع اشکال برنامه نوشته‌ایم.

۳.۱۰ فایل m.createFeatures

```
templateDatabase = [];
for i = 1:4
  fileName = [int2str(i) '.jpg'];
  im = imread(['library/' fileName]);
  temporary  = TotalFeatureExt(im);
  temporary(:,5) = i;
  templateDatabase = [templateDatabase; temporary];
end;
save('templates.mat','templateDatabase');
```

این فایل به منظور ایجاد ماتریس دیتابیس اثر انگشت از روی تصاویر موجود در کتابخانه و ذخیره آن در فایل mat.templates نوشته شده است.

۴. تقدیر و تشکر

از دکتر جواد صدری که با راهنمایی‌هایشان ما را در تهیه این پروژه یاری فرمودند صمیمانه سپاسگزاریم.

۵. مراجع

- RAVI J, K B RAJA, VENUGOPAL K R, “Fingerprint Recognition Using Minutiae Score Matching”, *International Journal of Engineering Science and Technology* Vol 1)2(, 2009, 35-42
- Raymond Thai, *Fingerprint Image Enhancement and Minutiae Extraction*, 2003
- Rafael C .Gonzalez, *Digital Image Processing in MATLAB*, 2004
- Wuzhili, Fingerprint Recognition, Slides