DETECÇÃO DA BORDA DA PUPILA ATRAVÉS DE MORFOLOGIA MATEMÁTICA

Orlando Rollo de Carvalho[7], Wanilson Miranda de Figueiredo[8], Éldman de Oliveira Nunes[9]

Resumo. A segurança é uma preocupação constante no contexto atual da tecnologia da informação e uma das formas mais eficazes de mantê-la é a autenticação do usuário, e nesse contexto, a biometria apresenta-se como uma ótima alternativa. A biometria da íris apresenta uma técnica de identificação do indivíduo segura e confiável. O passo inicial para se reconhecer um individuo pela íris é o reconhecimento da borda da pupila, sendo este o objeto de estudo deste artigo. É teorizado, em linhas gerais, um método para a detecção automática das bordas da pupila em imagens digitais. São aplicadas algumas técnicas de processamento digital de imagens, algumas das quais utilizam métodos de morfologia matemática sobre imagens de teste, buscando avaliar a viabilidade de tal método automático de detecção.

Palavras-chave: íris, biometria, morfologia matemática, detecção de bordas, processamento digital de imagens.

Summary. Security is a constant worry in our current context of information technology and one of the most effective ways to keep it is the user's authentication and in this context biometry is a great alternative. Iris biometry presents a safe and reliable individual identification technique. The first step to recognize an individual by his iris is the recognization of the pupil rim, which is the subject of this article. It is elaborated, in general, as a method to detect automatically the rim of the pupil in digital images. Some digital image processing techniques, some of which use methods of mathematical morphology, are applied in images of test in order to assess the if it is viable such automatic method of detection.

Keywords: Íris, biometry, mathematical morfology, edge detection, digital image processing

1. Introdução

A segurança é uma preocupação constante em qualquer ramo de negócio e uma necessidade crescente, especialmente no aspecto de autenticação de usuários em áreas de acesso restrito, e a garantia desta autenticação deve ser feita da forma mais precisa possível.

O processo de autenticação se baseia em algo que o usuário possua, como cartões ou chaves (autenticação por propriedade), por algo que o usuário saiba, como uma senha (autenticação por conhecimento) ou por alguma característica comportamental ou fisiológica (autenticação por característica). A autenticação por propriedade possui a desvantagem de estar sujeita à utilização indevida por perda, furto, roubo ou clonagem. A autenticação por conhecimento possui a desvantagem de estar sujeita ao esquecimento ou descobrimento. A autenticação por característica, entretanto, possui a vantagem de se basear em características inerentes à pessoa e é um emergente campo de estudo conhecido pelo termo Biometria.

[7] Escola de Administração do Exército (EsAEx), Salvador, Brasil. orollo@bol.com.br.
[8] Escola de Administração do Exército (EsAEx), Salvador, Brasil. wanilson_m@bol.com.br.
[9] Escola de Administração do Exército (EsAEx), Salvador, Brasil. eldman@bol.com.br

As íris têm um grande potencial para autenticação de pessoas em sistemas de segurança, devido a sua extrema singularidade relativa a cada ser humano.

Este artigo propõe-se a tratar de apenas um dos passos necessários à identificação da íris: a detecção da borda da pupila

A abordagem aqui mostrada baseia-se na captura da imagem seguida de um pré–processamento (melhoria) da imagem e da detecção propriamente dita das bordas da pupila. Para esta melhoria e detecção foi elaborado, no ambiente de desenvolvimento Delphi, versão 5.0, um programa, apresentado com mais detalhes no item 4.2.

Para a apresentação deste método, este artigo está estruturado da seguinte forma: no item 2 é apresentado o conceito de biometria; no item 3 é apresentado com mais detalhes o processo de reconhecimento da íris e da pupila; no item 4 é descrito o método desenvolvido neste trabalho para a detecção das bordas da pupila; o item 5 comenta os resultados obtidos; o item 6 apresenta as conclusões sobre a abordagem; e o 7, as referências bibliográficas utilizadas neste artigo.

2. Biometria

A Biometria é uma técnica de reconhecimento de pessoas através das características físicas inerentes exclusivamente a cada ser humano. A biometria permite a identificação de pessoas através de padrões de voz, DNA, digitais, retina e íris humanas e é uma área que vem se desenvolvendo muito nos últimos anos.

O reconhecimento da íris é um problema bastante complexo que consiste em identificar a área da imagem de um olho onde está a íris (detecção) e, em seguida, reconhecer a qual indivíduo ela pertence, baseando-se em seus padrões característicos. A identificação de um indivíduo através da íris possui vantagens como: o processo de identificação da íris é muito mais seguro quando comparado ao processo de identificação da impressão digital; a complexidade da íris humana impossibilita a sua falsificação; o padrão da íris não muda com tempo, como ocorre com a voz.

As dificuldades referentes ao uso deste processo são: a impossibilidade de aplicá-lo em pessoas com doenças oftalmológicas graves, como catarata, cegueira parcial ou total, ou que não possuam o globo ocular; a captação da imagem ocular deve ser feita através de um método simples e não-invasivo, sem expor o usuário do sistema a desconforto; existem alguns processos já criados para o reconhecimento de íris, porém tais processos estão protegidos por patentes industriais.

3. Reconhecimento da íris

O processo completo de reconhecimento de íris é composto dos seguintes passos:

- Captura da imagem: a imagem é capturada através de um dispositivo apropriado e de forma confortável ao usuário, visando a obter uma imagem do olho humano com resolução e luminosidade adequadas;

Figura 1 - imagem de olho

- Pré-processamento (melhoria) da imagem:

- Transformação da imagem em tons de cinza;

Figura 2 - imagem em tons de cinza

- Contraste: Aumento das diferenças entre os tons de cinza;

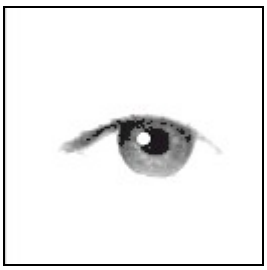


Figura 3 - Imagem com contraste aumentado

- Binarização: Conversão da imagem para somente preto e branco;

Figura 4 - imagem binarizada

– Detecção das bordas: a imagem é rastreada e, através de um método de detecção de bordas, tem suas linhas ressaltadas, de maneira que reste apenas o destaque das formas presentes na imagem;

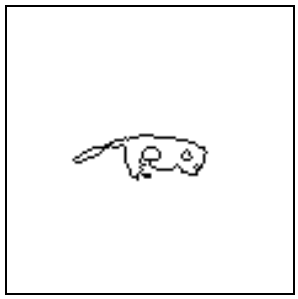


Figura 5 - Imagem com bordas destacadas

- Definição da área da íris: com as bordas da íris e pupila encontradas é possível definir a área da íris pela subtração da área da pupila da área formada pela circunferência externa da íris;

- Codificação da íris: a íris é codificada através da captura de algumas características suas, de modo que se possa armazená-la em um banco de dados (BD) integrado a um sistema de segurança;

- Identificação: Comparação da íris com outras armazenadas em um banco de dados.

4. Detecção da bordas da pupila

4.1 O Processo

O método utilizado aqui propõe-se a realizar a melhoria (transformação para tons de cinza, aumento de contraste, binarização) da imagem ocular capturada e a detecção de suas bordas.

Esta melhoria deverá ser automatizada através de um algoritmo que analise certas características da imagem, particularmente o seu histograma. O histograma de uma imagem é *um conjunto de números indicando o percentual de pixels naquela imagem que possuem um determinado nível de cinza. Um histograma é normalmente representado por um gráfico de barras que fornece para cada nível de cinza o número (ou percentual) de pixels correspondentes na imagem* (FILHO & NETO, 1999).

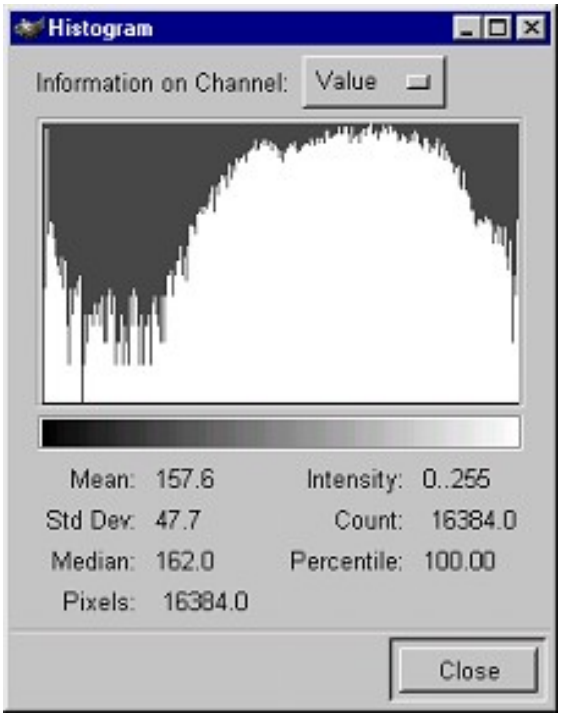


Figura 6 - Histograma de uma imagem gerado pelo software *Gimp*

De posse das informações do histograma, seria possível fazer os ajustes necessários dos parâmetros das sub-etapas do pré-processamento. Por exemplo, com base na análise do histograma, o programa determinaria a partir de qual tom de cinza os pixels presentes na imagem representam a área da pupila. Com esta informação, a imagem pode ser binarizada de forma a deixar presentes em tom preto somente os pixels que fazem parte da pupila, isolando-a do resto da imagem. Processo semelhante, também baseado em informações do histograma, poderia ser utilizado para se obter uma melhor detecção da área da íris.

Em alguns casos, porém, não é possível destacar os pixels da pupila com exatidão, pois a mesma pode apresentar uma variação um tanto elevada de tonalidade, de acordo com a qualidade da imagem. Esta variação pode ocasionar "buracos" na imagem binarizada. Com vistas a amenizar este problema, pode ser utilizada uma operação da morfologia matemática chamado *fechamento*. As operações na morfologia matemática, elaborada, a partir dos anos 60 por Georges Matheron e Jean Serra, procuram identificar as estruturas geométricas presentes em uma imagem. A morfologia matemática tem como um dos fundamentos a teoria dos conjuntos, e busca extrair de um conjunto desconhecido (a imagem a ser analisada) informações sobre geometria e topologia pela transformação deste conjunto (imagem) através de outro conjunto menor, totalmente conhecido, chamado elemento estruturante. De modo geral, este elemento contém características geométricas e/ou topológicas relacionadas com a informação que pretendemos extrair da imagem de interesse.

O fechamento é uma operação que permite remover pontos ruidosos presentes no interior de objetos em uma imagem binária. Conceitualmente, o fechamento é uma composição de duas outras operações de morfologia matemática, mais precisamente o fechamento é obtido pela *dilatação* da imagem pelo elemento estruturante seguida da *erosão* do resultado pelo mesmo elemento estruturante.

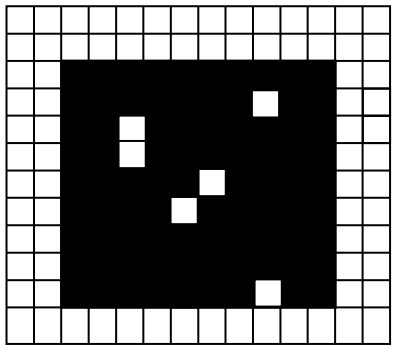


Figura 7 - Detalhe de imagem binarizada com ruídos

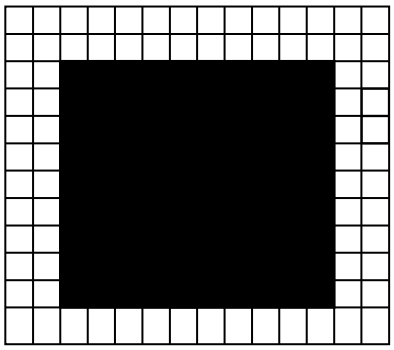


Figura 8 - Imagem após o fechamento

O processo de detecção de bordas propriamente dito utiliza-se de dois métodos da morfologia matemática, uma importante área de suporte do processamento digital. A primeira operação de morfologia matemática aplicada para que se obtenha as bordas na imagem binária é a erosão. Esta operação consiste em "erodir" uma imagem binária utilizando um elemento estruturante, que neste caso, tem dimensão 3 x 3 pixels e é em formato de cruz, com o ponto central no meio da estrutura, conforme mostra a figura 9, abaixo:

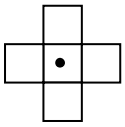


Figura 9 – elemento estruturante em cruz

Este elemento estruturante percorre toda a imagem e, para cada pixel, verifica se a cruz está totalmente inserida em uma área preenchida de pixels pretos. Caso afirmativo, este pixel é repetido em uma imagem temporária. Caso contrário, o pixel receberá a cor branca na imagem temporária. O resultado disto é uma imagem cujo volume das formas presentes é menor do que antes.

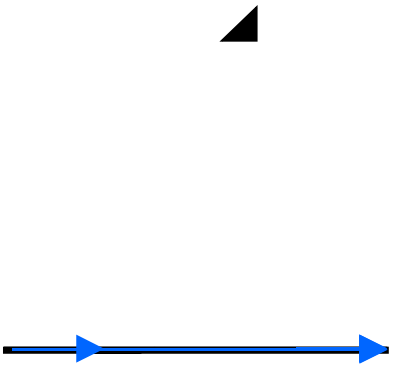


Figura 10 - Elemento estruturante passando pela imagem binária original

Figura 11 - Detalhe da imagem temporária erodida

A segunda operação a ser aplicada é a diferença. A imagem original é subtraída da imagem temporária erodida, resultando em um conjunto de pixels que ressaltam as diferenças entre as formas originais e as erodidas, desta forma desenhando as bordas presentes na imagem original.

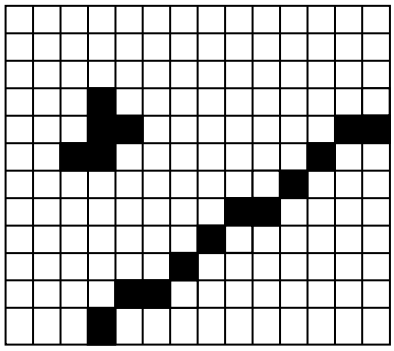


Figura 12 - Diferença entre as imagens, com as bordas destacadas.

A detecção de bordas através de morfologia matemática também pode ser feita através do seguinte processo: uma

dilatação da imagem seguida de uma operação de diferença entre a imagem dilatada e a original. Porém, desta maneira são obtidas bordas um pouco diferentes das bordas obtidas pelo método erosão+diferença.

4.2 As Ferramentas utilizadas

Para implementar o processo acima descrito e realizar os testes necessários, foi desenvolvido um programa em Delphi 5.0, o qual contém um menu chamado *Arquivo*, com as opções de carregar um arquivo do tipo Bitmap e de sair do programa, e um menu *Imagem*, com as operações de seccionar, contrastar, binarizar, detectar bordas e gerar o histograma de uma imagem.

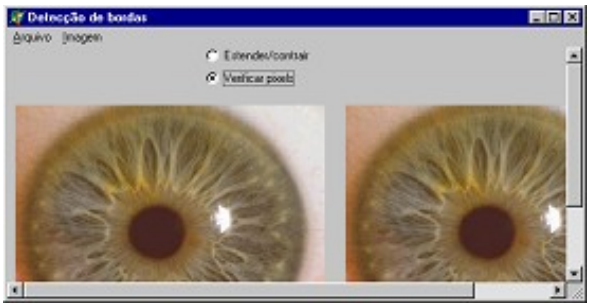


Figura 13 - Interface do programa

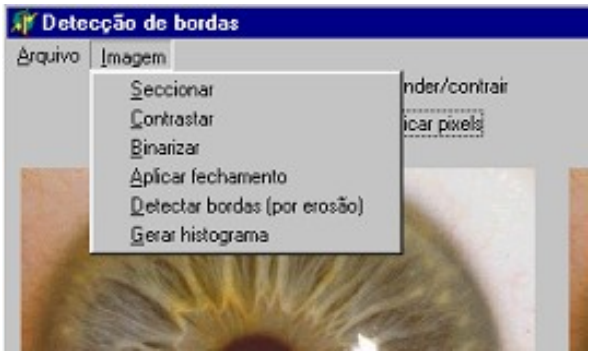


Figura 14 - Detalhe do menu *Imagem*

A opção *Seccionar* secciona a imagem de tal forma que uma área central da imagem seja escolhida para o processamento das operações que buscam localizar as bordas da pupila. Este procedimento foi implementado tendo em vista a utilização de imagens de olhos padronizadas, capturadas sempre sob as mesmas circunstâncias (posicionamento, luminosidade adequada, etc). Com estes requisitos satisfeitos, é possível selecionar uma área de grande probabilidade na imagem onde esteja localizada a íris e a pupila e, visando a uma melhora de velocidade na execução do programa, aplicar o processo de detecção apenas à área selecionada da imagem. Restringindo-se a área de processamento também evita-se o problema de lidar com elementos dificultadores para a detecção das bordas da pupila, como cílios, sombras e a própria íris, em alguns casos. Ao fim do seccionamento, apenas uma mensagem é emitida.

Figura 15 - Imagem com a área central destacada

A opção *Contrastar* permite aumentar as diferenças entre os tons de cinza presentes na imagem, fazendo com que os elementos presentes na imagem fiquem mais destacados em relação ao fundo. Baseado em um determinado valor de tom de cinza, este procedimento deixa mais claro os pixels que forem mais claros que este valor e mais escuros os pixels mais escuros. O objetivo disto é procurar realçar os elementos de interesse, como a íris e a pupila.

Figura 16 - Imagem com o contraste aumentado

A opção *Binarizar* transforma uma imagem em escala de cinza em uma imagem binária, com somente pixels brancos e pretos. Em particular no programa implementado, este procedimento calcula a média dos tons de cinza presentes na imagem e realiza a binarização baseado nesta média: pixels com valores acima dela tornam-se brancos e pixels com valores abaixo dela tornam-se pretos.

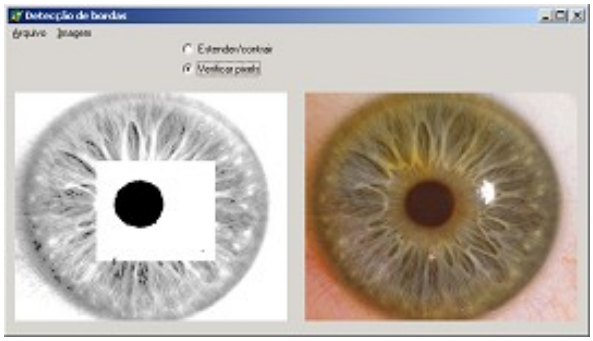


Figura 17 - Imagem binarizada

A opção *Aplicar Fechamento* realiza a operação de fechamento sobre a imagem 5 vezes seguidas, visando a eliminar alguns ruídos que possam atrapalhar na definição da borda da pupila.

A opção *Detectar bordas (por erosão)* destaca as bordas presentes na imagem, através dos processos de erosão e subtração, conforme explicado no item 4.1.

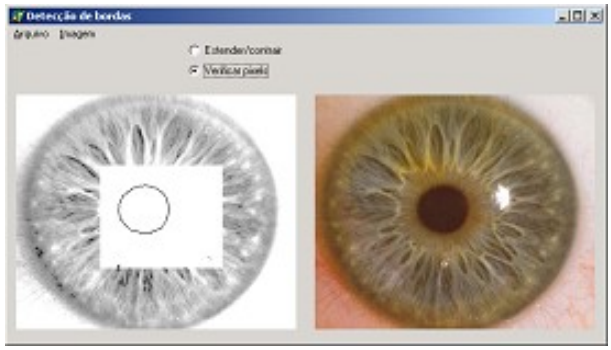


Figura 18 - Bordas destacadas

O programa possui ainda uma opção *Gerar histograma*, a qual gera o histograma da imagem, para que possa ser analisado dentro de outras operações.

5. Resultados

Para a realização dos testes do programa, foram utilizadas algumas imagens de íris e olhos obtidas na Internet. Em virtude da grande diferença de qualidade e resolução entre as imagens utilizadas, não foi possível especificar um método baseado na análise do histograma da imagem para automatizar todo o processo de detecção de bordas. Segue-se uma seqüência de três imagens de olhos que tiveram as bordas das pupilas detectadas através da operação manual do programa desenvolvido.

Figura 19 - Imagem 1: Secção binarizada

Figura 20 - Imagem 1: Bordas detectadas

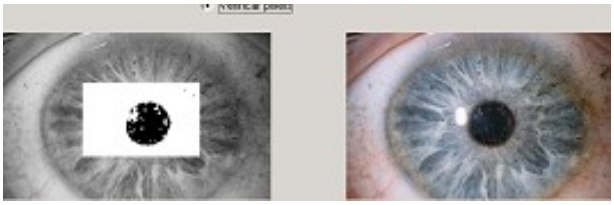


Figura 21 - Imagem 2: Binarização com problemas

Figura 22 - Imagem 2: Melhoria através da operação de fechamento

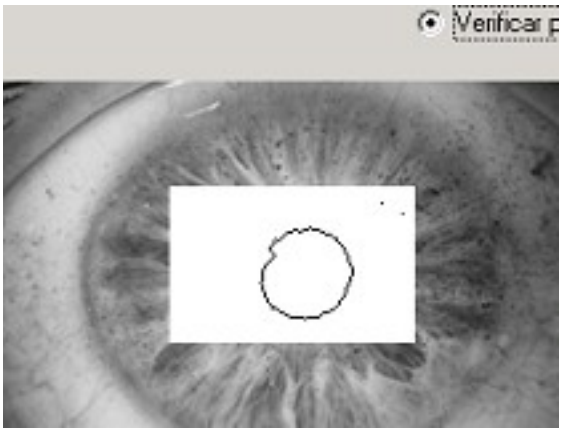


Figura 23 - Imagem 2: Resultado final desta imagem

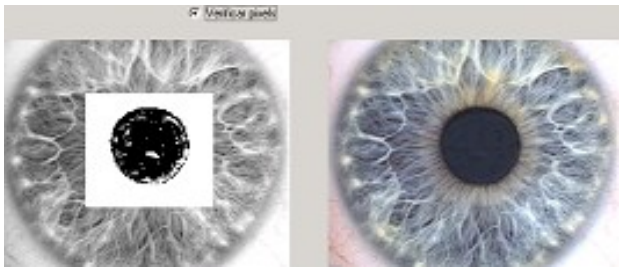


Figura 24 - Imagem 3: Binarização

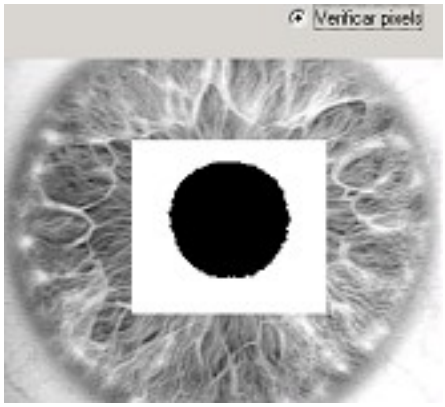


Figura 25 - Imagem 3: Melhoria através de morfologia matemática (fechamento)

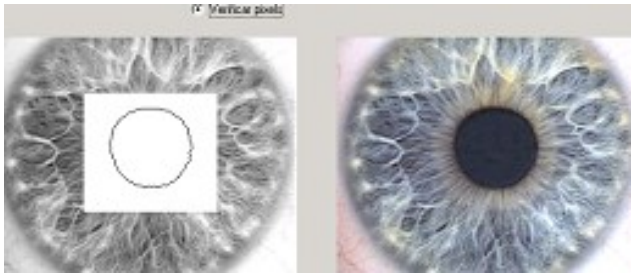


Figura 26 - Imagem 3: Resultado da detecção de bordas

Em dois dos exemplos mostrados, o processo de binarização não obteve bons resultados, em virtude de problemas nas imagens ocasionados pela iluminação inadequada na imagem 2, e pela baixa resolução da imagem 3. Após a aplicação do método de fechamento, houve uma melhoria das formas binárias presentes nas imagens. Nesses casos, conseguiu-se chegar a resultados razoável e bom, respectivamente às imagens 2 e 3, com relação à borda da pupila.

6. Conclusões

O sucesso da detecção da borda da pupila depende muito da qualidade da imagem utilizada, e por conseqüência, todo o processo de detecção da íris será muito influenciado pelos fatores qualidade e padronização das imagens. Para a eventual continuidade deste trabalho, visando chegar ao objetivo de detecção e reconhecimento da íris humana, muitas tarefas precisam ser implementadas, tais como: inicialmente, definir um padrão a ser adotado quanto ao tamanho, qualidade, resolução, luminosidade e posicionamento das imagens utilizadas durante o processo; localizar o centro da pupila através de métodos como a Transformada de Hough Randômica ou através de segmentos de reta paralelos e perpendiculares que tangenciem as bordas da pupila, conforme utilizado em (ARANA et al) - vide figura 27; e a detecção das bordas da íris tendo como ponto de partida o centro da pupila, já que são dois objetos concêntricos.

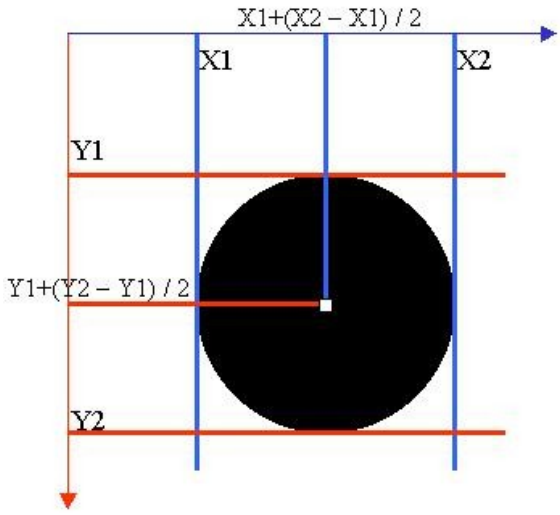


Figura 27 – Sugestão para localização do centro da pupila

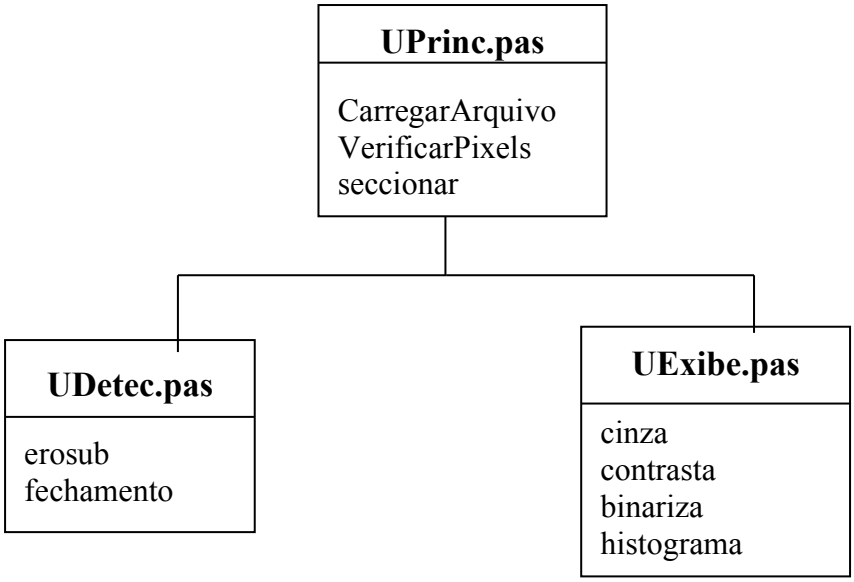
Referências

FILHO, O.M.; NETO, H.V. **Processamento Digital de Imagens**. Rio de Janeiro. Brasport, 1999.

ARANA, G.; CALDERÓN, M.; LARRE, N. C. **Caracterización del Patrón del Iris para Verificación de Identidad**. Bogotá. Departamento de Ingeniería Eletrônica - Pontificia Universidad Javeriana, Colombia, 2001.

APÊNDICE A

Diagrama Hierárquico das unidades do programa.



CarregarArquivo: lê um arquivo de Bitmap através da caixa de diálogo padrão de abrir imagem.

VerificarPixels: exibe através de uma mensagem na tela os 3 valores dos canais RGB de um determinado pixel que é clicado na imagem.

seccionar: realiza a operação de determinar uma área central da imagem que delimitará onde serão aplicados os processos de manipulação da imagem.

erosub: procedimento de detecção de bordas em imagens binárias através da operações de morfologia matemática erosão e diferença.

fechamento: procedimento que implementa a operação de fechamento, utilizando como elemento estruturante uma cruz de 3 x 3 pixels.

cinza: transforma uma imagem colorida para uma imagem em escala de cinza.

contrasta: procedimento que executa a aumento de contraste em uma imagem em escala de cinza.

binariza: transforma uma imagem em tons de cinza para somente preto e branco. Recebe como parâmetro o valor de tom de cinza que é utilizado como limiar.

histograma: lê a imagem em tons de cinza e calcula seu histograma.

APÊNDICE B

Código fonte do programa de detecção das bordas da pupila.

Arquivo DetIris.dpr

```
program DetIris;

uses
  Forms,
  UPrinc in 'UPrinc.pas' {Form1},
  Uexibe in 'Uexibe.pas',
  UdtEqul in 'UdtEqul.pas' {Form2},
  UDetec in 'UDetec.pas';

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TForm2, Form2);
  Application.Run;
end.
```

Arquivo Uprinc.pas

```
unit UPrinc;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtDlgs, ExtCtrls, StdCtrls, ComCtrls, Uexibe, Menus, Udetec, jpeg;

type
  TForm1 = class(TForm)
    quadro: TImage;
    OpImgDlg: TOpenPictureDialog;
    quadrorig: TImage;
    mnmenu: TMainMenu;
    mnOpImg: TMenuItem;
    Imagem1: TMenuItem;
    mnSeccio: TMenuItem;
    mnContras: TMenuItem;
    mnBinariz: TMenuItem;
    mnSair: TMenuItem;
    Arquivo1: TMenuItem;
    mnDetBrdEro: TMenuItem;
    rbverifpix: TRadioButton;
    RadioButton1: TRadioButton;
    Gerarhistograma1: TMenuItem;
    mnfechamen: TMenuItem;
    procedure quadroClick(Sender: TObject);
    procedure seccionar;
    procedure mnOpImgClick(Sender: TObject);
    procedure mnSeccioClick(Sender: TObject);
    procedure mnContrasClick(Sender: TObject);
    procedure mnBinarizClick(Sender: TObject);
    procedure mnSairClick(Sender: TObject);
```

```
    procedure btGrHistoClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure quadrorigClick(Sender: TObject);
    procedure mnDetBrdEroClick(Sender: TObject);
    procedure Gerarhistograma1Click(Sender: TObject);
    procedure mnfechamenClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  //Minhas variáveis
  expand,imgloaded,seccionada: boolean;
  area: array[1..2] of TPoint;

implementation

uses UdtEqul;

{$R *.DFM}

procedure TForm1.quadroClick(Sender: TObject);
var tamquadro:TRect; clic:TPoint; r,g,b:byte;
begin
if rbVerifPix.Checked then
  begin
  clic:=quadro.ScreenToClient(mouse.cursorpos);
  with quadro.canvas do
      begin
      r:=GetRValue(Pixels[clic.x,clic.y]);
      g:=GetGValue(Pixels[clic.x,clic.y]);
      b:=GetBValue(Pixels[clic.x,clic.y]);
      end;
  showmessage('R:'+inttostr(r)+'   G:'+inttostr(g)+'   B:'+inttostr(b));
  end
else //É para expandir ou contrair a imagem
  begin
  expand:=not(expand);
  if expand then
     quadro.Stretch:=true
  else
     begin
     quadro.Stretch:=false;
     quadro.Canvas.Draw(0, 0, quadro.Picture.Bitmap)
     end
  end
end;

procedure TForm1.seccionar;
var i,j: byte; imga,imgl: word;
begin
imgl:=quadro.Picture.Width;
imga:=quadro.Picture.Height;

{Calcula o ponto superior esquerdo da área de trabalho utilizando
os seguintes parâmetros: a área é aproximadamente 1/3 da largura
e altura totais, acrescidos de 10% em cada lado}
```

```
area[1].X:=round((imgl/3)*0.9);
area[1].Y:=round((imga/3)*0.9);
area[2].X:=round((imgl*2/3)*1.1);
area[2].Y:=round((imga*2/3)*1.1);
seccionada:=true;
end; //procedure seccionar

procedure TForm1.mnOpImgClick(Sender: TObject);
begin
if OpImgDlg.execute then
  begin
  quadro.Picture.LoadFromFile(OpImgDlg.FileName);
  quadrorig.Picture.LoadFromFile(OpImgDlg.FileName);
  imgloaded:=true;
  seccionar
  end
end;

procedure TForm1.mnSeccioClick(Sender: TObject);
begin
seccionar;
showmessage('Cálculo do seccionamento efetuado!');
end;

procedure TForm1.mnContrasClick(Sender: TObject);
begin
cinza(quadro);
form2.Showmodal;
end;

procedure TForm1.mnBinarizClick(Sender: TObject);
begin
binariza(Form1.quadro,area,seccionada)
end;

procedure TForm1.mnSairClick(Sender: TObject);
begin
Form1.close
end;

procedure TForm1.btGrHistoClick(Sender: TObject);
begin
histograma(quadro);
limiarpup(quadro)
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
quadrorig.Picture:=quadro.Picture;
seccionada:=false;
end;

procedure TForm1.quadrorigClick(Sender: TObject);
var tamquadro:TRect; clic:TPoint; r,g,b:byte;
begin
if rbVerifPix.Checked then
 begin
 clic:=quadrorig.ScreenToClient(mouse.cursorpos);
 with quadrorig.canvas do
    begin
```

```
     r:=GetRValue(Pixels[clic.x,clic.y]);
     g:=GetGValue(Pixels[clic.x,clic.y]);
     b:=GetBValue(Pixels[clic.x,clic.y]);
     end;
  showmessage('R:'+inttostr(r)+'   G:'+inttostr(g)+'   B:'+inttostr(b));
  end
else //É para expandir ou contrair a imagem
  begin
  expand:=not(expand);
  if expand then   quadrorig.Stretch:=true
  else
    begin
    quadrorig.Stretch:=false;
    quadrorig.Canvas.Draw(0, 0, quadrorig.Picture.Bitmap)
    end
  end
end;

procedure TForm1.mnDetBrdEroClick(Sender: TObject);
begin
erosub(quadro,area,seccionada)
end;

procedure TForm1.Gerarhistograma1Click(Sender: TObject);
begin
histograma(quadro);
limiarpup(quadro)
end;

procedure TForm1.mnfechamenClick(Sender: TObject);
begin
fechamento(quadro,area,5,seccionada);
end;

end.
```

Arquivo UDetec.pas

```
unit UDetec;

interface

Uses  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtDlgs, ExtCtrls, StdCtrls, ComCtrls;

procedure erosub(quadro:TImage; p:array of TPoint; seccionada:boolean);
procedure fechamento(quadro:TImage; p:array of Tpoint; iteracoes:byte; seccionada:boolean);

implementation

function cor(pixel:longint):byte;
begin
   cor:=pixel and 255;
end;

procedure erosub(quadro:TImage; p:array of TPoint; seccionada:boolean);
var i,j{,imgl,imga}:word; temp:byte; qd2:TPicture;
begin
```

```
qd2:=TPicture.Create;
qd2.Bitmap:=quadro.picture.bitmap;
{Este procedimento realiza a detecção de bordas APENAS EM IMAGENS
BINARIZADAS, ou seja, imagens somente com pixels em preto ou
branco. O procedimento inicialmente faz a erosão da imagem original
utilizando como "elemento estruturante" a cruz. Esta parte do
código é identificada pela sequência de "AND"s abaixo.

Em seguida o procedimento faz a "subtração" da imagem original
da imagem erodida, deixando visível apenas as diferenças entre
elas, que devem ser as bordas da imagem}

for i:=p[0].x+1 to p[1].x -1 do
  for j:=p[0].y to p[1].y -1 do
    with quadro.Canvas do
    begin
    if (cor(Pixels[i-1,j])=0) and (cor(Pixels[i,j])=0) and
      (cor(Pixels[i+1,j])=0) and (cor(Pixels[i,j-1])=0) and
      (cor(Pixels[i,j+1])=0) then
      qd2.bitmap.Canvas.pixels[i,j]:=RGB(0,0,0)
    else
      qd2.bitmap.Canvas.pixels[i,j]:=RGB(255,255,255);
    temp:=255 - cor(qd2.bitmap.Canvas.pixels[i,j]) + cor(Pixels[i,j]);
    qd2.bitmap.Canvas.pixels[i,j]:=RGB(temp,temp,temp)
    end;
quadro.picture.Bitmap:=qd2.Bitmap
end; //procedure erosub

procedure fechamento(quadro:TImage; p:array of Tpoint; iteracoes:byte; seccionada:boolean);
var i,j,k,imgl,imga:word; qd2:TPicture;
begin
if not seccionada then
  begin
  showmessage('A imagem ainda não foi seccionada. Operação cancelada.');
  exit
  end;
imgl:=quadro.picture.Width;
imga:=quadro.picture.Height;
qd2:=TPicture.Create;
qd2.Bitmap:=quadro.picture.bitmap;
//Realiza a dilatação da imagem
for k:=1 to iteracoes do
begin
for i:=p[0].x to p[1].x do
  for j:=p[0].y to p[1].y do
    with quadro.Canvas do
    begin
    if (cor(Pixels[i-1,j])=0) or (cor(Pixels[i,j])=0) or
      (cor(Pixels[i+1,j])=0) or (cor(Pixels[i,j-1])=0) or
      (cor(Pixels[i,j+1])=0) then
      qd2.bitmap.canvas.pixels[i,j]:=RGB(0,0,0)
    else
      qd2.bitmap.canvas.pixels[i,j]:=RGB(255,255,255);
    end; //with
quadro.picture.Bitmap:=qd2.Bitmap
end; //for k
//E então a erosão
for k:=1 to iteracoes do
begin
for i:=p[0].x to p[1].x do
```

```
  for j:=p[0].y to p[1].y do
    with quadro.canvas do
    begin
    if (cor(Pixels[i-1,j])=0) and (cor(Pixels[i,j])=0) and
      (cor(Pixels[i+1,j])=0) and (cor(Pixels[i,j-1])=0) and
      (cor(Pixels[i,j+1])=0) then
      qd2.bitmap.canvas.pixels[i,j]:=RGB(0,0,0)
    else
      qd2.bitmap.canvas.pixels[i,j]:=RGB(255,255,255);
    end; //with
quadro.picture.Bitmap:=qd2.Bitmap;
end; //for k
showmessage('Fim')
end; //procedure fechamento

end.
```

Arquivo UExibe.pas

```
unit Uexibe;

interface

Uses Graphics, SysUtils, Dialogs, extctrls, Windows;

procedure cinza(quadro: TImage);
procedure contrasta(quadro: TImage; percDark,percBrig,perCorte:real);
procedure binariza(quadro: TImage; p:array of TPoint; seccionada:boolean);
procedure histograma(quadro: TImage);
procedure limiarpup(quadro: TImage);

implementation

var histo:array[0..255] of real;
type pico=record
  cor:byte;
  qtd:real
  end;

procedure cinza(quadro: TImage);
var corcnz: byte; i,j:word;
begin
with quadro.Canvas do
  begin
  for i:=0 to quadro.Picture.Width -1 do
    for j:=0 to quadro.Picture.Height -1 do
      begin
      //Calcula o valor do cinza:   cinza = 0.3R + 0.59G + 0.11B
      corcnz:=trunc(0.3*GetRValue(Pixels[i,j])+0.59*GetGValue(Pixels[i,j])+0.11*GetBValue(Pixels[i,j]
));
       {Coloca o tom de cinza em todos os canais e passa como a nova
        cor para o pixel}
       Pixels[i,j]:=RGB(corcnz,corcnz,corcnz)
       end
  end;
end; //procedure cinza;

procedure contrasta(quadro: TImage; percDark,percBrig,perCorte:real);
```

```
{Este procedimento acha o maior e o menor valor de pixel da imagem e faz
um "corte" entre os pixels, separando para serem escurecidos, usando como
valores padrões, em 50% os pixels de valores 8% mais baixos dentro da
faixa de valores da imagem, e clareando em 70% os pixels de valores 92%
mais altos}
var faixa,menor,maior,cor,pcorte,novacor:byte; i,j:word; bydir:longint;
begin
bydir:=255;
menor:=quadro.Canvas.Pixels[0,0] and bydir;  maior:=menor;
for i:=0 to quadro.Picture.Width -1 do
  for j:=0 to quadro.Picture.Height -1 do
    begin
    {Realiza um E binário para guardar na variável "cor" somente o valor
     do byte mais à direita do valor de Pixels[i,j]}
    cor:=quadro.Canvas.Pixels[i,j] and bydir;
    if cor<menor then
      menor:=cor;
    if cor>maior then
      maior:=cor
    end;
faixa:=maior-menor;
pcorte:=round(faixa*perCorte)+menor;
showmessage('Valor do corte: '+inttostr(pcorte));
with quadro.Canvas do
  for i:=0 to quadro.Picture.Width -1 do
    for j:=0 to quadro.Picture.Height -1 do
      begin
      cor:=Pixels[i,j] and bydir;
      if cor<=pcorte then
        novacor:=round(percDark*GetRValue(Pixels[i,j]))
      else
        if round(percBrig*GetRValue(Pixels[i,j]))<=255 then
          novacor:=round(percBrig*GetRValue(Pixels[i,j]))
        else
          novacor:=255;
      Pixels[i,j]:=RGB(novacor,novacor,novacor)
      end
end; //procedure contrasta;

procedure binariza(quadro:TImage; p:array of TPoint; seccionada:boolean);
var i,j:word; cont:longword; bydir:longint; cor,pcorte:byte;
   sum,tomedio:real;
begin
if not seccionada then
  begin
  showmessage('A imagem ainda não foi seccionada. Operação cancelada.');
  exit
  end;
bydir:=255; cont:=0; sum:=0;
//Calcula o valor médio de tom dos pixels
for i:=p[0].x to p[1].x do
  for j:=p[0].y to p[1].y do
    begin
    cor:=quadro.Canvas.Pixels[i,j] and bydir;
    if cor<>255 then
      begin
      inc(cont);
      sum:=sum+cor
      end
    end;
```

```
tomedio:=sum/cont;
pcorte:=round(strtofloat(inputbox('Corte para binarizar imagem',
'O valor médio de tom de cinza atual da imagem é '+format('%.1f',[tomedio])+
' . Forneça um valor de cinza para binarizar a imagem: ',
format('%.1f',[tomedio])));
//Binariza a imagem
for i:=p[0].x to p[1].x do
  for j:=p[0].y to p[1].y do
    with quadro.Canvas do
      begin
      cor:=Pixels[i,j] and bydir;
      if cor<pcorte then
        Pixels[i,j]:=RGB(0,0,0)
      else
        Pixels[i,j]:=RGB(255,255,255)
      end
end; //procedure binariza

procedure limiarpup(quadro: TImage);
const n=6; {n é o número de picos que serão armazenados para que
seja feita uma verificação de qual é o maior pico em tons escuros}
var picos:array[1..n] of pico; i,j,maior,cont,cor:byte;
   maiorqtd:real; bydir:longint;
begin
cont:=0;
for i:=1 to n do
  begin
  picos[i].cor:=0;
  picos[i].qtd:=0
  end;
if (histo[0]>histo[1]) then
  begin
  inc(cont);
  picos[cont].cor:=0;
  picos[cont].qtd:=histo[0];
  end;
for i:=1 to 254 do
  if (histo[i-1]<histo[i]) AND (histo[i]>histo[i+1]) then
    begin
    inc(cont);
    picos[cont].cor:=i;
    picos[cont].qtd:=histo[i];
    if cont>=n then break;
    end;
maiorqtd:=picos[1].qtd;
maior:=1;
for i:=2 to n do
  begin
  if picos[i].qtd>maiorqtd then
    begin
    maiorqtd:=picos[i].qtd;
    maior:=i;
    end
  end;
showmessage('Pico -> '+inttostr(picos[maior].cor));
picos[maior].cor:=picos[maior].cor+12;
showmessage('Pico aumentado-> '+inttostr(picos[maior].cor));
bydir:=255;
for i:=0 to quadro.Picture.Width -1 do
  for j:=0 to quadro.Picture.Height -1 do
```

```
      with quadro.Canvas do
        begin
        cor:=Pixels[i,j] and bydir;
        if cor<=picos[maior].cor then
          Pixels[i,j]:=RGB(0,0,0)
        else
          Pixels[i,j]:=RGB(255,255,255)
        end
end; //procedure limiarpup;

procedure histograma(quadro: TImage);
var numpix:longword; i,j,imgl,imga:word; cor:byte;
    bydir:longint; contpix:array[0..255] of longword;
begin
for i:=0 to 255 do
   contpix[i]:=0;

bydir:=255;
imgl:=quadro.Picture.Width;
imga:=quadro.Picture.Height;
numpix:=imgl*imga;

for i:=0 to imgl -1 do
   for j:=0 to imga -1 do
      begin
      cor:=quadro.Canvas.Pixels[i,j] and bydir;
      inc(contpix[cor]);
      end;
for i:=0 to 255 do
   histo[i]:=contpix[i]/numpix;
showmessage('Histograma gerado.');
end; //procedure histograma;

end.
```

Arquivo UdtEqul.pas

```
unit UdtEqul;

{Esta Unit implementa a interface necessária para permitir obter do usuário os parâmetros para o processo
de aumento de contraste de uma imagem em escala de cinza}

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ComCtrls, StdCtrls, UPrinc, Uexibe;

type
  TForm2 = class(TForm)
    Label1: TLabel;
    UpDown1: TUpDown;
    edNumEq: TEdit;
    trbPcorte: TTrackBar;
    Label2: TLabel;
    UpDown2: TUpDown;
    edPerDark: TEdit;
```

```
    Label3: TLabel;
    UpDown3: TUpDown;
    edPerBrig: TEdit;
    Label4: TLabel;
    bkOk: TButton;
    btCancel: TButton;
    lbperCorte: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure bkOkClick(Sender: TObject);
    procedure btCancelClick(Sender: TObject);
    procedure trbPcorteChange(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.DFM}

procedure TForm2.FormCreate(Sender: TObject);
begin
edNumEq.text:='1';
edPerDark.Text:='50';
edPerBrig.text:='70';
trbPcorte.position:=8;
end;

procedure TForm2.bkOkClick(Sender: TObject);
var i:byte; pdark,pbrig,percorte:real;
begin
percorte:=trbPCorte.position/100;
pdark:=strtoint(edPerDark.text)/100;
pbrig:=strtoint(edPerBrig.text)/100 + 1;
for i:=1 to strtoint(edNumEq.text) do
   contrasta(Form1.quadro,pdark,pbrig,percorte);
Form2.close
end;

procedure TForm2.btCancelClick(Sender: TObject);
begin
Form2.close
end;

procedure TForm2.trbPcorteChange(Sender: TObject);
begin
lbpercorte.Caption:=inttostr(trbPCorte.position)+' % da escala de cinza';
end;

end.
```