UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

TORTUGA: UM PROTÓTIPO PARA IDENTIFICAÇÃO DE CÁGADOS DA ESPÉCIE PHRYNOPS WILLIAMSI

GUILHERME OECKSLER BERTOLDI

BLUMENAU
2016

GUILHERME OECKSLER BERTOLDI

TORTUGA: UM PROTÓTIPO PARA IDENTIFICAÇÃO DE CÁGADOS DA ESPÉCIE PHRYNOPS WILLIAMSI

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. Aurélio Faustino Hoppe, Mestre - Orientador

BLUMENAU
2016

TORTUGA: UM PROTÓTIPO PARA IDENTIFICAÇÃO DE CÁGADOS DA ESPÉCIE PHRYNOPS WILLIAMSI

Por

GUILHERME OECKSLER BERTOLDI

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Prof. Aurélio Faustino Hoppe, Mestre – Orientador, FURB
Membro:	Prof. Dalton Solano dos Reis, Mestre – FURB
Membro:	Prof. Matheus Carvalho Viana, Dr. – FURB

Blumenau, 6 de julho de 2016

Dedico este trabalho à minha família, que prestou todo o apoio necessário para conclusão do mesmo.

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

Aos meus pais e meu irmão, que concederam todo o suporte e motivação necessária para a conclusão desta nova etapa da minha vida.

Aos meus amigos e professores, pelo auxílio e conhecimento a mim prestados.

Aos biólogos André Santos e José Carlos Rocha, por proverem as imagens dos cágados utilizadas no desenvolvimento do trabalho.

Ao meu orientador, Aurélio Faustino Hoppe, pela ajuda na escolha do tema e pelo auxílio prestado através do seu conhecimento durante todo o desenvolvimento do trabalho.

Quanto maior o conhecimento, menor o ego, quanto maior o ego, menor o conhecimento.

Albert Einstein

RESUMO

Este trabalho apresenta um protótipo para identificação de cágados da espécie *Phrynops williamsi* a partir de uma imagem capturada. A identificação é realizada por meio dos descritores de forma que representam computacionalmente a listra em formato de ferradura e da listra circular localizadas na parte inferior da cabeça do cágado. A imagem informada é transformada em escalas de cinza, binarizada, filtrada por meio de operações morfológicas, segmentada por meio da extração do contorno dos objetos e tem suas componentes selecionadas com base nas suas características geométricas. A partir da extração dessas características, o protótipo é capaz de identificar se o cágado de entrada é um dos cágados cadastrados na base de dados. Os resultados obtidos demonstram que o protótipo alcançou uma taxa de 85,71% de acerto nas comparações intra-classe e 85,17% nas comparações inter-classe.

Palavras-chave: Processamento de imagens. Descritores de Fourier. Segmentação. Classificação.

ABSTRACT

This work presents a prototype for the identification of *Phrynops williamsi* turtles from a captured image. This identification is performed using shape descriptors who computationally represent the horseshoe-shaped and circular-shaped bands that can be found on the ventral surface of the turtle's head. The input image is converted to gray-scale, binarized, filtered with morfologic operations, segmented based on the objects' contours and the components are selected based on their geometric characteristics. With the extraction of these characteristics, the prototype is then able to recognize if the turtle from the input image is one of the registered turtles at the database. Results show that the prototype has reached a success rate of 85.71% in intra-class comparisons and 85.17% in inter-class comparisons.

Keywords: Image processing. Fourier descriptors. Segmentation. Classification.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API – *Application Programming Interface*

DFT – *Discrete Fourier Transform*

EA – *Enterprise Architect*

EE – Elemento Estruturante

FD – *Fourier Descriptor*

FPD – *Farthest Point Distance*

IDFT – Inverse *Discrete Fourier Transform*

RF – Requisitos Funcional

RNF – Requisito Não Funcional

UML – *Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

Atualmente, estima-se que existam no planeta Terra milhões de diferentes tipos de organismos vivos compartilhando a biosfera e, que segundo Araújo e Bossolan (2006, p. 1), o reconhecimento destas espécies está diretamente relacionado à história do homem. Linnaeus produziu em sua obra chamada *Systema Naturae*, um extenso sistema de classificação de plantas e animais, no qual dividiu os reinos animal, vegetal e mineral em espécies e deu a cada uma delas um nome distinto (HICKMAN; ROBERTS; LARSON, 2001, p. 197).

Este sistema é utilizado atualmente e consiste na divisão dos reinos em espécies, no agrupamento destas espécies em gêneros, gêneros em ordens e ordens em classes (HICKMAN; ROBERTS; LARSON, 2001, p. 197). Ainda segundo os autores, esta divisão taxonômica foi consideravelmente expandida desde a proposta por Linnaeus, sendo que existem agora sete níveis de classificação mandatórios no reino animal: reino, filo, classe, ordem, família, gênero e espécie.

Segundo Hickman, Roberts e Larson (2001, p. 560), a classe réptil é composta por cerca de 7.000 espécies e ocupam uma grande variedade de *habitats* terrestres e aquáticos. Pough, Janis e Heiser (2008, p. 5) apontam que a ordem Testudines é composta por cerca de 300 espécies de tartarugas, que, provavelmente, são um dos répteis do grupo de vertebrados mais reconhecíveis devido à presença de carapaça.

Nos dias atuais, há muitos estudos de marcação e análise de comportamento dos animais (DODD, 2009, p. 123). Segundo McDiarmid et al. (2012, p. 146), os marcadores utilizados na identificação de tartarugas, por exemplo, são comumente criados por meio de marcações incisivas feitas no casco das mesmas, caracterizando-se em um método invasivo. Além disso, esse tipo de marcação causa *stress* aos cágados e, em eventuais circunstâncias, resulta na morte do animal devido aos ferimentos gerados. Contudo, Burghardt (2008, p. 5) comenta que biólogos preferem utilizar métodos não invasivos, visto que estes não causam ferimentos ao animal, são mais fáceis e possuem um custo inexpressivo quando utilizados na identificação de espécies de animais.

Diante deste cenário, este trabalho apresenta um protótipo capaz de identificar cágados da espécie *Phrynops williamsi* por meio de um método não-invasivo, utilizando-se de características extraídas da imagem do animal para gerar um identificador único.

1.1 OBJETIVOS

O objetivo deste trabalho é desenvolver um protótipo de Processamento de Imagens que faça a identificação de cágados da espécie *Phrynops williamsi* por meio de um método não-invasivo.

Os objetivos específicos do trabalho são:

a) efetuar a segmentação da listra em formato de ferradura localizada na parte inferior da cabeça do animal;
b) estabelecer um identificador único a partir das características extraídas da ferradura existente na parte inferior da cabeça dos cágados;

disponibilizar uma interface gráfica que permita ao usuário realizar o cadastro e consulta dos cágados.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está dividido em quatro capítulos. O primeiro capítulo apresenta os objetivos e a motivação para desenvolvimento do trabalho. O segundo capítulo trata da fundamentação teórica do trabalho, explicando os principais conceitos e técnicas utilizadas no desenvolvimento do protótipo. No terceiro capítulo é descrita a arquitetura do trabalho por meio de diagramas, o detalhamento da implementação do protótipo e os resultados dos testes obtidos na validação do mesmo. As conclusões e limitações levantadas a partir do trabalho, assim como sugestões para trabalhos futuros são demonstradas no quarto e último capítulo.

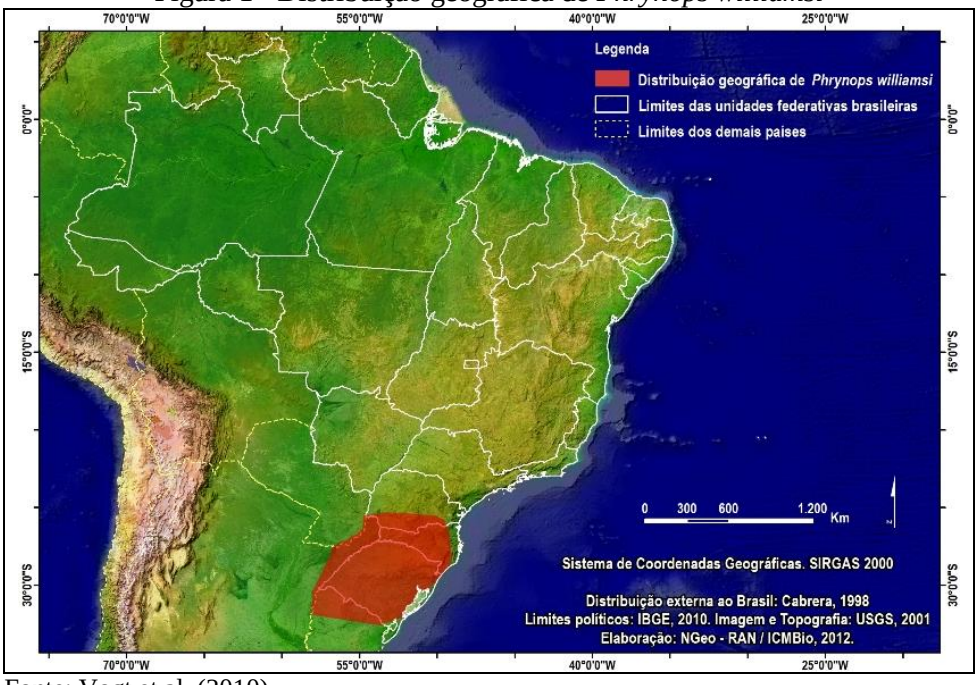
2 FUNDAMENTAÇÃO TEÓRICA

Na seção 2.1 são apresentadas algumas características da espécie *Phrynops williamsi*, espécie de cágado a ser identificada pelo protótipo. Já a seção 2.2 demonstra a teoria sobre os descritores de Fourier, apresentado conceitos e as equações utilizadas no cálculo dos mesmos. A seção 2.3 introduz o conceito de segmentação de imagens, técnica utilizada na separação das regiões de interesse do plano de fundo da imagem. A seção 2.4 presenta o conceito de morfologia matemática, detalhando as operações de dilatação e erosão. Por fim, na seção 2.5 são apresentados os trabalhos correlatos ao protótipo desenvolvido.

2.1 PHRYNOPS WILLIAMSI

Os cágados da espécie *Phrynops williamsi*, mais conhecidos como Cágado-de-ferradura, habitam áreas com altitude inferior a 500 metros da costa leste de Santa Catarina e Rio Grande do Sul, assim como a metade norte do Uruguai, conforme é monstrado na Figura 1 (RHODIN; MITTERMEIER, 1983, p. 60). São animais de hábitos diurnos que vivem em rios de grande porte com margens lodosas ou rochosas (TEIXEIRA; RIBAS, 1999 apud VOGT et al., 2010). Segundo Rhodin e Mittermeier (1983, p. 70), são encontrados em rios de correnteza rápida com rochas sobressalentes ao nível da água.

Figura 1 - Distribuição geográfica de *Phrynops williamsi*



Fonte: Vogt et al. (2010).

Os espécimes de *Phrynops williamsi* vivem em áreas de brejos, restingas e lagoas de baixadas associadas à Mata Atlântica. Mesmo a espécie tendo uma extensão de ocorrência no pais de cerca 377.558,10 km², é considerada rara, sendo difícil encontrar um exemplar

(VOGT et al., 2010). Ainda segundo os autores, o problema de identidade taxonômica inviabiliza o conhecimento sobre tamanho, estrutura e distribuição da população. Associado a esse fato, Spier et al. (2014, p. 57) comentam que essa espécie pode estar correndo risco de extinção local na região dos rios Pelotas, Uruguai e do Peixe devido às suas exigências de *habitat* e a problemas como a poluição, desmatamento e a implantações de usinas hidrelétricas.

Em relação a carapaça desta espécie, as fêmeas atingem um tamanho máximo de comprimento de 33cm, e nos machos, este tamanho chega a 29,4cm (FREIBERG, 1970 apud VOGT et al., 2010). Possui formato ligeiramente oval com comprimento 15% maior que a largura em indivíduos juvenis e, em adultos essa diferença chega a 46%, conforme ilustra a Figura 2a (RHODIN; MITTERMEIER, 1983, p. 60). Sua coloração é marrom com pequenas reticulações pretas que se estendem por toda a superfície e apresentam finas bordas de cor laranja amarelado (VAZ-FERREIRA;SIERRA, 1960 apud RHODIN;MITTERMEIER, 1995, p. 60).

Na superfície ventral da cabeça, apresentam um par de barbatanas normalmente sem pigmentação. Seu plastrão é amplo e de coloração amarelada, seu pescoço é curto e a cabeça relativamente estreita quando comparada ao seu plastrão, conforme mostra a Figura 2b.

Figura 2 - *Phrynops williamsi* capturada acidentalmente em redes

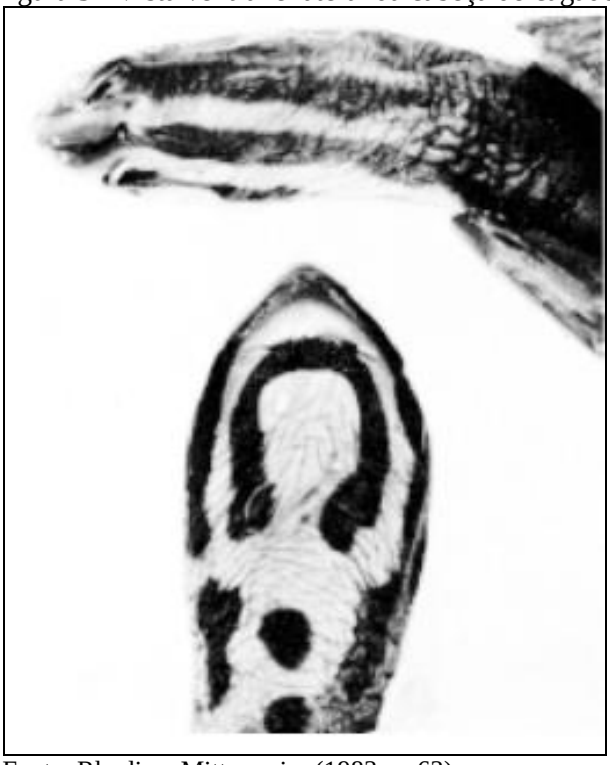
(a) visão da carapaça

(b) visão da cabeça e pescoço

Fonte: Spier et al. (2014, p. 57).

Outra característica desta espécie é a existência de três listras pretas localizadas na região da cabeça, sendo que uma delas, localizada na região ventral do pescoço, possui o formato de uma ferradura. Estas listras normalmente não se conectam entre si conforme ilustrado na Figura 3 (RHODIN; MITTERMEIER, 1983, p. 60).

Figura 3 - Vista ventral e lateral da cabeça do cágado



Fonte: Rhodin e Mittermeier (1983, p. 63).

A partir da Figura 3, pode-se observar que a primeira faixa fica localizada na parte superior da cabeça e segue das narinas até a base do pescoço. Já a segunda faixa, única para cada cágado, fica situada na parte inferior da cabeça e possui o formato de uma ferradura. Enquanto, a terceira faixa segue aproximadamente da boca até o mesmo nível da faixa em formato de ferradura (RHODIN; MITTERMEIER, 1983, p. 63).

Segundo Spier et al. (2014, p. 58), ações direcionadas para a conservação, monitoramento e manejo desta espécie de cágado são prejudicadas devido a carência de informações básicas que permitem estabelecer tais ações, como período de reprodução, ambiente utilizado para tal, oviposição e alimentação. Sendo que, o objetivo da monitoração de uma determinada espécie é compreender a faixa etária, densidade, razão sexual e taxas de sobrevivência da sua população. Através da monitoração, também é possível determinar a variação no número de indivíduos da espécie durante um período de tempo (BALESTRA et al., 2015, p. 116). Ainda segundo os autores, uma monitoração bem planejada permite aos biólogos, avaliar o impacto de determinada prática de manejo e conservação, doenças, caça e conservação de hábitat de uma espécie.

A monitoração dos quelônios é realizada por meio da marcação dos espécimes capturados. A variedade de técnicas de marcação deste grupo de animais é grande. Recomenda-se que estes procedimentos sejam realizados com materiais esterilizados, presando pela biossegurança do animal. A técnica de marcação mais tradicional de quelônios

é a realização de furos ou cortes nos escudos marginais da carapaça (BALESTRA et al, 2015, p. 128).

O método de marcação onde são realizados cortes nos escudos marginais da carapaça do animal possui baixo custo e possui a desvantagem do possível desaparecimento das marcas devido a regeneração natural. Esta desvantagem é mais evidente em indivíduos muito jovens ou recém nascidos. Em adultos, o corte é realizado na carapaça através de pequenas serras ou seguetas (Figura 4a). Recomenda-se que os cortes tenham o formato quadrado ou retangular (Figura 4b), para não sejam confundidos com possíveis mordidas de outros animais (BALESTRA et al, 2015, p. 128).

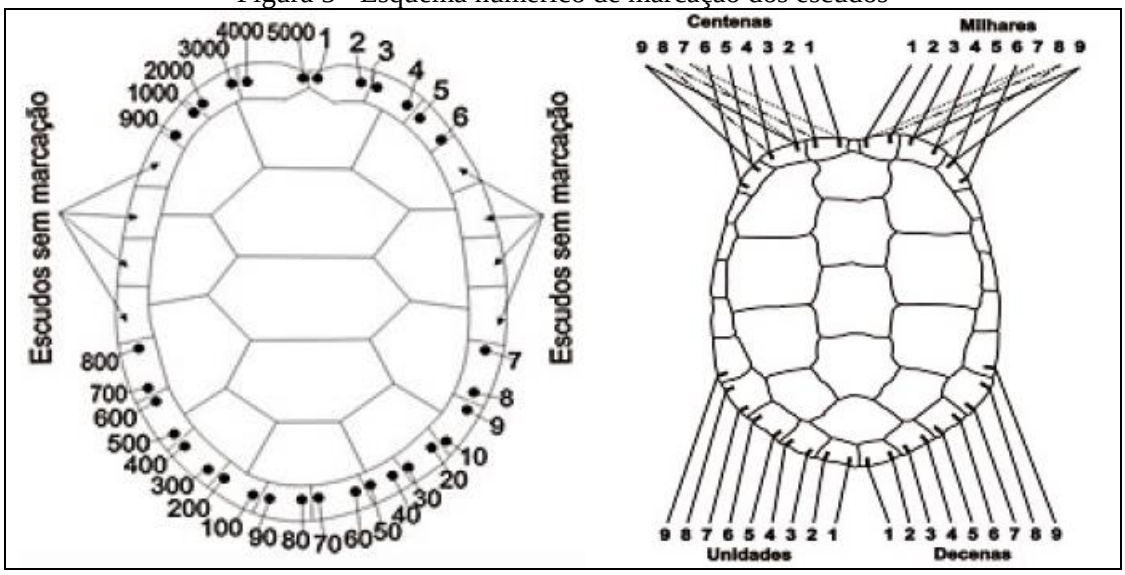
Figura 4 - Marcação com o uso de serras manuais

(a) serra utilizada na marcação | (b) corte realizado com serra

Fonte: Balestra et al (2015, p. 129).

Caso o objetivo do estudo seja apenas averiguar a captura e recaptura sem a necessidade de individualizar os espécimes por meio da marcação, basta que um escudo marginal seja escolhido e marcado. No entanto, quando pretende-se avaliar o crescimento e a movimentação dos espécimes, é necessário que estes sejam individualizados. Com esta técnica de marcação é possível individualizar os espécimes capturados por meio de uma codificação numérica gerada de acordo com a disposição dos cortes nos escudos (BALESTRA et al, 2015, p. 128). A Figura 5 exibe dois esquemas de numeração utilizados.

Figura 5 - Esquema numérico de marcação dos escudos



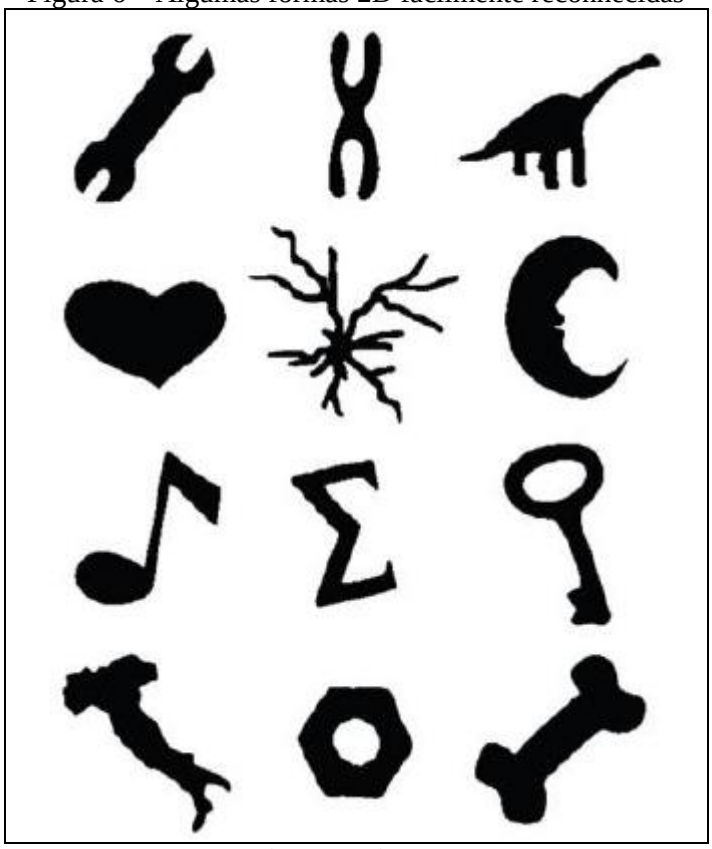
Fone: Balestra et al. (2015, p. 129).

Como demonstrado na Figura 5, este esquema permite a geração de um grande número de combinações diferentes. O identificador numérico é gerado a partir da disposição dos cortes nos escudos marginais. A Figura 5 ilustra um esquema de marcação onde os escudos são divididos em quatro grupos, sendo que cada um desses é responsável por representar a unidade, dezena, centena e a parte milhar do identificador. Cada grupo deve conter cinco escudos, sendo que o corte pode ser realizado no lado esquerdo ou direito (BALESTRA et al, 2015, p. 128).

2.2 DESCRITORES DE FOURIER

Dentre os aspectos envolvidos na percepção visual, a forma ou formato dos objetos possui grande destaque. De forma geral, formas 2D normalmente são modelos de objetos pertencentes a uma mesma classe. Essa afirmação pode ser observada na Figura 6, onde os objetos podem ser prontamente reconhecidos mesmo na falta de informações como cor, textura, profundidade e movimento (COSTA; CESAR JR., 2000, p. 2).

Figura 6 – Algumas formas 2D facilmente reconhecidas

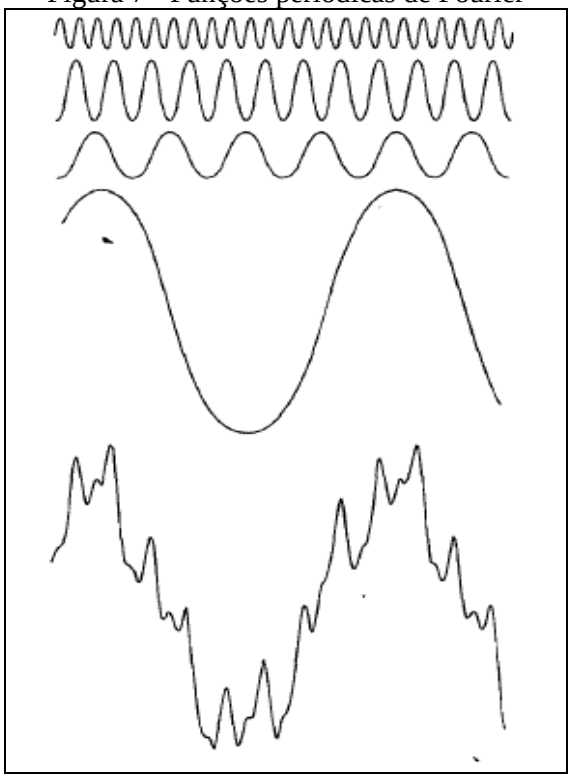


Fonte: Costa e Cesar Jr. (2000, p. 4).

Zahn e Roskies (1972, p. 1) citam que a discriminação de formas planas é um dos mais familiares e fundamentais problemas no reconhecimento de padrões. Segundo Costa e Cesar Jr. (2000, p. 459), os Descritores de Fourier (FDs) são um dos principais métodos de representação de formas utilizados em aplicações de visão computacional e de reconhecimento de padrões.

A contribuição mais conhecida do matemático francês Jean Baptiste Joseph Fourier é a Série de Fourier, apresentada em 1822. Nesta contribuição ele afirmou que qualquer função periódica, independente da sua complexidade, pode ser representada (Figura 7) por uma soma de senos e/ou cossenos em diferentes frequências, sendo cada um deles multiplicado por um coeficiente distinto. Quando a função não é periódica, porém finita, esta ainda pode ser representada pela integral de senos e/ou cossenos multiplicados por uma função de ponderação. Neste caso a formulação é chamada de Transformada de Fourier (GONZALES; WOODS, 2008, p. 222).

Figura 7 - Funções periódicas de Fourier



Fonte: Gonzales e Woods (2008, p. 223).

A Figura 7 demonstra a contribuição de Fourier, onde uma função periódica pode ser representada pela somatória de senos e cossenos. A quinta função demonstrada diz respeito a somatória das quatro funções superiores.

A ideia básica por traz desta abordagem consiste em representar a forma de um objeto em um sinal uni ou bidimensional, aplicar a transformada de Fourier e calcular os descritores a partir desta representação (COSTA; CESAR JR., 2000, p. 459). Segundo Gonzales e Woods (2008, p. 842), poucos descritores de Fourier são suficientes para capturar a essência básica do contorno da forma e, podem ser utilizados como base na distinção de fronteiras de diferentes formatos, visto que estes coeficientes ainda carregam as informações básicas da forma.

Uma das características mais importantes destas representações é de que uma função expressa por meio de uma série ou transformada de Fourier, pode ser completamente reconstruída através do processo inverso (GONZALES; WOODS, 2008, p. 222). O Quadro 1 demonstra a equação da *Discrete Fourier Transform* (DFT), utilizada na obtenção da transformada de Fourier.

Quadro 1 - Fórmula da DFT

$$F(u) = \sum_{x=0}^{M-1} f(x)e^{-j2\pi ux/M} \qquad u = 0,1,2,\ldots,M-1$$

Fonte: Gonzales e Woods (2008, p. 244).

O Quadro 2 por sua vez demonstra sua inversa, a *Inverse Discrete Fourier Transform* (IFDT), utilizada na reconstrução da função.

Quadro 2 - Fórmula da IDFT

$$f(x) = \frac{1}{M} \sum_{x=0}^{M-1} F(u)e^{\,j2\pi ux/M} \qquad x = 0,1,2,\ldots,M-1$$

Fonte: Gonzales e Woods (2008, p. 244).

Nas equações demonstradas, `M` representa a quantidade de valores da transformada, `x` o enésimo valor da transformada e `u` o enésimo descritor.

Segundo (GONZALES; WOODS, 2008, p. 840), o contorno de `K` pontos de um objeto sobre um plano cartesiano pode ser representado através de pares de coordenadas (x_0, y_0), (x_1, y_1), ..., (x_k, y_k). A partir desta representação, cada par de coordenada pode ser representado através de um número complexo, conforme equação demonstrada no Quadro 3.

Quadro 3 - Coordenada como número complexo

$$s(k) = x(k) + jy(k)$$

Fonte: Gonzales e Woods (2008, p. 840).

Nesta equação, a coordenada `x` é representada pela parte real da equação, enquanto que a coordenada `y` é representada pela parte imaginária. Ainda segundo os autores, uma das vantagens desta representação é que a dimensão do problema foi reduzida de 2D para 1D. A seção 2.2.1 descreve outro tipo de representação da forma proposta por El-ghazal, Basir e Belkasim (2009, p. 576), chamada de *Farthest Point Distance* (FPD).

2.2.1 *Farthest Point Distance*

A assinatura de forma apresentada por El-ghazal, Basir e Belkasim (2009, p. 576), chamada de *farthest point distance*, consiste na utilização da distância do ponto analisado até a seu centroide, somada à distância deste centroide até o ponto mais distante do ponto analisado. Segundo os autores, uma das vantagens desta abordagem é a consideração das distâncias dos cantos da figura, elemento muito importante no sistema de visão humana e

desconsiderado por muitas das técnicas existentes (EL-GHAZAL; BASIR; BELKASIM, 2009, p. 576).

A fórmula utilizada no cálculo da assinatura FPD em um ponto do contorno (x(u), y(u)) é demonstrada no Quadro 4.

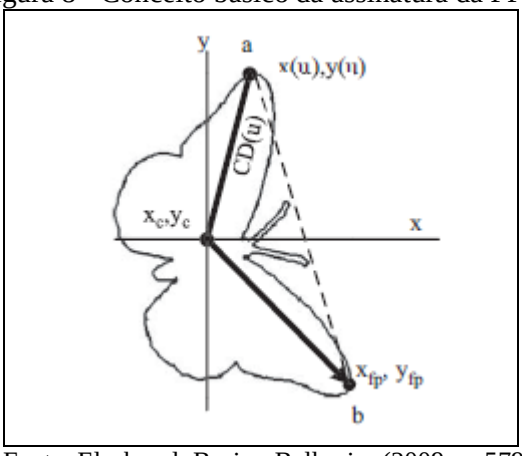
Quadro 4 - Equação da FPD

$$FPD(u) = \sqrt{([x(u) - x_c]^2 + [y(u) - y_c]^2)} + \sqrt{\left([x_{fp}(u) - x_c]^2 + \left[y_{fp}(u) - y_c\right]^2\right)}$$

Fonte: El-ghazal, Basir e Belkasim (2009, p. 579).

Na equação demonstrada, $\left(x_{fp}(u), y_{fp}(u)\right)$ diz respeito ao ponto mais distante de $(x(u), y(u))$ e (x_c, y_c) representa o centroide do objeto. A Figura 8 ilustra um exemplo do conceito básico da assinatura do ponto mais distante.

Figura 8 - Conceito básico da assinatura da FPD



Fonte: El-ghazal, Basir e Belkasim (2009, p. 579).

A Figura 8 demonstra a obtenção da assinatura FPD do ponto `a`, sendo que o ponto mais distante de `a` é representado pelo ponto `b`. Pode ser observado na imagem, que a assinatura é formada pela distância do ponto `a` até o centroide junto da distância do centroide até o ponto `b`.

2.3 SEGMENTAÇÃO DE IMAGENS

A segmentação de imagens consiste no processo de subdividir uma imagem nas suas regiões de interesse ou objetos (GONZALES; WOODS, 2008, p. 711). Segundo (COSTA; CESAR JR., 2000, p. 235), este processo permite a detecção e separação do conjunto de pixels que constituem os objetos que estão sobre análise. A segmentação geralmente é descrita, de forma análoga a um processo visual, como a separação do primeiro plano do plano de fundo (RUSS, 1995, p. 345).

Segundo (RUSS, 1995, p. 345), é um dos passos mais utilizados na redução da imagem em informação. É também uma das etapas mais difíceis do processamento de imagens quando realizado sobre imagens não triviais (GONZALES; WOODS, 2008, p. 711). (COSTA; CESAR JR., 2000, p. 235) comentam que a segmentação é necessária quando deseja-se realizar a análise dos objetos da imagem.

No momento em que os objetos ou regiões de interesse da imagem foram detectados, é importante suspender a segmentação pois a precisão deste processo determina o eventual sucesso ou falha na análise da imagem (GONZALES; WOODS, 2008, p. 711). Segundo (COSTA; CESAR JR., 2000, p. 235), a realização desta tarefa resulta em uma versão binarizada da imagem de entrada.

Um dos métodos de segmentação mais utilizados em aplicações de segmentação de imagens é a limiarização (COSTA; CESAR JR., 2000, p. 248). Segundo (GONZALES; WOODS, 2008, p. 760), a alta velocidade computacional, baixa complexidade de implementação e suas propriedades intuitivas são alguns dos fatores que corroboram para este fato.

O algoritmo de limiarização utiliza a similaridade da intensidade dos pixels, ou seja, o particionamento de uma imagem é realizado sobre regiões similares de acordo com um critério pré-definido (GONZALES; WOODS, 2008, p. 712). A ideia por trás deste método consiste em definir um valor de limiar, percorrer os pixels da imagem e marcar os pixels que possuírem valor maior que este limiar como sendo pontos do objeto. Os pixels não marcados são então considerados pontos do plano de fundo (GONZALES; WOODS, 2008, p. 760). Esta associação é dada pela equação demonstrada no Quadro 5.

Quadro 5 - Equação de limiarização

$$g(x,y) = \begin{cases} 1 & se\ f(x,y) > T \\ 0 & se\ f(x,y) \leq T \end{cases}$$

Fonte: Traduzido de Gonzales e Woods (2008, p. 760).

Na equação demonstrada, `T` diz respeito ao valor do limiar, `f(x,y)` à função do valor do pixel e `g(x,y)` à imagem segmentada.

2.4 MORFOLOGIA MATEMÁTICA

A morfologia matemática tem origem no trabalho dos matemáticos franceses J. Serra e G. Matheron publicado no meio dos anos 1960 e tem sido aplicada a uma série de problemas de processamento de imagens (COSTA; CESAR JR., 2000, p. 255). Segundo Gonzales e Woods (2008, p. 650), a morfologia matemática oferece uma abordagem poderosa e unificada

para inúmeros problemas de processamento de imagens. Conforme Costa e Cesar Jr. (2000, p. 255), pré-processamento de imagens, filtros de ruídos, detecção de formas e associação de padrões são alguns dos problemas onde a morfologia matemática vem sendo aplicada.

A mais vasta classe de operações de processamento de imagens binárias é por vezes coletivamente descrita como operações morfológicas (ASTOLA; DOUGHERTY, 1994; CHERMANT; COSTER, 1985; SERRA, 1982 apud RUSS, 1995, p. 433). Essa classe é composta pelas operações de erosão, dilatação e suas derivadas. Elas, podem ser entendidas como operações de adição e remoção de pixels em imagens (RUSS, 1995, p. 434).

Todas as operações morfológicas são baseadas em um elemento estruturante (EE), espécie de sub-imagem utilizada na inspeção das propriedades de interesse de uma imagem. Todo EE deve possuir uma origem, geralmente localizada no centro de gravidade do mesmo e deve variar de acordo com o problema a ser resolvido. Quando as operações são realizadas sobre imagens, é necessário que os elementos estruturantes sejam *arrays* retangulares (GONZALES; WOODS, 2008, p. 651).

As Seções 2.4.1 e 2.4.2 descrevem, respectivamente, as operações de erosão e dilatação.

2.4.1 Erosão

Conforme comentado anteriormente, a operação de erosão morfológica envolve a imagem binária a ser processada e um elemento estruturante. O resultado da operação de erosão é a redução dos objetos contidos na imagem, e esta deve variar de acordo com o EE utilizado (COSTA; CESAR JR., 2000, p. 259). Segundo Costa e Cesar Jr. (2000, p. 260), a erosão é comumente aplicada no pré-processamento de imagens, pois pode ser utilizada no processo de redução de ruídos.

A erosão morfológica consiste em transladar o EE sobre os pixels de um objeto e manter os pixels onde o EE deslocado está completamente contido dentro da mesma (COSTA; CESAR JR., 2000, p. 259). Esta operação pode ser representada pela equação demonstrada no Quadro 6.

Quadro 6 - Equação da erosão morfológica

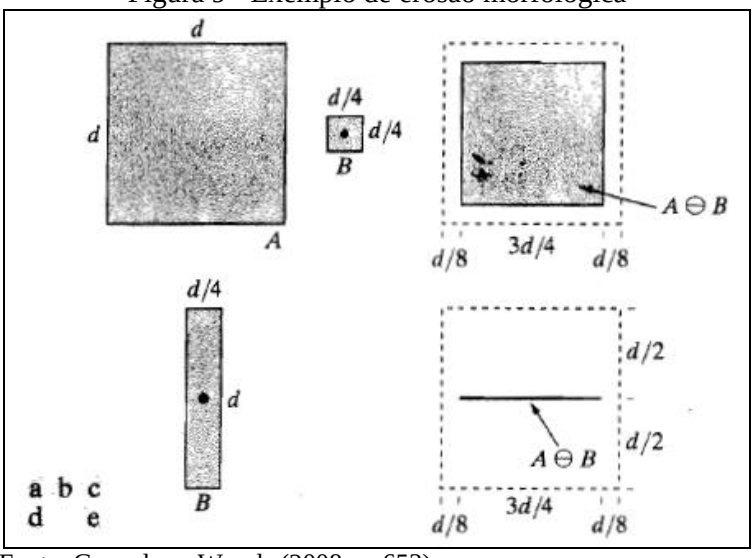
$$A \ominus B = \{z | (B)_z \subseteq A\}$$

Fonte: Gonzales e Woods (2008, p. 653).

Na equação demonstrada, `A` diz respeito a imagem a ser processada e `B` ao EE transladado pelo conjunto de pixels `z`. Conforme demonstrado na equação, a translação de `A`

por B é o conjunto de todos os pixels z, onde B transladado por z, está contido em A. Na Figura 9 é ilustrado um exemplo do resultado da operação de erosão.

Figura 9 - Exemplo de erosão morfológica



Fonte: Gonzales e Woods (2008, p. 653).

Os elementos ilustrados na Figura 9 seguem a mesma nomenclatura da equação mostrada no Quadro 6. A imagem (c) demonstra o resultado da erosão de A por B. Conforme comentado anteriormente, a utilização de um EE de diferentes dimensões (d) pode ser gerar um resultado distinto conforme demonstrado por (e). As linhas tracejadas não dizem respeito à operação de erosão e são utilizadas apenas para referência (GONZALES; WOODS, 2008, p. 653).

2.4.2 Dilatação

De forma semelhante a operação de erosão, a operação de dilatação morfológica envolve a imagem binária a ser processada e um elemento estruturante (COSTA; CESAR JR., 2000, p. 255). De acordo com (GONZALES; WOODS, 2008, p. 655), de modos oposto a operação de erosão, a operação de dilatação aumenta ou engrossa os objetos da imagem.

Segundo Gonzales e Woods (2008, p. 655), uma das aplicações mais simples da dilatação é o preenchimento de lacunas. Costa e Cesar Jr. (2000, p. 255) afirmam que o preenchimento dos buracos ou lacunas dependem do formato e ponto de origem do EE utilizado. A operação morfológica de dilatação é representada pela equação apresentada no Quadro 7.

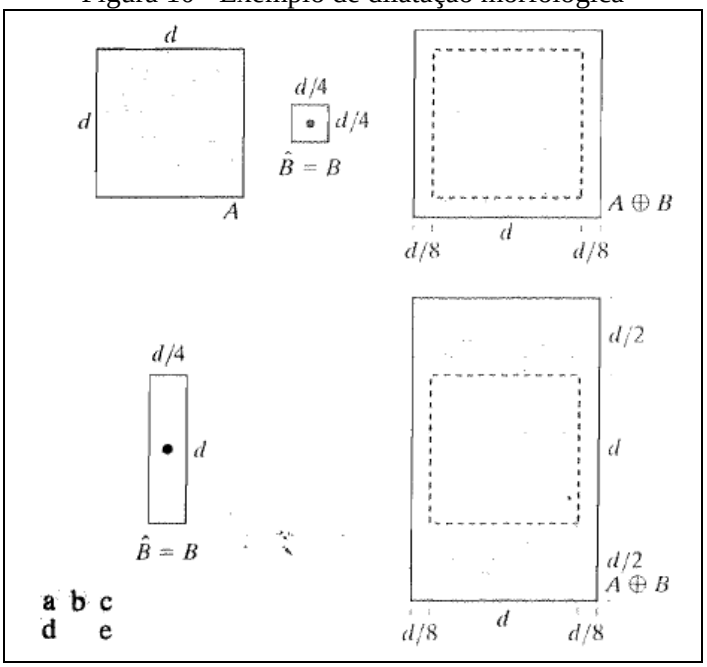
Quadro 7 - Equação da dilatação morfológica

$$A \oplus B = \{z | (\hat{B})_z \cap A \neq \emptyset\}$$

Fonte: Gonzales e Woods (2008, p. 655).

Nesta equação, A diz respeito à imagem analisada, B ao elemento estruturante e z o conjunto de pixels pelos quais o EE será deslocado. Esta equação consiste em refletir o EE (B) sobre sua origem e deslocar essa reflexão pelo conjunto de pixels z. A dilatação de A por B é o conjunto de todos os deslocamentos z, onde $\hat{B}$ e A se sobrepõem por ao menos um pixel (GONZALES; WOODS, 2008, p. 655). Na Figura 10 é ilustrado um exemplo da operação de dilatação morfológica.

Figura 10 - Exemplo de dilatação morfológica



Fonte: Gonzales e Woods (2008, p. 656).

Os elementos ilustrados na Figura 10 seguem a mesma nomenclatura da equação mostrada no Quadro 7. A imagem (c) demonstra o resultado gerado pela operação de dilatação de (a) por (b). A imagem (e) demonstra o resultado da mesma operação, porém utilizando um EE de diferentes dimensões, sendo maior verticalmente do que horizontalmente. Pode ser observado que o resultado dilatação também é maior verticalmente. As linhas pontilhadas nas figuras (c) e (e) mostram o contorno objeto original para referência, estas não fazem parte da operação de dilatação (GONZALES; WOODS, 2008, p. 655).

2.5 TRABALHOS CORRELATOS

Não foram encontrados trabalhos diretamente relacionados ao objeto de estudo deste tema. Abaixo, são apresentados e discutidos alguns trabalhos que mais se assemelham ou que utilizaram técnicas que por ventura podem ser utilizadas para tentar criar um identificador único para cágados da espécie *Phrynops williamsi*. A subseção 2.5.1 descreve o trabalho proposto por Baboo e Vigneswari (2014). A subseção 2.5.2 descreve o artigo escrito por

Beugeling e Branzan-Albu (2014). A subseção 2.5.3 descreve o trabalho de conclusão de curso de Cassaniga (2012) que mesmo não tendo relação com cágados ou tartarugas, serve como base para a implementação e utilização das técnicas de processamento e extração de características de imagens.

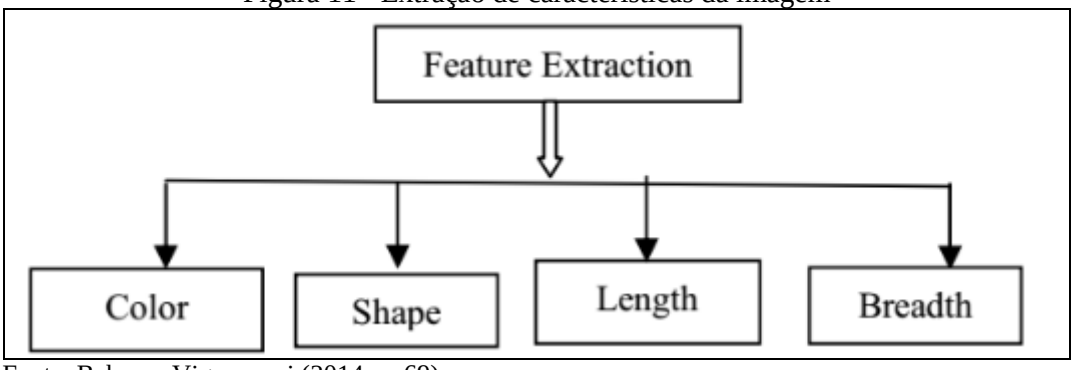
2.5.1 *Identification of olive ridley turtle using feature extraction*

Baboo e Vigneswari (2014) apresentaram em seus estudos, uma ferramenta para identificação de tartarugas da espécie *Lepidochelys olivácea* no ambiente MATLAB. O objetivo da ferramenta proposta é fazer a identificação das tartarugas através da extração de suas características a partir de uma imagem de entrada. Atualmente, a identificação destas tartarugas é realizada por rastreamento e aplicação de *tags*, um processo caro e custoso. O método proposto no estudo reduz os fatores de custo e tempo.

A primeira etapa do processo de identificação é o pré-processamento da imagem de entrada. Nesta etapa, os ruídos da imagem são removidos através do método `medfilt2`, que aplica o filtro da mediana sobre a imagem. Segundo Costa e Cesar Jr. (2000, p. 235), o filtro da mediana é uma operação não linear comumente utilizada na remoção de ruídos impulsivos, onde os pixels ruidosos se diferem significantemente dos pixels vizinhos.

Com a imagem pré-processada é iniciada a etapa de extração das características da imagem, conforme mostra a Figura 11.

Figura 11 - Extração de características da imagem



Fonte: Baboo e Vigneswari (2014, p. 69).

Conforme o fluxo demonstrado na Figura 11, o programa proposto extrai as características de Cor, Forma, Comprimento e Largura da imagem. Cada uma destas é extraída, calculada e posteriormente comparada com valores de limiares pré-definidos que irão estabelecer se a tartaruga da imagem é da espécie *Lepidochelys olivácea*.

A extração da característica de Cor da imagem é realizada por meio do método *colormap*. Este método retorna o mapa de cores da imagem, uma matriz de três posições de

largura contendo a intensidade de cada um dos canais de cores vermelho, verde e azul da imagem (MATHWORKS, 2016).

Para extração da forma, é utilizado o método `roipoly` que separa a região de interesse da imagem. Este método ativa uma ferramenta poligonal interativa que permite que o usuário faça a seleção da região de interesse a partir da seleção dos vértices do polígono (MATHWORKS, 2016). Quando esta região se encontra dentro do limiar a tartaruga da imagem é da espécie *Lepidochelys olivácea*.

A função `gininput` é utilizada na extração da largura e comprimento da tartaruga sobre da imagem. Esta função permite que o usuário selecione `n` pontos da imagem através do cursor do mouse e posteriormente retorna as coordenadas x/y destes (MATHWORKS, 2016). As características de Largura e Comprimento são obtidas a partir da distância Euclidiana calculadas sobre os pontos selecionados.

Os autores não apresentaram os resultados obtidos com a ferramenta desenvolvida. Apenas comentam que foi apresentado uma nova metodologia para identificação desta espécie de tartaruga que está em risco de extinção. Também citaram que futuramente as técnicas apresentadas podem ser adaptadas de modo a serem utilizadas na identificação de outras espécies de tartarugas.

2.5.2 *Computer vision-based identification of individual turtles using characteristic patterns of their plastrons*

Beugeling e Branzan-albu (2014) propuseram um algoritmo para identificação automática e individual de tartarugas baseado nas imagens dos seus plastrões (Figura 12). O algoritmo proposto visa diminuir o tempo empregado por cientistas na identificação não-invasiva de tartarugas durante a monitoração de uma determinada população. A Figura 12 demonstra imagens dos plastrões, sendo que a imagem do item "a" é o plastrão de uma tartaruga jovem e a imagem do item "b" o plastrão de uma tartaruga adulta.

Figura 12 - Plastrões de uma tartaruga jovem (a) e de uma tartaruga adulta (b)

(a) (b)

Fonte: Beugeling e Branzan-Albu (2014, p. 203).

No algoritmo desenvolvido por Beugeling e Branzan-albu (2014), a imagem submetida passa por um pré-processamento na qual sua orientação é ajustada de acordo com os pontos informados pelo usuário e tem o fundo removido. Com a imagem pré-processada, um mapa binário é extraído por meio da técnica de limiarização, em que cada componente conexa diz respeito a uma região vermelha do plastrão. Então, características de concavidade das bordas são extraídas após a aplicação do operador morfológico de erosão para gerar o vetor de características utilizado como entrada da rede neural. A rede neural utilizada é do tipo *Multi-Layer Perceptron* e gera como saída o ID da tartaruga cadastrada no banco de dados que mais se assemelha com a imagem submetida.

Os resultados obtidos foram satisfatórios quando as características foram segmentadas manualmente, chegando a uma taxa de 81,3% de acerto. O mesmo não aconteceu quando as características foram segmentadas de modo automático, onde foi obtida uma taxa de acerto de 39,5%. Segundo os autores, um fator que contribuiu com a baixa taxa de acerto no modo de segmentação automática foi a baixa qualidade das imagens utilizadas.

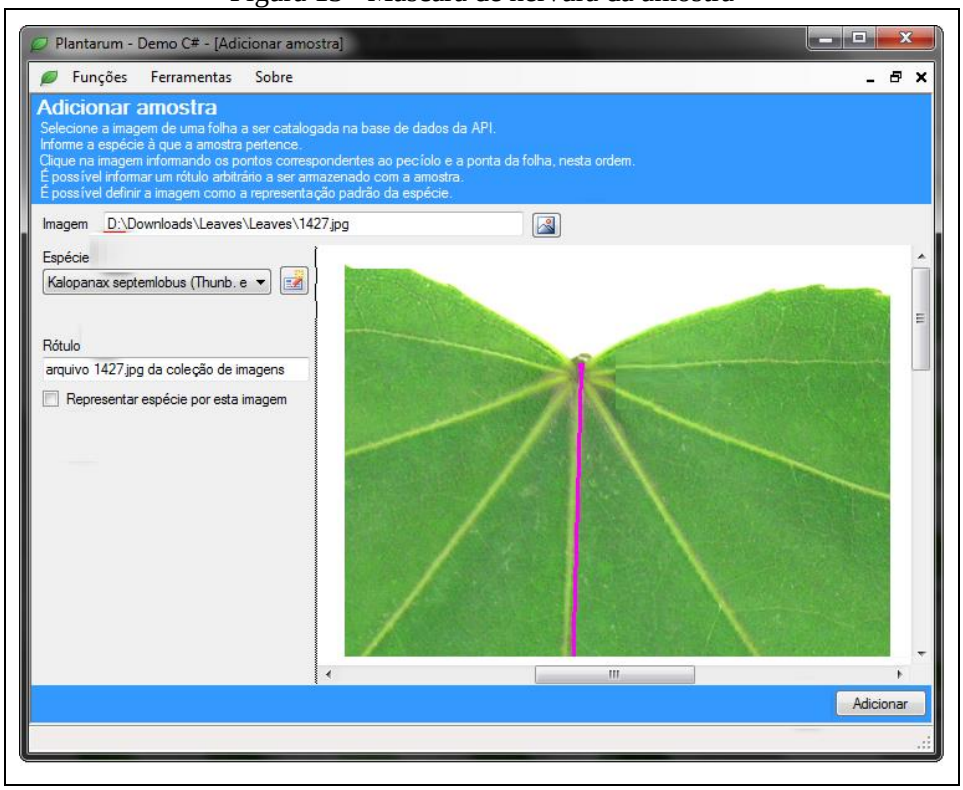
2.5.3 Planatarum: API para reconhecimento de plantas

Cassaniga (2012) apresentou o protótipo de uma API para classificação de plantas a partir de imagens digitais das folhas. O objetivo é auxiliar a árdua tarefa de classificação de espécies de plantas que comumente é realizado por especialistas e reduzir o nível de intervenção humana na extração das características.

A API foi desenvolvida utilizando as linguagens de programação C e C++ e faz uso da biblioteca de visão computacional *Open Source Computer Vision* (OpenCV). A API recebe como entrada a imagem da folha, as coordenadas do pecíolo e da ponta da folha. Esta imagem passa pela etapa de pré-processamento em que é transformada em tons de cinza, é binarizada,

tem o seu contorno extraído e a *bounding box* correspondente a folha é obtida. A partir disso, são extraídas algumas características que serão utilizadas na classificação, tais como circularidade, dispersão, magreza, descritores de Fourier, máscara de nervura (Figura 13), lacunaridade e estatísticas de cor. Após a realização deste procedimento, as características extraídas são enviadas para uma Rede Neural Probabilística, que é responsável por determinar a espécie da planta de entrada por meio da comparação das características de entrada com características de outras espécies previamente cadastradas na base de dados.

Figura 13 - Máscara de nervura da amostra



Fonte: Cassaniga (2012, p. 63).

Os resultados obtidos com a API desenvolvida foram satisfatórios. A taxa de acerto média obtida na identificação das plantas chegou a marca de 94,06%. Sendo que a taxa mínima de precisão, ou seja, a espécie que gerou a menor taxa de acerto, foi de 78,57%. Segundo Cassaniga (2012, p. 98), um fator que contribuiu para o alcance destas taxas foi a utilização de características de naturezas distintas (baseadas em cor, textura e contorno).

3 DESENVOLVIMENTO DO PROTÓTIPO

Neste capítulo são demonstradas as etapas do desenvolvimento do protótipo proposto. Na seção 3.1 são apresentados os principais requisitos. A seção 3.2 demonstra a especificação do protótipo. A seção 3.3 apresenta de forma detalhada a implementação do protótipo. Por fim, a seção 3.4 apresenta os resultados dos testes, sugestões e melhorias.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Os Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF) do protótipo de reconhecimento de cágados da espécie *Phrynops williamsi* são:

a) fazer a identificação da ferradura localizada na parte inferior da cabeça do cágado (Requisito Funcional - RF);
b) calcular o *bounding box* da ferradura, a centroide, dispersão e a circularidade dos componentes da imagem (RF);
c) fazer a identificação da listra circular localizada próxima a ferradura, caso esta exista (RF);
d) identificar a listra da ferradura e da listra circular por meio dos descritores de Fourier (RF);
e) possuir uma interface onde o usuário possa cadastrar, gerar um identificador único para os cágados e demonstrar as etapas do Processamento da Imagem (RF);
f) reconhecer os cágados de acordo com o seus identificadores únicos (RF);
g) armazenar em uma base de dados o identificador único para fins de reconhecimento futuros (RF);
h) ser desenvolvida na linguagem C# para na plataforma Desktop (Requisito Não-Funcional - RNF);
i) utilizar o ambiente de desenvolvimento Microsoft Visual Studio 2013 (RNF);
j) ser desenvolvida utilizando o *wrapper* Emgu CV (RNF).

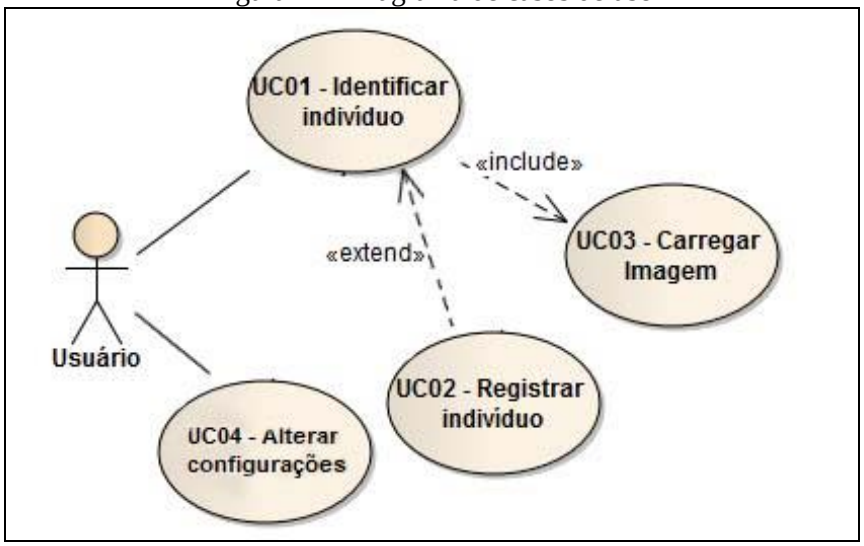
3.2 ESPECIFICAÇÃO

A especificação do protótipo foi representada através da *Unified Modeling Language* (UML), utilizando a ferramenta *Enterprise Architect* (EA). Neste trabalho foram desenvolvidos os diagramas de casos de uso, de pacotes e de atividades.

3.2.1 Diagrama de casos de uso

Nesta seção são apresentados os diagramas de casos de uso do protótipo, exibidos na Figura 14. Identificou-se apenas um ator, denominado `Usuário`, o qual utiliza todas as funcionalidades do protótipo.

Figura 14 - Diagrama de casos de uso



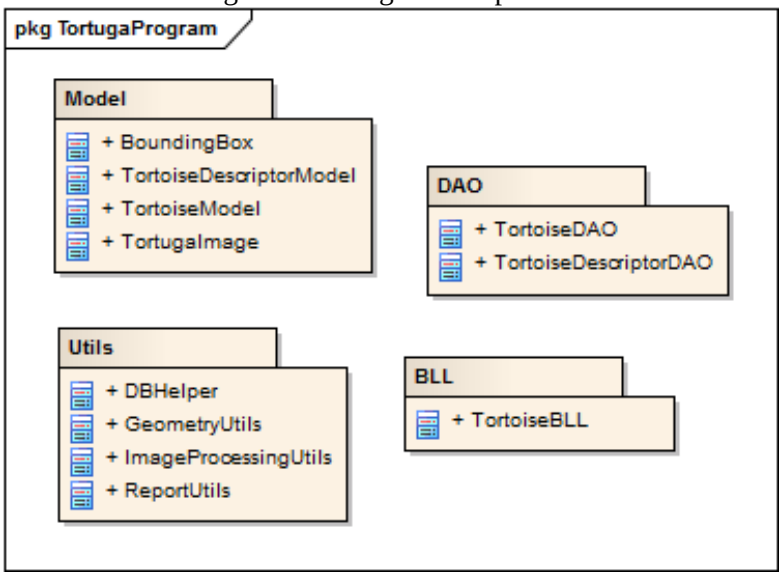
O caso de uso `UC01- Identificar indivíduo` diz respeito ao comportamento de identificar um cágado a partir de uma imagem informada pelo usuário. Após o carregamento da imagem, são ilustradas na tela as etapas realizadas no processamento da imagem. Caso o indivíduo da imagem carregada não esteja cadastrado no banco de dados, é questionado ao usuário se este deseja cadastrá-lo. Este cadastro é realizado no `UC02 – Registrar indivíduo`, no qual o usuário deverá informar dados adicionais do indivíduo `Nome do pesquisador, Cidade/Estado da coleta, Latitude, Longitude, Data da coleta, E-mail do pesquisador e Observações`. Conforme descrito anteriormente, no `UC03 – Carregar Imagem` o protótipo deve exibir uma janela para que o usuário possa fazer a seleção da imagem do cágado em seu computador. Após o carregamento da imagem, é realizado o processo de identificação do indivíduo.

No caso de uso `UC04 – Alterar configurações` o usuário pode realizar a alteração de alguns parâmetros de configuração do sistema ou restaurar os valores padrão do protótipo através do botão `Reset`. O detalhamento dos casos de uso `UC01`, `UC02` e `UC03` está no Apêndice A. O caso de uso `UC04` não é detalhado devido a sua baixa complexidade operacional.

3.2.2 Diagrama de pacotes

O diagrama de pacotes descreve a arquitetura do protótipo separando suas classes em módulos. O protótipo foi dividido em quatro pacotes principais, sendo eles os pacotes `Utils`, `Model`, `BLL` e `DAO`. A Figura 15 ilustra tal diagrama.

Figura 15 - Diagrama de pacotes

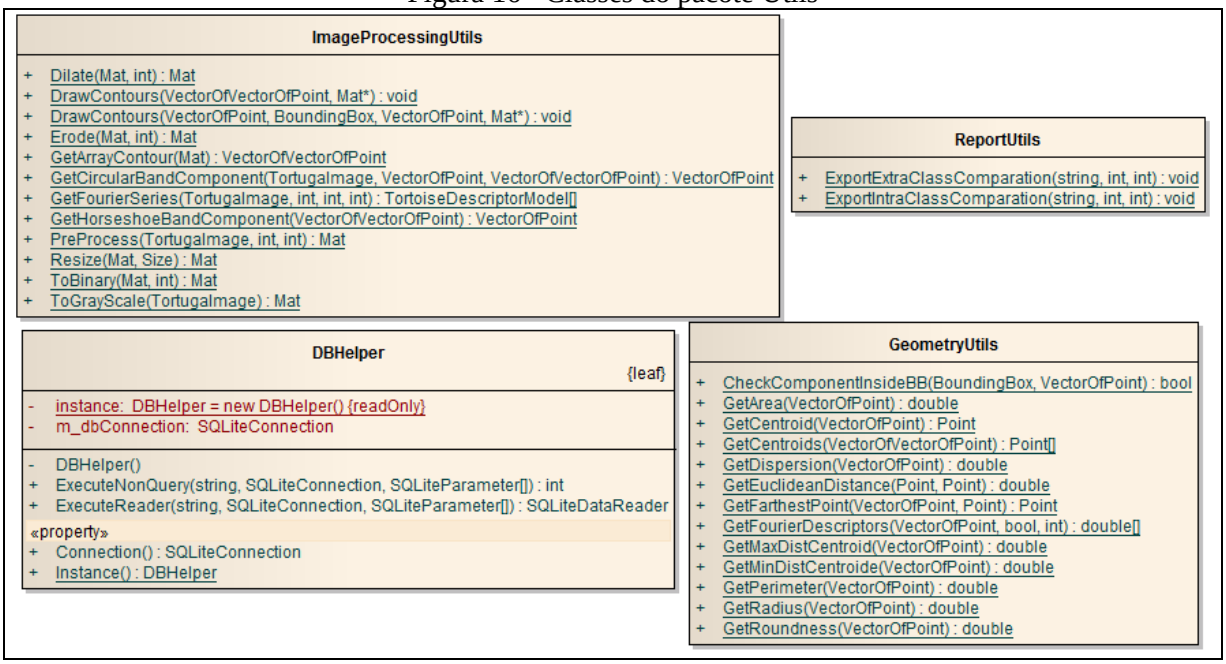


As seções a seguir descrevem cada um dos pacotes individualmente, demonstrando e descrevendo os objetivos das suas classes. A seção a seguir descreve as classes do pacote `Utils`, classes que contém os métodos de processamento de imagens, de geometria, configuração de banco de dados e de relatórios.

3.2.2.1 Pacote `Utils`

O pacote `Utils` engloba os principais métodos do protótipo e suas classes são ilustradas na Figura 16.

Figura 16 - Classes do pacote Utils



A classe `ImageProcessingUtil` contém os métodos de processamento de imagens. Os métodos `Dilate` e `Erode` são responsáveis pela aplicação dos operadores morfológicos de dilatação e erosão, respectivamente. O método `ToBinary`, por sua vez, realiza a binarização da imagem e `ToGrayScale` converte a imagem em escala de cinza. As listras em formato de ferradura e circular localizadas na parte inferior da cabeça dos cágados são obtidas por meio dos métodos `GetHorseshoeBandContour` e `GetCircularBandContour`.

O método `PreProcess` é responsável pelo encapsulamento das chamadas dos métodos utilizados no pré-processamento (`ToGrayScale`, `ToBinary`, `Dilate` e `Erode`). A série de Fourier de uma imagem é calculada pelo método `GetFourierSeries`. Os métodos `GetArrayContour` e `DrawCountour` são responsáveis pela extração e desenho do contorno dos objetos da imagem.

A classe `GeometryUtils` implementa as operações geométricas utilizadas pelo protótipo na identificação dos cágados. Os métodos `GetArea`, `GetCentroid`, `GetDispersion`, `GetParimeter` e `GetRoundness` são responsáveis, respectivamente, pelo cálculo das características de área, centroide, dispersão e perímetro de um objeto da imagem. O cálculo da distância euclidiana entre dois pontos é realizado pelo método `GetEuclideanDistance`.

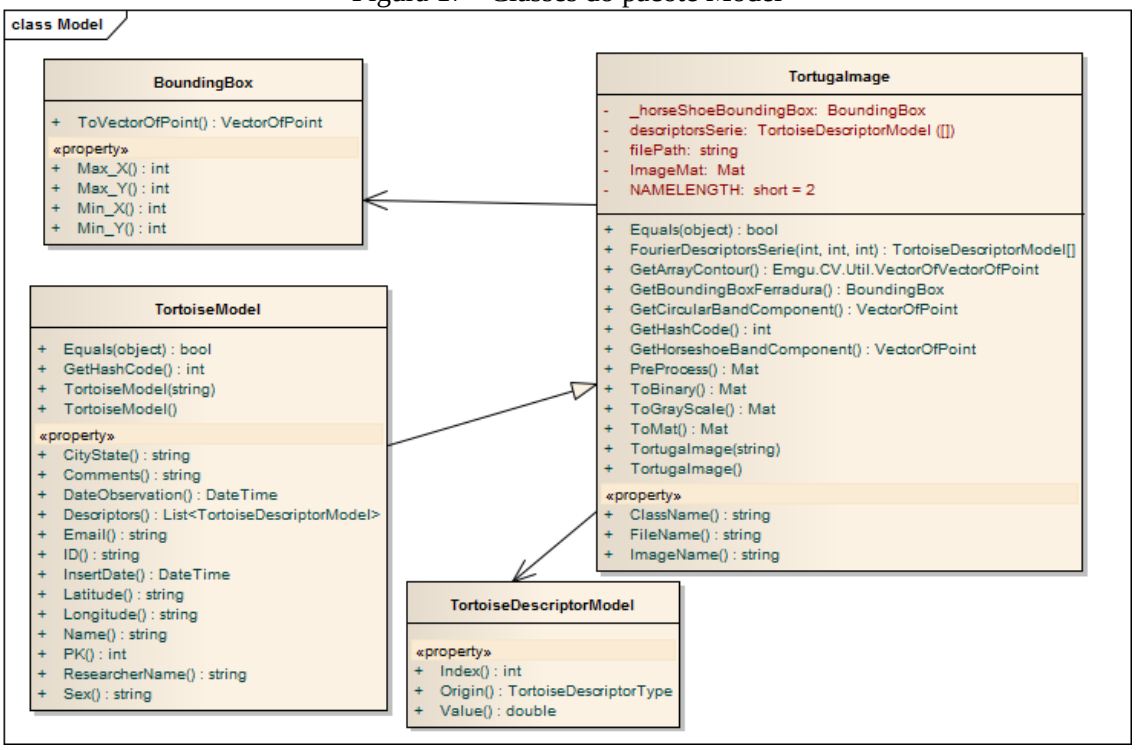
No método `GetComponentInsideBB` é checado se pelo menos um dos pontos do *array* informado está contido dentro do *bounding box* passado por parâmetro. O método `GetFarthestPoint` retorna o ponto mais distante do ponto enviado por parâmetro. As distâncias dos pontos do contorno mais próximo e mais distante do centroide de um objeto são obtidas através dos métodos `GetMaxDistCentroid` e `GetMinDistCentroid`.

A classe `DBHelper` implementa métodos que auxiliam na comunicação com o banco de dados. O método `ExecuteNonQuery` é responsável pela execução de instruções SQL que não possuem retorno no banco de dados. No método `ExecuteReader`, instruções de consultas são executadas no banco de dados. A classe *ReportUtil* é responsável pela exportação do resultado das comparações em um arquivo do tipo texto separados por um delimitador (`.csv`). A próxima seção descreve de forma detalhada as classes do pacote `Model`.

3.2.2.2 Pacote `Model`

As classes demonstradas na Figura 17 fazem parte do pacote `Model` e são representações dos objetos/entidades envolvidas no problema abordado pelo protótipo.

Figura 17 - Classes do pacote Model



A classe `TortugaImage` é a principal classe de dados utilizada pelo protótipo. Ela representa a imagem de um cágado. A série de Fourier da imagem deve ficar armazenada no atributo `descriptorsSerie`. O atributo `filePath` armazena o caminho do arquivo físico enquanto que a propriedade `ImageMat` retorna uma instância da classe `Emgu.CV.Mat` referente à imagem. O *bounding box* da listra em formato de ferradura contida na imagem é armazenada em `_horseShoeBoundingBox`.

Os métodos `GetHorseShoeBandComponent` e `GetCircularBandComponent` fazem a seleção da listra em formato de ferradura e da listra circular, respectivamente. O método

`PreProcess` realiza o pré-processamento da imagem em questão. Os outros métodos desta classe apenas invocam seus respectivos métodos da classe `Utils`.

A classe `TortoiseModel` representa um cágado e herda os atributos, propriedades e métodos da classe `TorugaImage`. Esta classe define e armazena os atributos dos cágados que devem ser informados pelos usuários durante o registro dos mesmos.

A classe `BoundingBox` representa a estrutura de um *bounding box*. Esta possui apenas os atributos `Max_X`, `Max_Y`, `Min_X` e `Min_U`. Nestes são armazenados as quatro coordenadas que quando combinadas geram o *bounding box*.

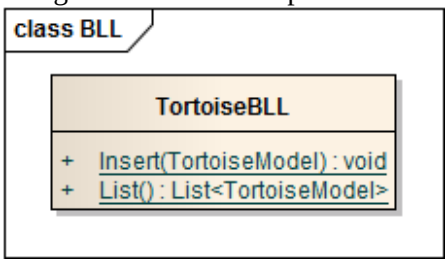
A classe `TortoiseDescriptorModel` é a entidade que representa um descritor de Fourier. Cada descritor calculado deve armazenar a sua origem, indicando se este é proveniente da listra em formato de ferradura ou da listra em formato circular. Esta informação é mantida no atributo `Origin`. O atributo `Index` armazena o índice do descritor na sua série de Fourier. Por fim, o valor do descritor é armazenado no atributo `Value`.

A seção seguir descreve as classes do pacote `BLL`, camada responsável por controlar o fluxo de informações.

3.2.2.3 Pacote `BLL`

O pacote `BLL` engloba as classes que implementam as regras de negócios do protótipo. Estas classes são responsáveis por intermediar o fluxo de dados entre a camada de apresentação e a camada de acesso aos dados. A Figura 18 mostra a classe do pacote `BLL`.

Figura 18 - Classe do pacote `BLL`



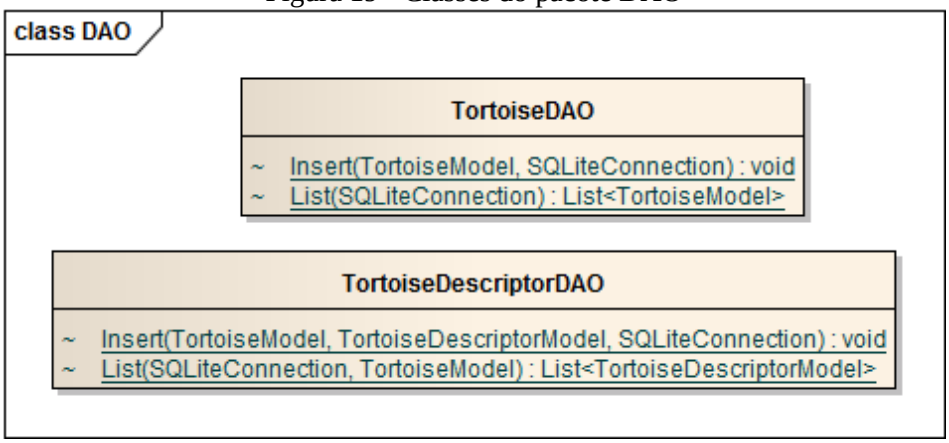
Conforme demonstrado na Figura 18, o pacote `BLL` possui apenas a classe `TortoiseBLL`. Ela é responsável pela implementação das regras de negócios referente ao manuseio e persistência dos cágados na base de dados. O método `Insert` realiza a inserção de um cágado na base de dados, realizando todas as consistências e regras de negócio envolvidas neste procedimento. Nele, os descritores da imagem do cágado são inseridos no banco e a data de inclusão do cágado é preenchida com a data atual. O método `List` é responsável pela consulta e retorno da lista de cágados cadastrados na base de dados.

Na próxima seção são descritas as classes do pacote `DAO`. Estas classes são responsáveis pela persistência e acesso aos dados da aplicação.

3.2.2.4 Pacote `DAO`

O pacote `DAO`, diz respeito a camada de acesso a dados. As classes deste pacote, ilustradas na Figura 19, são responsáveis pela comunicação com o banco de dados.

Figura 19 - Classes do pacote DAO

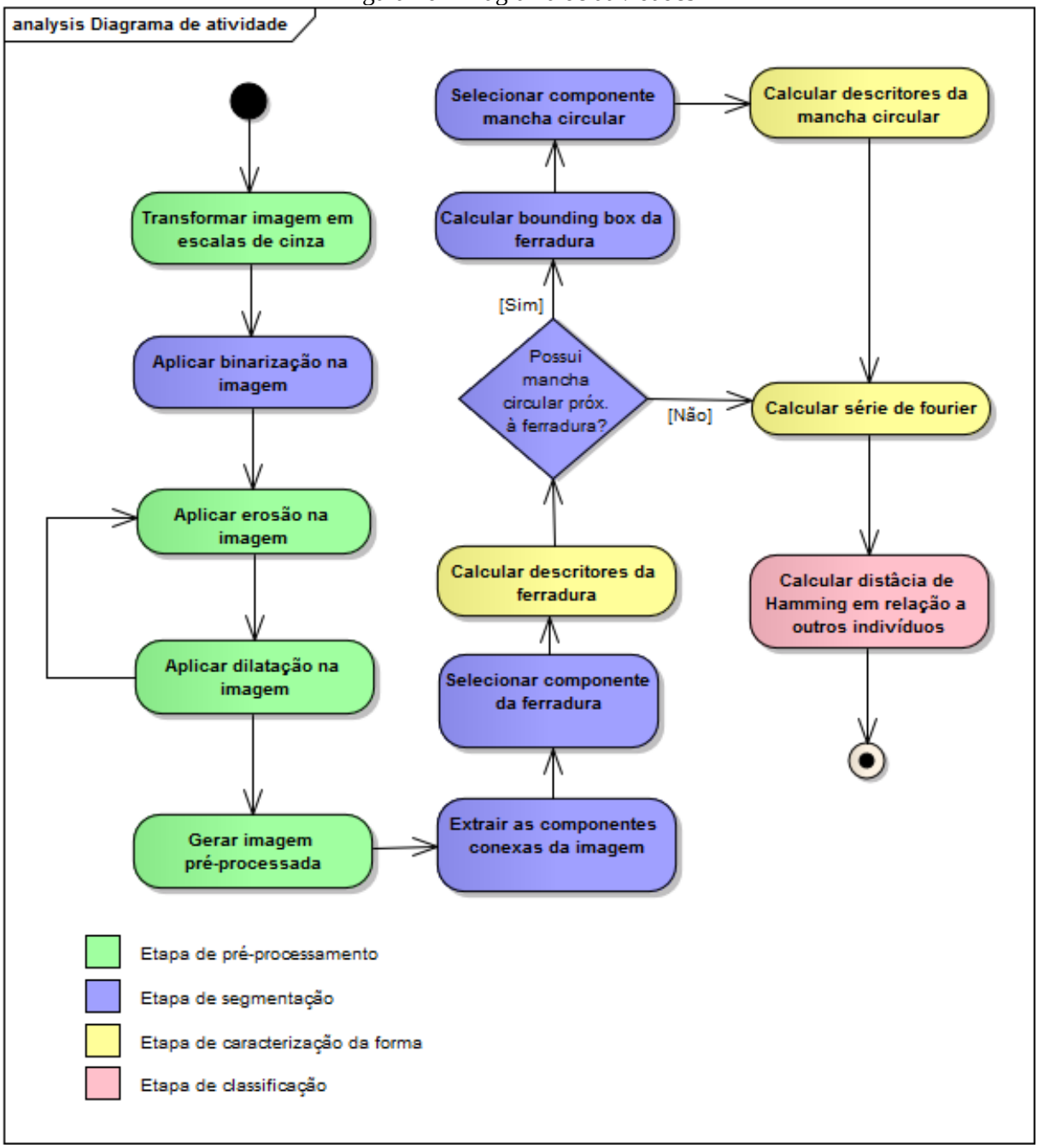


A classe `TortoiseDescriptorDAO` permite a consulta e a persistência dos descritores de Fourier no banco de dados. O método `Insert` realiza a inserção de um descritor, enquanto que `List` realiza a consulta e o retorno da lista dos mesmos. A classe responsável pela inserção e consulta dos cágados é a `TortoiseDAO` por meio dos métodos `Insert` e `List`.

3.2.3 Diagrama de atividades

O diagrama de atividades é responsável por demonstrar o fluxo dos processos realizados pelo protótipo. A Figura 20 mostra as atividades realizadas no processo de identificação dos cágados a partir de uma imagem de entrada.

Figura 20 - Diagrama de atividades



Conforme demonstrado na Figura 20, a primeira etapa do fluxo realizado pelo protótipo de identificação dos cágados é o pré-processamento da imagem de entrada. Esta etapa se inicia com a conversão da imagem em escala de cinza, em que as cores da imagem original são convertidas para escala de cinza. As próximas atividades da etapa de pré-processamento voltam a ser realizadas após a binarização da imagem.

Com a imagem em escala de cinza, a próxima etapa é a segmentação da imagem. A primeira atividade desta etapa é a realização da limiarização onde a imagem é binarizada. Este processo converte a imagem atual em uma nova imagem, em que cada pixel deve conter apenas as cores preto e branco. O nível de cinza de cada um dos pixels analisados é o fator

determinante para a definição de tal cor. Esta atividade é essencial para o processo de identificação visto que as cores da imagem não são relevantes na classificação do indivíduo, mas sim a forma das listras localizadas na parte inferior da cabeça dos cágados.

Após a binarização da imagem são aplicadas operações morfológicas a fim de remover possíveis ruídos, sendo estas operações constituintes da etapa de pré-processamento, conforme comentado anteriormente. Neste processo, são aplicados duas vezes a operação de erosão seguida da operação de dilatação sobre a imagem. A dupla aplicação destas operações se mostrou muito eficiente na remoção de ruídos das imagens dos cágados.

Encerrada a etapa de pré-processamento, o protótipo retorna a etapa de segmentação, em que é feita a identificação das componentes conexas da imagem. Esta identificação é realizada por meio da extração do contorno dos objetos. Posteriormente, é feita a seleção da região de interesse, ou seja, da componente conexa referente a listra em formato de ferradura. A seleção é realizada através da análise das características geométricas extraídas das componentes.

A próxima e última atividade da etapa de segmentação consiste em verificar se o indivíduo analisado possui uma listra circular próxima a ferradura. Caso esta verificação seja verdadeira, o *bounding box* da ferradura é calculado e a componente da listra circular é selecionada de forma semelhante a seleção da ferradura.

Em seguida, têm-se a etapa de caracterização da forma, em que, primeiramente, é realizado o cálculo dos descritores de Fourier da região da ferradura selecionada e da região da listra circular, caso exista. Para realização do cálculo são utilizadas as distâncias radiais somadas da *farthest point distance* de cada um dos pixels do contorno das componentes. Com os descritores calculados, a série de Fourier é obtida através dos descritores calculados. Esta série diz respeito ao identificador único gerada para o cágado.

Na última etapa do fluxo é realizada a classificação do cágado. Esta classificação é efetuada através das distâncias de Hamming, que são calculadas através da comparação do identificador único da imagem analisada com os identificadores das outras imagens cadastradas no banco de dados.

3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação. A seção 3.3.1 apresenta de forma detalhada as técnicas e ferramentas utilizadas. A seção 3.3.2 descreve a operacionalidade da implementação.

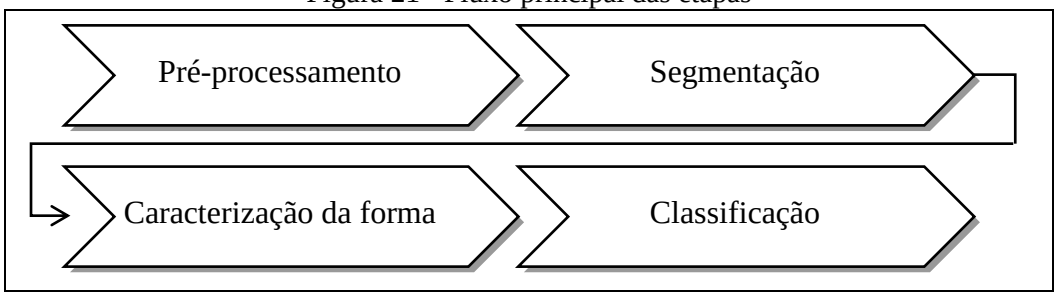
3.3.1 Técnicas e ferramentas utilizadas

O protótipo foi desenvolvido na linguagem C# (C-Sharp) na IDE Microsoft Visual Studio 2013. As tecnologias utilizadas no desenvolvimento estão listadas abaixo:

a) EmguCV: *Wrapper* da biblioteca OpenCV para a linguagem C#. É uma biblioteca de visão computacional que contém uma série de algoritmos de processamento de imagens. Neste trabalho, foram utilizados os algoritmos de transformação de imagens em escala de cinza, segmentação, operações morfológicas, extração de contorno entre outros;
b) Windows Forms: Plataforma da Microsoft baseada no Framework .Net, utilizada na criação de interfaces de aplicativos para a plataforma Windows. A interface do protótipo aqui apresentado foi desenvolvida fazendo uso desta plataforma junto do framework .NET 4.5;
c) SQLite: Biblioteca aberta escrita na linguagem C que implementa uma base de dados relacional. Interpreta a maioria das instruções SQL, não precisa ser instalada nem configurada. É um banco de dados leve e fica armazenado em um único arquivo no disco rígido do computador.

Nas próximas seções são detalhados os passos adotados na implementação do protótipo de identificação de cágados da espécie *Phrynops williamsi*. O detalhamento está dividido em etapas e possui exemplos do código fonte implementado, assim como, os resultados dessas implementações. As etapas do fluxo geral do protótipo são apresentadas na Figura 21.

Figura 21 - Fluxo principal das etapas



Conforme ilustrado na Figura 21, o protótipo possui as seguintes etapas: pré-processamento, segmentação, caracterização da forma e classificação. A seção a seguir descreve a etapa de pré-processamento.

3.3.1.1 Pré-Processamento

Na etapa de pré-processamento a imagem original é transformada em escala de cinza, é binarizada e filtrada através de operações morfológicas. As seções a seguir destinam-se a descrever as etapas do processo de pré-processamento.

3.3.1.1.1 Conversão da imagem em escala de cinza

A primeira etapa do algoritmo de pré-processamento é o carregamento da imagem em memória a partir do caminho informado pelo usuário. O protótipo utiliza a forma das regiões de interesse para fazer a identificação dos cágados, ou seja, a cor da imagem não é relevante. Portanto, é necessário remover as cores da imagem, mantendo apenas a sua iluminação, facilitando o seu armazenamento e processamento.

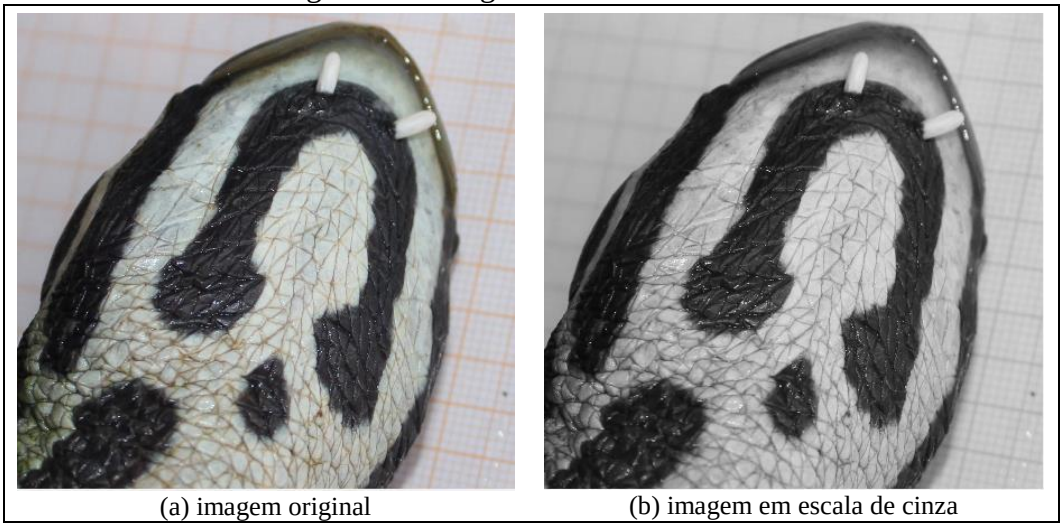
Uma nova imagem é gerada a partir da imagem em memória, tendo as cores convertidas para escala de cinza. Esta conversão é realizada através do método `CvtColor` fornecido pela biblioteca EmguCV, conforme mostra a linha 1 do Quadro 8.

Quadro 8- Código responsável pela conversão da imagem em escala de cinza

```
01.  CvInvoke.CvtColor(imgIn.ToMat(), imgDest,
          Emgu.CV.CvEnum.ColorConversion.Bgr2Gray);
02.  return imgDest;
```

A Figura 22 ilustra o resultado da execução do processo de conversão da imagem em escala de cinza. Na Figura 22a é demonstrada a imagem original e na Figura 22b é demonstrada a imagem em escala de cinza.

Figura 22 - Imagem em escala de cinza



(a) imagem original (b) imagem em escala de cinza

O próximo passo é a binarização da imagem. Esta atividade é realizada para segmentar o plano de fundo da imagem das regiões de interesse.

3.3.1.1.2 Binarização

A binarização é realizada através do método `Threshold`, disponibilizado pela biblioteca EmguCV. O processo de binarização consiste e transformar os pixels de escala de cinza em apenas duas cores, preto ou branco. Este processo consiste na análise de todos os pixels da imagem, e caso seu tom de cinza possua valor maior que o valor do limiar, este é transformado na cor branca, caso contrário, na cor preta. O valor do limiar (padrão 72) pode ser alterado pelo usuário a qualquer momento na janela de configurações do protótipo.

No Quadro 9 é demonstrado o código responsável pela binarização da imagem. O valor do limiar é aqui identificado pela variável `threshold`, a binarização é realizada através do método `Threshold` fornecido pela biblioteca EmguCV.

Quadro 9 - Quadro responsável pela binarização

```
01.  var imgDest = new Mat();
02.  CvInvoke.Threshold(imgIn, imgDest, threshold, 255,
          Emgu.CV.CvEnum.ThresholdType.BinaryInv);
```

A Figura 23 ilustra o processo de binarização da imagem em escala de cinza. A Figura 23a demonstra a imagem em escala de cinza e a Figura 23b demonstra a imagem binarizada.

Figura 23 - Imagem binarizada

(a) imagem em escala de cinza (b) imagem binarizada

Conforme pode ser visto na imagem Figura 23b, a imagem binarizada possui uma série de imperfeições e ruídos. Estas imperfeições/ruídos dificultam a identificação correta do cágado, portanto é necessário que estas sejam removidas. A correção destas imperfeições/ruídos é realizada através da aplicação de operações morfológicas de erosão e dilatação.

As seções a seguir descrevem detalhadamente a implementação destas operações. Nesta etapa são aplicadas duas vezes a o filtro de erosão seguido do filtro de dilatação, que fora a combinação que gerou melhores resultados dentre várias combinações testadas.

3.3.1.1.3 Erosão

O filtro de erosão consiste no afinamento das componentes eliminando pixels dispersos. O Quadro 10 apresenta o código responsável pela aplicação do operador morfológico de erosão. O método responsável pela operação de erosão recebe o parâmetro `level` que permite alterar o tamanho do elemento estruturante a ser utilizado. As linhas 3 e 7 são responsáveis pela obtenção do elemento estruturante de acordo com o parâmetro `level`, utilizando o método `GetStructuringElement` disponibilizado pela biblioteca EmguCV. A erosão da imagem é realizada através do método *Erode* na linha 9, também disponibilizado pela biblioteca EmguCV.

Quadro 10 - Código responsável pela operação de erosão

```
01.  if (level == 1)
02.  {
03.   se = CvInvoke.GetStructuringElement(
                                Emgu.CV.CvEnum.ElementShape.Rectangle,
                                new Size(3, 3), new Point(-1, -1));
04.  }
05.  else
06.  {
07.   se = CvInvoke.GetStructuringElement(
                                Emgu.CV.CvEnum.ElementShape.Rectangle,
                                new Size(12, 12), new Point(-1, -1));
08.  }

09.  CvInvoke.Erode(imgIn, imgDest, se, new Point(-1, -1), 1,
          Emgu.CV.CvEnum.BorderType.Default, new MCvScalar(1));
```

A Figura 24 ilustra o resultado do processo de remoção de ruídos através do filtro de erosão, onde é demonstrado na Figura 24a a imagem binarizada e na Figura 24b a imagem após a operação de erosão.

Figura 24 - Processo de erosão

(a) imagem binarizada (b) imagem após erosão morfológica

Após a remoção de pixels dispersos por meio da operação de erosão, é aplicado o filtro de dilatação. Esta operação é responsável pelo preenchimento de pequenos buracos dos objetos.

3.3.1.1.4 Dilatação

O filtro de dilatação morfológica consiste no engrossamento das componentes da imagem, eliminando eventuais lacunas das mesmas. O código responsável pela aplicação do operador morfológico de dilatação é mostrado no Quadro 11, semelhante ao método responsável pela operação de erosão, este método também recebe o parâmetro `level` que permite alterar o tamanho do elemento estruturante a ser utilizado. As linhas 3 e 7 são responsáveis pela obtenção do elemento estruturante de acordo com o parâmetro `level`, sendo que estes podem variar entre os tamanhos de 3 linhas por 3 colunas (3x3) ou de 12 linhas por 12 colunas (12x12). A operação de dilatação é realizada através do método `Dilate` (linha 9), disponibilizado pela biblioteca EmguCV.

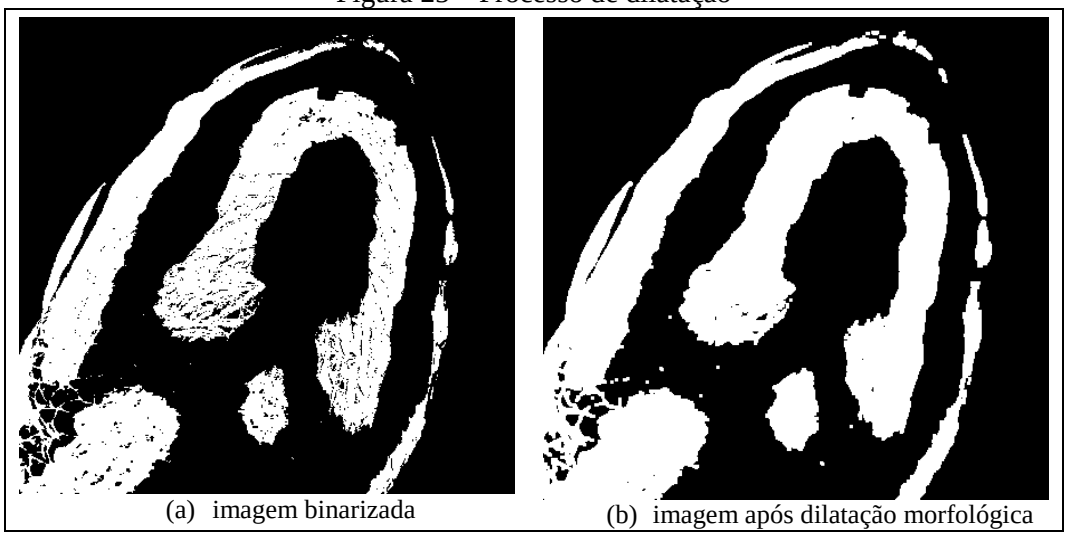
Quadro 11 - Código responsável pela operação de dilatação

```
01.  if (level == 1)
02.  {
03.   se = CvInvoke.GetStructuringElement(
                              Emgu.CV.CvEnum.ElementShape.Rectangle,
                              new Size(3, 3), new Point(-1, -1));
04.  }
05.  else
06.  {
07.   se = CvInvoke.GetStructuringElement(
                              Emgu.CV.CvEnum.ElementShape.Rectangle,
                              new Size(12, 12), new Point(-1, -1));
08.  }

09.  CvInvoke.Dilate(imgIn, imgDest, se, new Point(-1, -1), 1,
          Emgu.CV.CvEnum.BorderType.Default, new MCvScalar(1));
```

A Figura 25 ilustra o resultado do processo de preenchimento de lacunas através das operações de dilatação, onde é demonstrado na Figura 25a a imagem binarizada e na Figura 25b a imagem após a operação de dilatação.

Figura 25 – Processo de dilatação



(a) imagem binarizada (b) imagem após dilatação morfológica

O último passo da etapa de pré-processamento é a aplicação dos filtros de erosão e dilatação aplicados de forma conjunta, tendo como objetivo remover o máximo de ruídos da imagem.

3.3.1.1.5 Filtros morfológicos

A combinação de operações morfológicas para redução de ruídos que apresentou os melhores resultados utilizados no protótipo são mostrados abaixo:

a) aplicação do filtro de erosão com um EE de tamanho 3x3;
b) aplicação do filtro de dilatação com um EE de tamanho 12x12;
c) aplicação do filtro de erosão com um EE de tamanho 12x12;
d) aplicação do filtro de dilatação com um EE de tamanho 3x3.

Esta combinação foi obtida através de vários testes realizados em imagens de diversas amostras. No Quadro 12 é demonstrado o código responsável pela aplicação destas operações na imagem já binarizada. Pode-se observar que o valor passado no segundo parâmetro dos métodos `Erode` e `Dilate` é alternado, variando o tamanho do elemento estruturante utilizado nas operações conforme demonstrado nas seções anteriores.

Quadro 12 - Código responsável pela aplicação de abertura

```
01.  var img = Erode(binarizedImg, 1);
02.  img = Dilate(img, 2);
03.  img = Erode(img, 2);
04.  img = Dilate(img, 1);
```

A Figura 26 ilustra o resultado do processo de remoção de ruídos através das operações morfológicas, onde é demonstrado na Figura 26a a imagem binarizada e na Figura 26b a imagem após a aplicação dos operadores morfológicos.

Figura 26 – Processo de aplicação dos operadores morfológicos

(a) imagem binarizada

(b) imagem após aplicação dos operadores morfológicos

Com a imagem pré-processada inicia-se a etapa de segmentação. Nesta etapa o plano de fundo da imagem deve ser segmentado das regiões de interesse. A seção a seguir descreve tal etapa.

3.3.1.2 Segmentação

Nesta etapa é realizada a extração das componentes conexas para que posteriormente as regiões de interesse, listra em formato de ferradura e a listra circular, sejam selecionadas. A seção a seguir descreve a primeira etapa da segmentação, a extração das componentes conexas.

3.3.1.2.1 Extração das componentes conexas

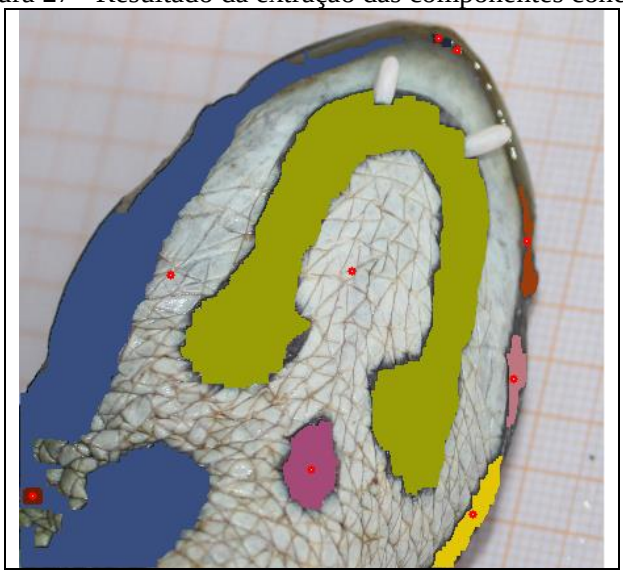
A extração das componentes conexas deve separar as regiões de interesse do plano de fundo da imagem. As componentes conexas são obtidas por meio da extração do contorno das formas. O código responsável pela extração das componentes conexas pode ser visto no Quadro 13.

Quadro 13 - Código responsável pela extração das componentes conexas

```
01.  CvInvoke.FindContours(imgIn, contours, hierachy,
                Emgu.CV.CvEnum.RetrType.Ccomp,
                Emgu.CV.CvEnum.ChainApproxMethod.ChainApproxSimple);
```

A partir do Quadro 13 na linha 1, pode ser observado que o método `FindCountours` da biblioteca EmguCV retorna um *array* de pontos sobre o plano cartesiano através da variável `contours`, onde cada elemento deste *array* representa uma componente conexa. A Figura 27 ilustra o resultado da extração das componentes conexas.

Figura 27 - Resultado da extração das componentes conexas



Na Figura 27 as componentes conexas foram coloridas para facilitar a visualização. Nela, pode-se observar que o plano de fundo foi segmentado e será ignorado. A centroide de cada componente é representada pelo ponto vermelho. Com as componentes extraídas, é possível fazer a seleção da listra em formato de ferradura.

3.3.1.2.2 Seleção da listra em formato de ferradura

A seleção da listra em formato de ferradura é realizada a partir do cálculo da área, dispersão da forma e distância do centroide até o ponto do contorno mais distante deste. Nesta etapa pretende-se identificar qual componente conexa diz respeito à ferradura.

No Quadro 14 é demonstrado o código responsável pela seleção. O primeiro passo da seleção é a obtenção da área de todas as componentes, cálculo realizado na linha 3. A lista contendo as áreas é então ordenada (linha 5) de forma descendente e apenas as três maiores componentes em relação a área são mantidas na lista (linha 8).

Quadro 14 - Código responsável pela seleção da ferradura

```
01.  for (int i = 0; i < contourVector.Size; i++)
02.  {
03.   areaList.Add(new KeyValuePair<int, double>(i,
                              (GeometryUtils.GetArea(contourVector[i]))));
04.  }

05.  var ordenedArea = areaList.OrderByDescending(x => x.Value).ToList();

06.  if(ordenedArea.Count > 3)
07.  {
08.   ordenedArea = ordenedArea.Take(3).ToList();
09.  }

10.  for (int i = 0; i < ordenedArea.Count; i++)
11.  {
12.   var kvp = ordenedArea[i];
13.   ordenedArea[i] =
               new KeyValuePair<int, double>(kvp.Key,
                              kvp.Value / GeometryUtils.GetMaxDistCentroid(
                                             (contourVector[kvp.Key])));
14.  }

15.  ordenedArea = ordenedArea.OrderByDescending(x => x.Value).ToList();

16.  if((ordenedArea[0].Value - ordenedArea[1].Value)
          < ordenedArea[1].Value / 2)
17.  {
18.   var reducedList = ordenedArea.Take(2).ToList();
19.   var dispertionList = new List<KeyValuePair<int, double>>
                                                 (reducedList.Count());

20.   for (int i = 0; i < reducedList.Count; i++)
21.   {
22.   var index = reducedList[i].Key;
23.   var dispertion = GeometryUtils.GetDispersion(contourVector[index]);
24.   dispertionList.Add(new KeyValuePair<int, double>(index, dispertion +
              GeometryUtils.GetArea(contourVector[index])));
25.   }

26.   var ordenedDisp = dispertionList.OrderBy(x => x.Value).ToList();
27.   result = contourVector[ordenedDisp[0].Key];
28.  }
29.  else
30.  {
31.   result = contourVector[ordenedArea[0].Key];
32.  }
```

Após encontrar as três maiores componentes, na linha 13 os valores da lista são atualizados com o quociente da área de cada componente pela distância do centroide até o ponto do contorno mais distante em relação a este. Este cálculo é realizado em decorrência de dois padrões encontrados nas imagens:

a) a componente da ferradura tende a ser a segunda maior componente em relação à área;

b) a distância do centroide da ferradura até o ponto do contorno mais distante do seu centroide tende a ser menor que a distância do centroide da maior componente até o ponto do contorno mais distante do seu centroide.

Analisando os resultados do cálculo realizado na linha 13, levando em consideração os dois padrões descritos acima, é possível afirmar que o valor referente a componente da ferradura será maior que o valor da maior componente. Portanto, é verificado se a diferença do valor do primeiro item pelo valor do segundo item da lista é superior à 50% do valor do segundo item. Caso esta condição seja verdadeira, é assumido que o primeiro item da lista diz respeito a listra em formato de ferradura (linha 31), caso contrário mais algumas verificações são necessárias para confirmar qual é a componente da ferradura, conforme demonstrado na sequência.

Caso a condição acima não seja atendida, é utilizada a dispersão das componentes para identificar a ferradura, visto que as duas maiores componentes devem possuir uma área muito semelhante. Primeiramente, a terceira componente conexa da lista é removida na linha 18. Após a remoção, uma nova lista é criada contendo a soma da dispersão da componente com a sua área (linha 24). Na sequência é feita a ordenação de modo ascendente da lista existente na linha 28. Por fim, é assumido que a componente da ferradura será o elemento que possuir o menor valor, ou seja, o primeiro item da lista (linha 27).

A última etapa do processo de segmentação, realizado caso o cágado analisado possua a listra circular próxima à ferradura, é o cálculo do *bounding box* da ferradura e a seleção da componente da listra circular.

3.3.1.2.3 Seleção da listra em formato circular

O Quadro 15 demonstra o código responsável pela seleção da componente referente à listra circular. Neste algoritmo são calculadas várias propriedades das formas, onde o perímetro é armazenado na variável `perimeter`, a área na variável `area` e a circularidade na variável `roundness`. O centroide da ferradura é identificado por `hsCentroid` e seu perímetro por `hsPerimeter`.

Quadro 15 - Código responsável pela seleção da listra circular

```
01.  for (int i = 0; i < contourVector.Size; i++)
02.  {
03.   // cálculo das propriedades da forma

04.   if (perimeter < (hsPerimeter / 2) &&
                 area > hsArea * 0.02f && roundness > 0.45f)
05.   {
06.   centroid = GeometryUtils.GetCentroid(contourVector[i]);

07.   bool insideBB =
                      GeometryUtils.VerificarComponenteDentroBoundingBox(
                          img.GetBoundingBoxFerradura(),
                          contourVector[i]);

08.   double centroidDist = GeometryUtils.GetEuclideanDistance(
                                                    centroid, hsCentroid);
09.   if(insideBB)
10.   {  centroidDist -= centroidDist * 0.3f; }

11.   centroidDistList.Add(
                  new KeyValuePair<int, double>(i, centroidDist));
12.   }
13.  }

14.  var ordenedDistList = centroidDistList.OrderBy(x => x.Value).ToList();

15.  if (ordenedDistList.Count > 0)
16.  {
17.   // cálculo das propriedades da componente

18.   bool insideBB = GeometryUtils.CheckComponentInsideBB(
                                   img.GetBoundingBoxFerradura(),
                                   contourVector[circularComponent.Key]);

19.   if (disp < 8 && circularComponent.Value < hsMaxDist * 1.5f &&
                  (GeometryUtils.GetMinDistCentroide(
                  contourVector[circularComponent.Key]) > 5 || insideBB))
20.   {
21.   result = contourVector[circularComponent.Key];
22.   }
23.  }
```

Na linha 4 é verificado se a componente analisada atende as três condições iniciais para ser a listra circular. São elas:

a) possuir perímetro menor que 50% do perímetro da ferradura;

b) possuir área maior que 2% da área da ferradura;

c) possuir grau de circularidade maior que 0,45.

Caso estas três condições sejam atendidas, é assumido que a componente está apta a ser a listra circular. As próximas validações realizadas para determinar se a componente é a mancha circular, fazem uso do centroide da componente calculada na linha 6.

O próximo passo da validação é verificar se pelo menos um dos pontos do contorno da componente analisada estão contidos dentro do *bounding box* da ferradura (linhas 8 e 9). Caso

esta condição seja atendida, maior é a probabilidade desta componente ser a listra circular. Portanto, a distância do centroide da ferradura até o centroide da componente analisada (`centroidDist`), é reduzida em 30% do seu valor, conforme mostra a linha 10. Esta redução é realizada pois, quanto menor o valor da *centroidDist* maior a probabilidade de a componente ser a listra circular.

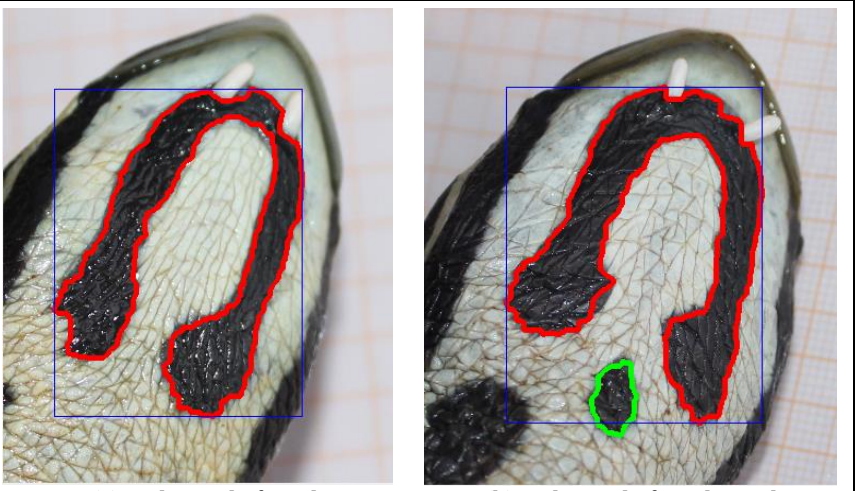
A lista das distâncias do centroide da ferradura até os centroides das componentes é ordenada ascendentemente na linha 14. Se a lista possuir pelo menos um item (linha 15) é assumido que o primeiro item diz respeito à listra circular.

A partir da linha 17, são feitas mais algumas verificações para validar se o primeiro item da lista é de fato a listra circular. Neste trecho de código, `hsMaxDist` armazena a distância do centroide até o ponto do contorno mais distante, `circularComponent` a componente analisada, `roundness` a circularidade da componente e `disp` a dispersão da componente. A componente será considerada a listra circular caso as três condições abaixo sejam atendidas:

a) deve possuir dispersão menor que oito;
b) a distância do seu centroide para o centroide da ferradura deve ser menor que 150% da distância entre do centroide da ferradura até o ponto do contorno mais distante do mesmo;
c) a distância entre o centroide da componente analisada e o ponto do contorno mais próximo deve ser maior que cinco ou, pelo menos um dos pontos da componente analisada deve estar contido dentro do *bounding box* da ferradura (linha 18).

Caso estas três condições sejam atendidas, é assumido que a componente analisada é a listra circular. A realização da validação destas três condições pode ser observada na linha 19. A Figura 28 ilustra o resultado do algoritmo de seleção das componentes. A Figura 28a ilustra a seleção da ferradura em um cágado que não possui a listra circular. Já a Figura 28b, ilustra a seleção da ferradura na cor vermelha e da listra circular na cor verde. Nas duas ilustrações o *bounding box* da ferradura pode ser visto na cor azul.

Figura 28 - Seleção da ferradura e da listra circular



(a) seleção da ferradura (b) seleção da ferradura e listra circular

Com as regiões de interesse selecionadas, o próximo passo é a caracterização das formas através dos descritores de Fourier. O cálculo destes descritores é demonstrado na seção a seguir.

3.3.1.3 Caracterização da forma

Nesta etapa os descritores de forma da listra em formato de ferradura e da listra circular são calculados. A partir destes descritores, obtém-se a série de Fourier. Ainda nesta seção é descrito o passo a passo para obtenção do *bounding box* da listra, do centroide e dos índices de dispersão e circularidade. Estas características são utilizadas no cálculo dos descritores e na seleção das regiões de interesse.

3.3.1.3.1 Bounding box

A primeira característica extraída da figura o *bounding box* da listra em formato de ferradura. O *bounding box* diz respeito a menor região limitante que abranja todos os pixels do objeto. Esta região geralmente possui o formato retangular. O Quadro 16 demonstra o código fonte responsável pelo cálculo do *bounding box*.

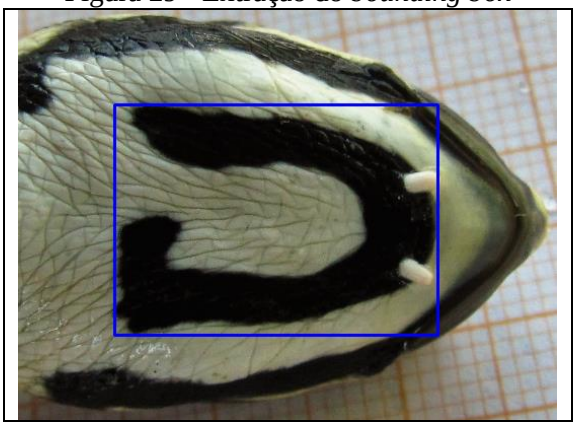
Quadro 16 - Código responsável pela extração do *bounding box*

```
01.  this._horseShoeBoundingBox = new BoundingBox();
02.  this._horseShoeBoundingBox.Max_X = horseshoe.Select(x => x.X).Max();
03.  this._horseShoeBoundingBox.Max_Y = horseshoe.Select(x => x.Y).Max();
04.  this._horseShoeBoundingBox.Min_X = horseshoe.Select(x => x.X).Min();
05.  this._horseShoeBoundingBox.Min_Y = horseshoe.Select(x => x.Y).Min();
```

Neste algoritmo, `horseshoe` armazena o `array` de pontos do contorno da listra em formato de ferradura utilizado na extração. Pode-se observar que entre as linhas 1 e 5 são obtidos o menor e maior valor das coordenadas `x` e `y` do `array` de pontos. Estes quatro valores

geram as quatro coordenadas do *bounding box*. A Figura 29 demonstra um exemplo do *bounding box* extraído da ferradura.

Figura 29 - Extração do *bounding box*



Pode-se observar que o *bounding box* da ferradura em azul é a menor região limitante que abrange todos os pixels da listra. A próxima característica extraída é o centroide, que representa o ponto central do objeto.

3.3.1.3.2 Centroide

O centroide, ou centro da massa, é obtido através do cálculo da média do somatório das coordenadas do contorno do objeto. Esta característica é utilizada na seleção da listra de formato circular, no cálculo da área, no cálculo dos descritores de Fourier e no cálculo da dispersão. A fórmula utilizada no cálculo desta característica é demonstrada no Quadro 17.

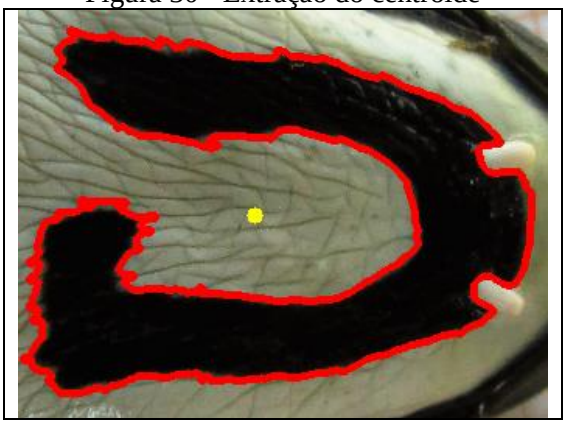
Quadro 17 - Fórmula para cálculo do centroide

$$M = \frac{\sum_{n=0}^{N-1} u(n)}{N}$$

Fonte: Costa e Cesar Jr. (2009, p. 426).

Nessa formula, `u(n)` representa a coordenada do ponto `n` do contorno do objeto, `N` o número de pontos do contorno do mesmo e `M` a coordenada do centroide deste. A Figura 30 ilustra um exemplo da extração do centroide da listra em formato de ferradura.

Figura 30 - Extração do centroide



Na ilustração, o centroide da ferradura é ilustrado pelo ponto amarelo localizado no centro da mesma. Outra característica calculada pelo protótipo é a dispersão, detalhada na seção a seguir. Esta característica é utilizada na seleção da listra e formato de ferradura.

3.3.1.3.3 Dispersão

A dispersão, ou irregularidade, é o índice que indica o nível de irregularidade da figura. Quanto maior este índice, maior é a diferença entre a maior e a menor distância do centroide até a borda da figura. A fórmula utilizada no cálculo da dispersão é demonstrada no Quadro 18.

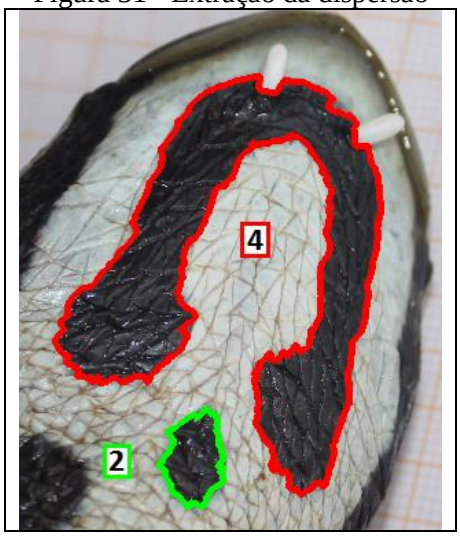
Quadro 18 - Fórmula para cálculo da dispersão

$$dispersão = \frac{max\left(\sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2}\right)}{min\left(\sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2}\right)}$$

Fonte: adaptado de Nixon e Aguado (2002, p. 280).

Nessa formula, $(\bar{x}, \bar{y})$ representa as coordenadas do centroide da figura. A Figura 31 ilustra um exemplo do índice da dispersão sobre dois objetos. Nesta ilustração, os índices são exibidos ao lado de seus respectivos objetos.

Figura 31 - Extração da dispersão



Na Figura 31, pode-se observar visualmente que o objeto em vermelho é mais irregular que o objeto em verde, ou seja, a diferença das distâncias entre os dois pontos mais distantes e os dois pontos mais próximos é maior. A próxima característica a ser obtida é a circularidade, descrita na seção a seguir. Esta característica é utilizada na obtenção da listra circular conforme descrito nas seções anteriores.

3.3.1.3.4 Circularidade

A circularidade é o coeficiente que indica o quão circular é um objeto. Este índice aumenta conforme o formato do mesmo se assemelha a um círculo. A fórmula utilizada na obtenção da circularidade é apresentada no Quadro 19.

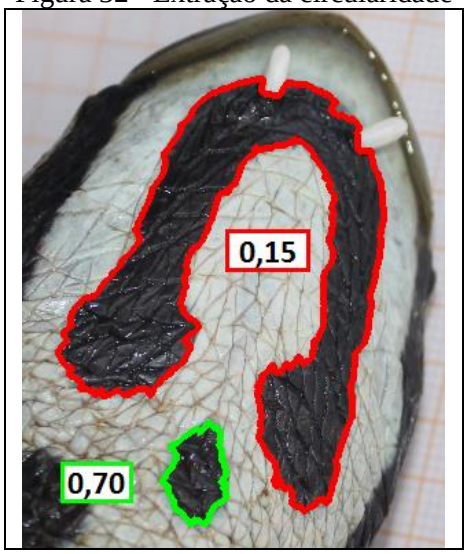
Quadro 19 - Fórmula para cálculo da circularidade

$$circularidade = \frac{4\pi A}{p^2}$$

Fonte: traduzido de Kadir et al. (2011, p.226).

Nessa formula, `A` representa a área da figura enquanto `p` diz respeito ao perímetro da mesma. A Figura 32 ilustra o índice da circularidade obtido da listra em formato de ferradura e da listra circular de um cágado.

Figura 32 - Extração da circularidade



Pode-se observar visualmente na Figura 32, que a listra em verde se assemelha mais à um círculo do que a listra em vermelho, justificando a diferença dos valores dos coeficientes exibidos ao lado de seus respectivos objetos. A próxima característica a ser calculada são os descritores de Fourier. Os descritores de Fourier são utilizados na representação do contorno dos objetos.

3.3.1.3.5 Descritores de Fourier

Para realização do cálculo dos descritores de Fourier, são obtidas a distância radial e o *farthest point distance* dos pontos do contorno analisado (ver seção 2.2.1). A partir destes valores, os descritores são obtidos através da transformada discreta de Fourier. No Quadro 20 é demonstrado o código responsável pelo cálculo dos descritores de Fourier.

Quadro 20 - Cálculo dos descritores de Fourier

```
01.  for (int n = 0; n < N; n++) // for each descriptor
02.  {
03.   for (int t = 0; t < T; t++) // for each contour point
04.   {
05.   double theta = (2 * Math.PI * n * t) / T; // Theta

06.   double radialDistance =
                            GetEuclideanDistance(centroid, imgContour[t]);

07.   var farthestPoint =
                            GetFarthestPoint(imgContour, imgContour[t]);
08.   var farthestPointDistance =
                            GetEuclideanDistance(centroid, farthestPoint);

09.   radialDistance += farthestPointDistance;

10.   fourierR[n] += radialDistance * Math.Cos(theta);
11.   fourierI[n] += radialDistance * -Math.Sin(theta);
12.   }

13.   fourierR[n] = fourierR[n] / T;
14.   fourierI[n] = fourierI[n] / T;
15.  }
```

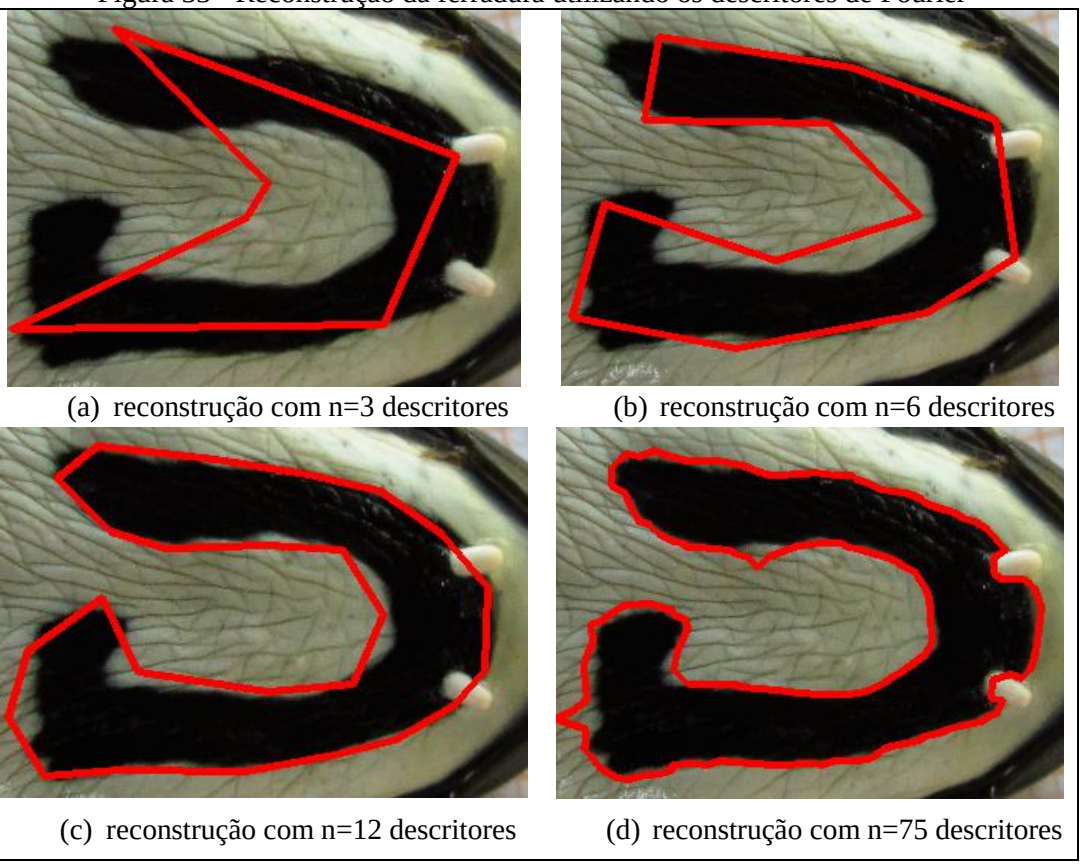
Neste algoritmo, `N` diz respeito a quantidade de descritores, `T` a quantidade de pontos do contorno, `n` identifica o descritor calculado e `t` o ponto do contorno analisado. O cálculo do valor de `tetha`, equivalente à `2πnt/T` pode ser visto na linha 5. Na linha 6 é calculada a distância radial do ponto analisado. Na linha 7 é localizado o ponto mais distante em relação ao ponto analisado para que na linha 8 seja calculada a *farthest point distance*, ou seja, a distância do ponto analisado até o centroide do objeto, somado da distância do centroide até ponto mais distante em relação a sua referência.

Com os valores de `tetha` e das distâncias calculados, são realizados os somatórios da parte real e da parte imaginária do descritor e armazenadas, respectivamente, nas variáveis `fourierR` e `fourierI`. Nas linhas 13 e 14 é realizada a divisão dos valores dos descritores pela quantidade de pontos do contorno.

Conforme comentado anteriormente, poucos descritores são suficientes para representar a forma do objeto. Portanto, após a realização do cálculo dos descritores, são mantidos apenas os `n` primeiros e `n` últimos coeficientes resultando em `2n` coeficientes. Para a listra em formato de ferradura são mantidos vinte e três (23) descritores e seis (6) descritores da listra circular. Estes valores podem ser alterados pelo usuário na tela de configurações do protótipo.

A Figura 33 apresenta um exemplo da reconstrução da listra em formato de ferradura utilizando conjuntos de descritores de diferentes tamanhos.

Figura 33 - Reconstrução da ferradura utilizando os descritores de Fourier



(a) reconstrução com n=3 descritores (b) reconstrução com n=6 descritores

(c) reconstrução com n=12 descritores (d) reconstrução com n=75 descritores

Na Figura 33a, pode-se observar que a reconstrução do objeto utilizando n=3 descritores geram apenas uma caricatura grosseira da ferradura. Por outro lado, com n=6 (Figura 33b) descritores é possível reconstruir o objeto com sua estrutura principal. Com n=12 (Figura 33c), detalhes mais finos da forma são reconstruídos. Finalmente, a reconstrução com n=75 (Figura 33d) descritores provê uma reprodução quase exata da forma original.

Com os descritores calculados, o próximo passo é a obtenção da série de Fourier utilizada na classificação do indivíduo.

3.3.1.3.6 Série de Fourier

A série de Fourier é obtida através da soma da parte real com a parte imaginária dos descritores. O Quadro 21 mostra o código responsável pelo cálculo da série de Fourier.

Quadro 21 - Cálculo da série de Fourier

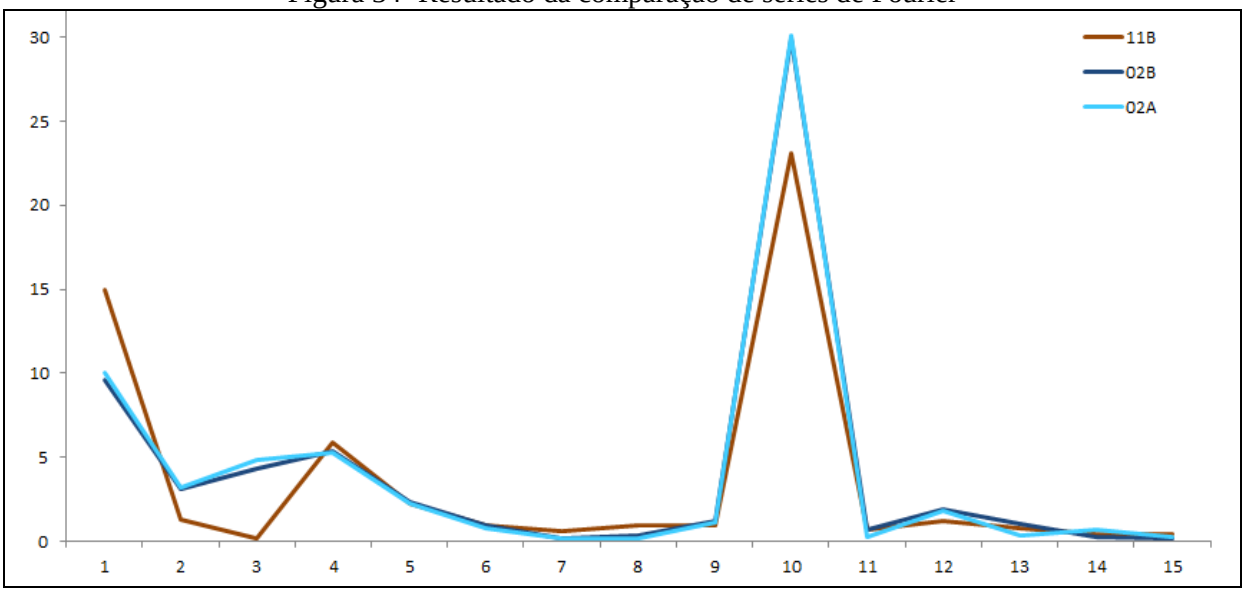
```
01.  for (int u = 0; u < series.Length; u++)
02.  {
03.   series[u] = Math.Abs(
                        Math.Sqrt(Math.Pow(descriptorsArray[0][u], 2) +
                        Math.Pow(descriptorsArray[1][u], 2)));
04.   if(u > 1)
05.   {
06.   series[u] /= series[0];
07.   }
08.  }
```

A partir do Quadro 21, pode ser visto, na linha 3, que são mantidas apenas as magnitudes dos descritores, com o objetivo de tornar os descritores invariantes à rotação. A invariância à escala é obtida através da divisão dos descritores pelo valor do primeiro descritor (linha 6). A invariância á translação não precisa ser tratada, pois trabalha-se com as distâncias radiais dos pontos, ou seja, todas as distâncias são calculadas em relação ao centroide do objeto, não tendo relação com a posição do mesmo na imagem.

3.3.1.4 Classificação

Nesta etapa é feita a identificação dos cágados. No momento em que o usuário carrega uma imagem, a série de Fourier deste cágado é calculada. Esta série diz respeito ao identificador único gerado para o cágado, e é comparada com os identificadores de outros cágados já cadastrados no banco de dados a fim de identificá-lo. A Figura 34 ilustra em um gráfico a série de Fourier.

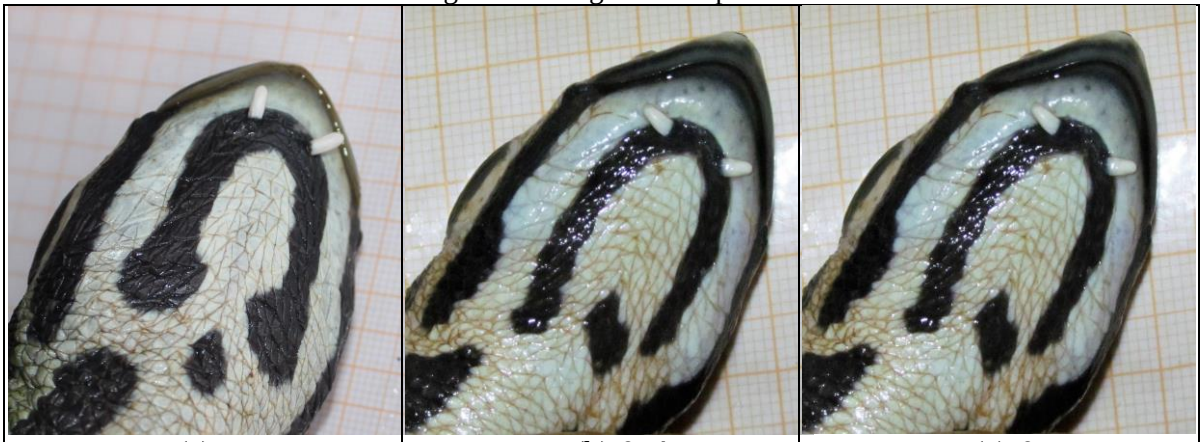
Figura 34- Resultado da comparação de séries de Fourier



A Figura 34 ilustra a comparação do identificador de três imagens distintas, sendo que duas delas são do mesmo cágado. Pode-se observar que as linhas azul claro e azul escuro, identificadores únicos gerados a partir de imagens distintas do mesmo cágado, são muito semelhantes em todos os seus valores. Enquanto a linha em vermelho, que representa o identificador único da imagem de outro cágado, se difere bastante das outras duas.

Como demonstrado na Figura 34, o identificador único gerado pelo protótipo é formado por um *array* de valores. Estes valores são a assinatura da forma, ou seja, variam de acordo com o formato da listra de ferradura, conforme demonstrado pela Figura 35.

Figura 35 - Cágados comparados



(a) 11B | (b) 02A | (c) 02B

A Figura 35 demonstra as imagens dos cágados que geraram os identificadores únicos mostrados na Figura 34. É visualmente possível perceber a diferença no formato das listras em formato de ferradura entre as Figuras 35a, 35b e 35c, justificando a diferença apresentada no gráfico da Figura 34.

A identificação dos cágados é feita através da distância de Hamming, que consiste na soma das diferenças de cada um dos itens das séries comparadas. Nesta soma estão sendo considerados apenas as diferenças que forem superiores a 10% do maior valor comparado. Caso o resultado dessa distância seja inferior a doze (padrão), este valor pode ser alterado pelo usuário na aba de configurações, os dois cágados comparados são considerados os mesmos indivíduos.

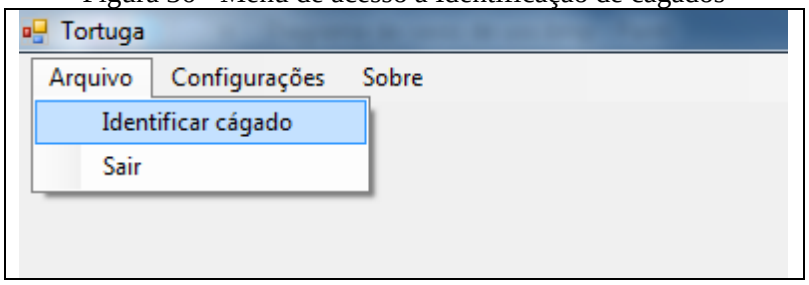
3.3.2 Operacionalidade da implementação

Visto que o projeto desenvolvido neste trabalho é um protótipo, o número de telas e funcionalidades disponibilizadas para o usuário é limitada. Neste, o usuário poderá fazer a identificação dos cágados através de uma imagem da parte inferior da sua cabeça. Outra funcionalidade disponibilizada é o registro dos cágados para posterior identificação dos mesmos. Também é permitido ao usuário alterar alguns parâmetros de configuração do protótipo. As seções a seguir demonstram as operacionalidades de tais funcionalidades.

3.3.2.1 Identificação do cágado

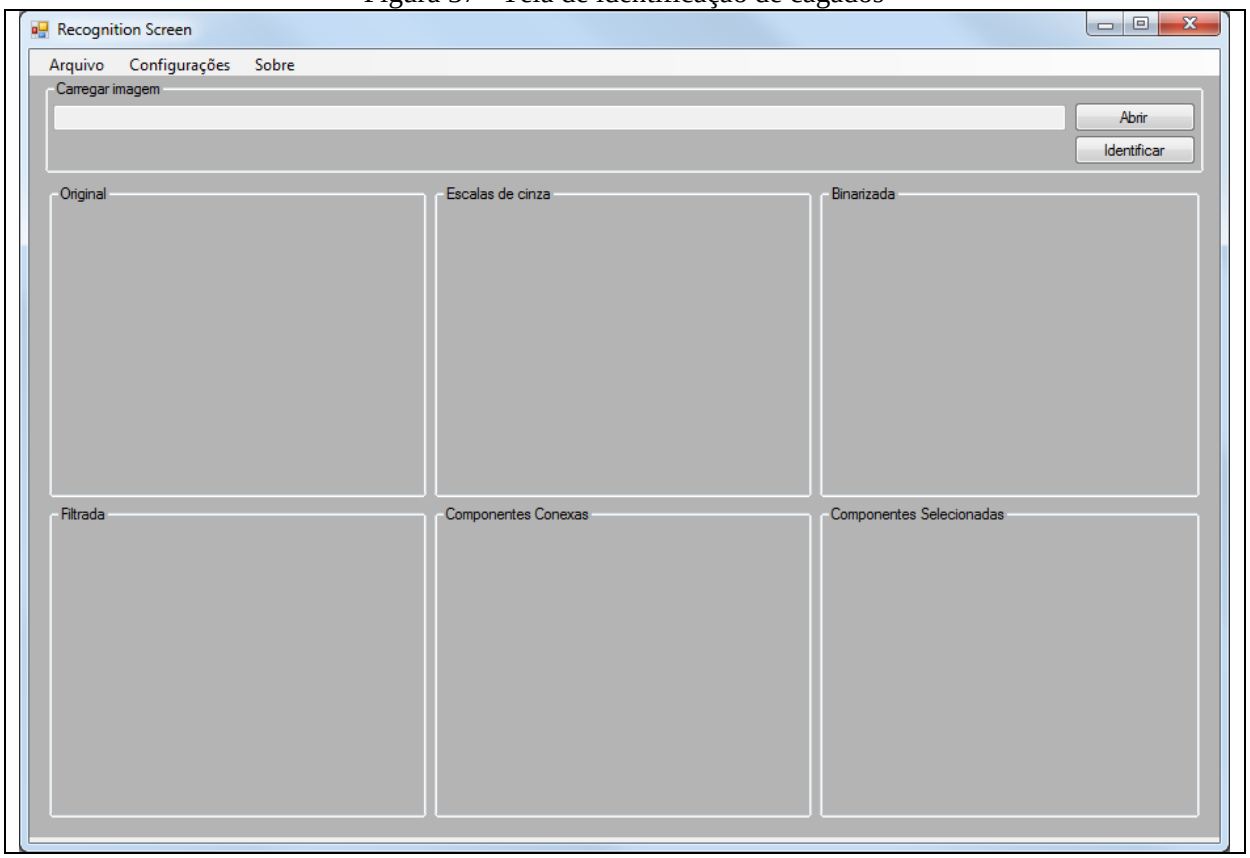
A identificação do cágado é principal funcionalidade do protótipo, e pode ser acessada a partir da opção `Identificar cágado` no menu principal que fica localizado na parte superior da tela, como demonstrado na Figura 36.

Figura 36 - Menu de acesso a Identificação de cágados



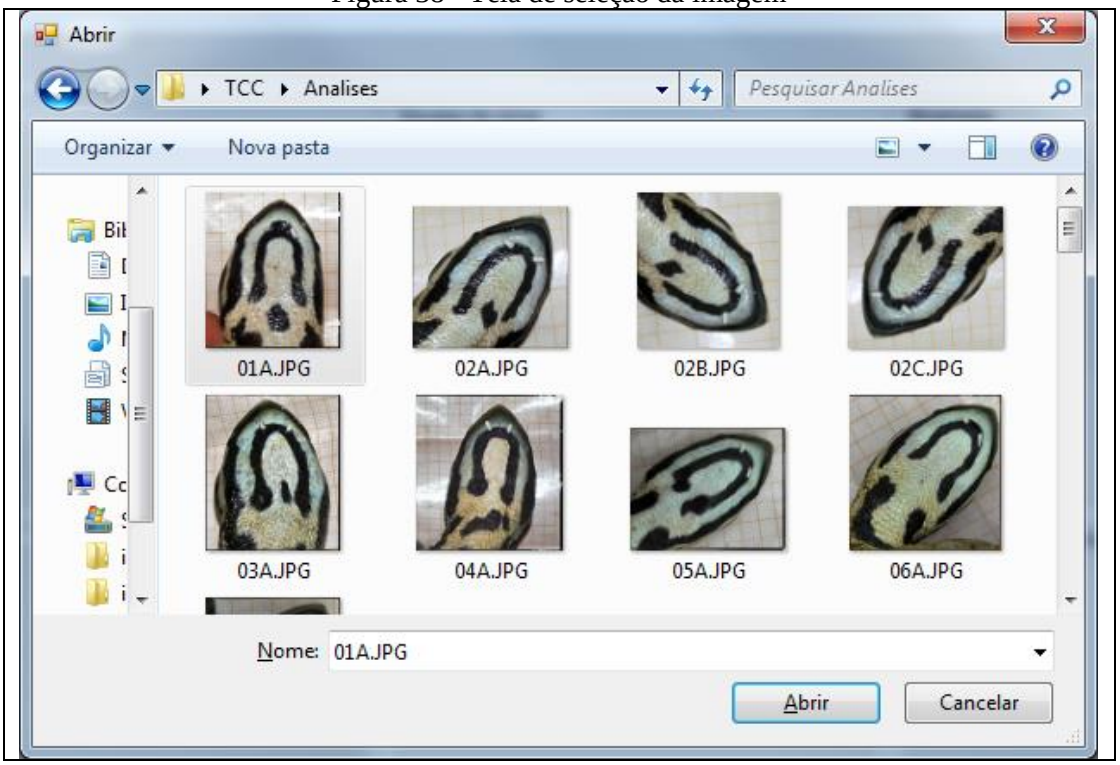
Após a seleção desta opção, o protótipo exibe a tela de identificação de cágados, conforme mostra a Figura 37.

Figura 37 - Tela de identificação de cágados



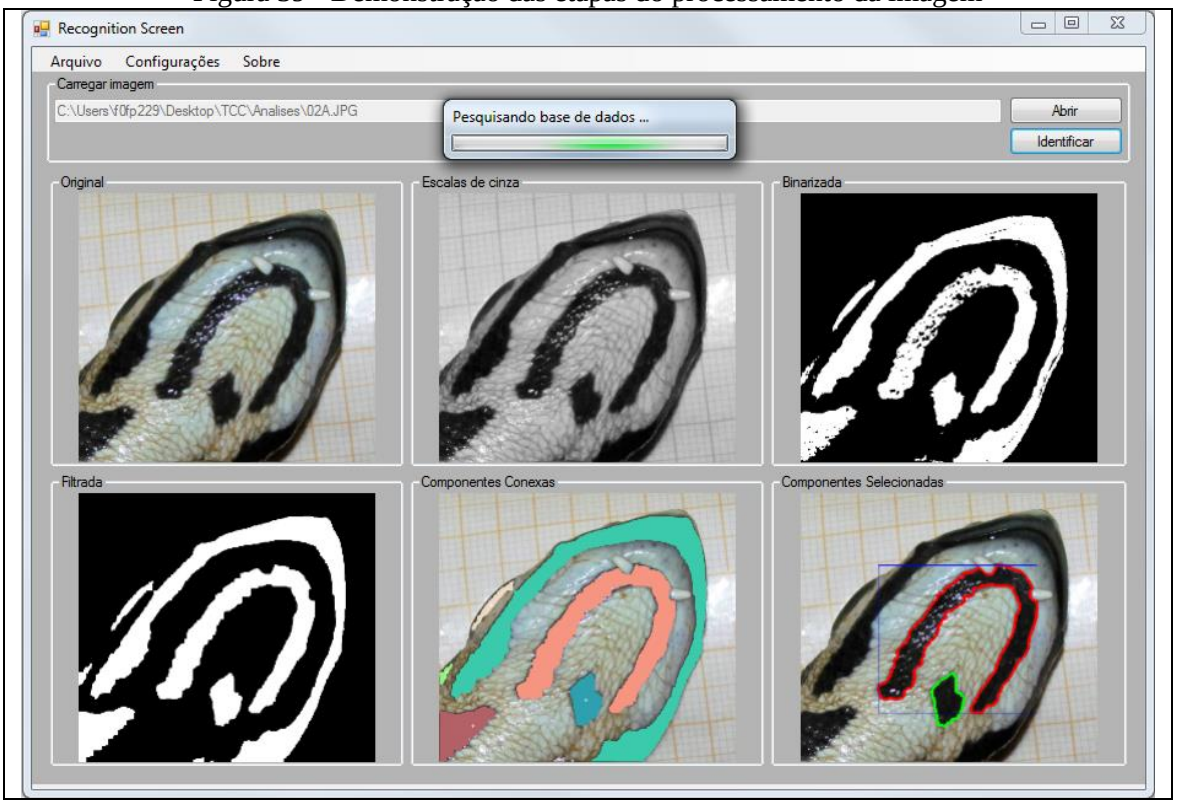
O primeiro passo a ser realizado nesta tela é o carregamento da imagem por meio do botão `Abrir`. No momento em que este botão é pressionado, é aberta uma nova tela do tipo *modal* que permite que o usuário selecione uma imagem armazenada no seu computador, como ilustrado na Figura 38.

Figura 38 - Tela de seleção da imagem



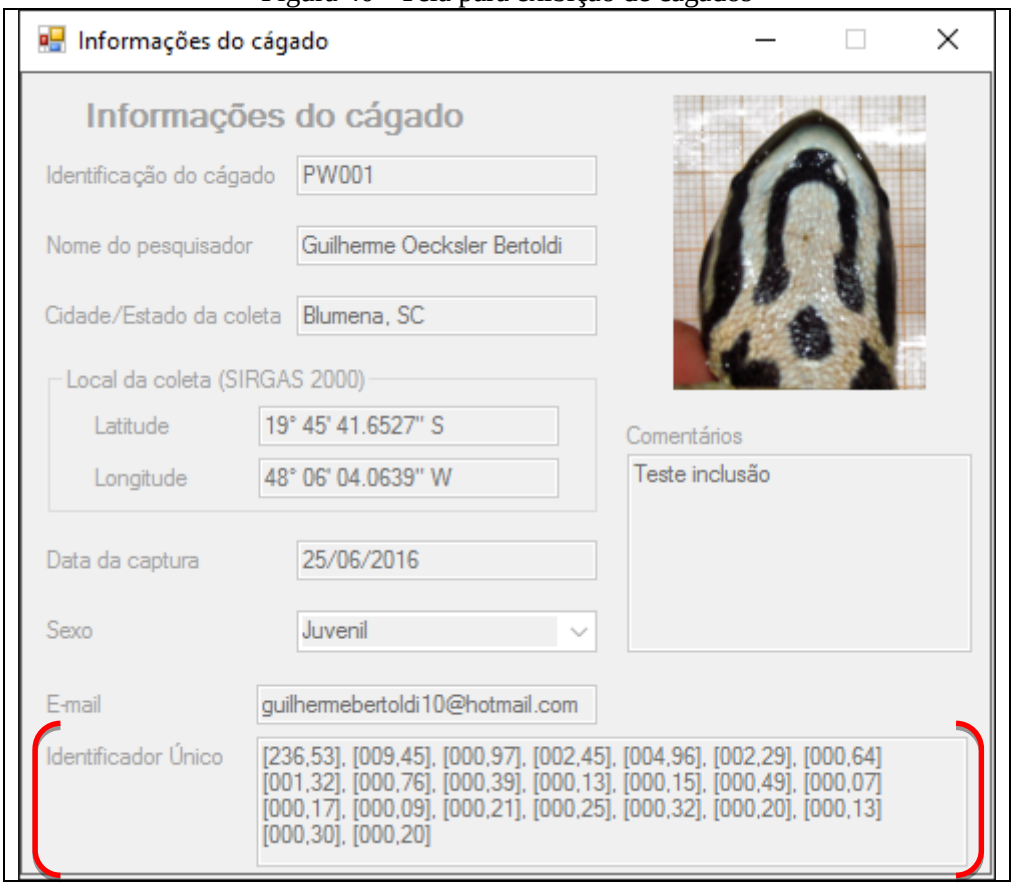
A imagem selecionada é carregada no campo `Imagem` da tela anterior. O próximo passo é a identificação do cágado, que é realizada por meio do botão `Identifica`. Ao clicar neste botão, a imagem selecionada será processada conforme descrito nas seções anteriores. Todas as etapas deste processamento são demostradas em tela, conforme pode ser visto na Figura 39.

Figura 39 - Demonstração das etapas do processamento da imagem



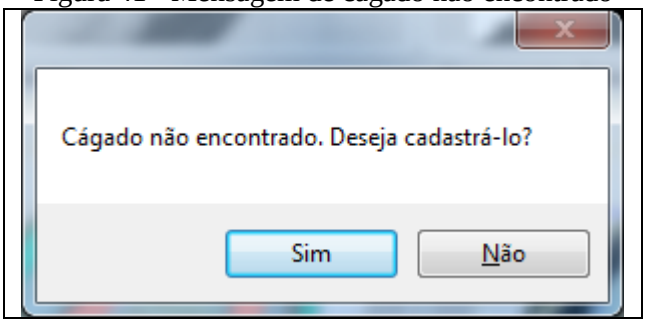
Após o processamento da imagem, o protótipo consulta a base de dados para verificar se o cágado da imagem processada já está cadastrado. Caso seja encontrado na base de dados um cágado semelhante, uma nova tela é exibida contendo as informações relativas ao indivíduo, conforme mostra a Figura 40.

Figura 40 - Tela para exibição de cágados



Em destaque é exibido o identificador único do cágado, gerado a partir do formato da sua listra. Caso o cágado ainda não esteja cadastrado no banco de dados, o sistema exibe uma mensagem informando o fato e questionando se usuário deseja realizar o cadastro do cágado, conforme ilustra a Figura 41.

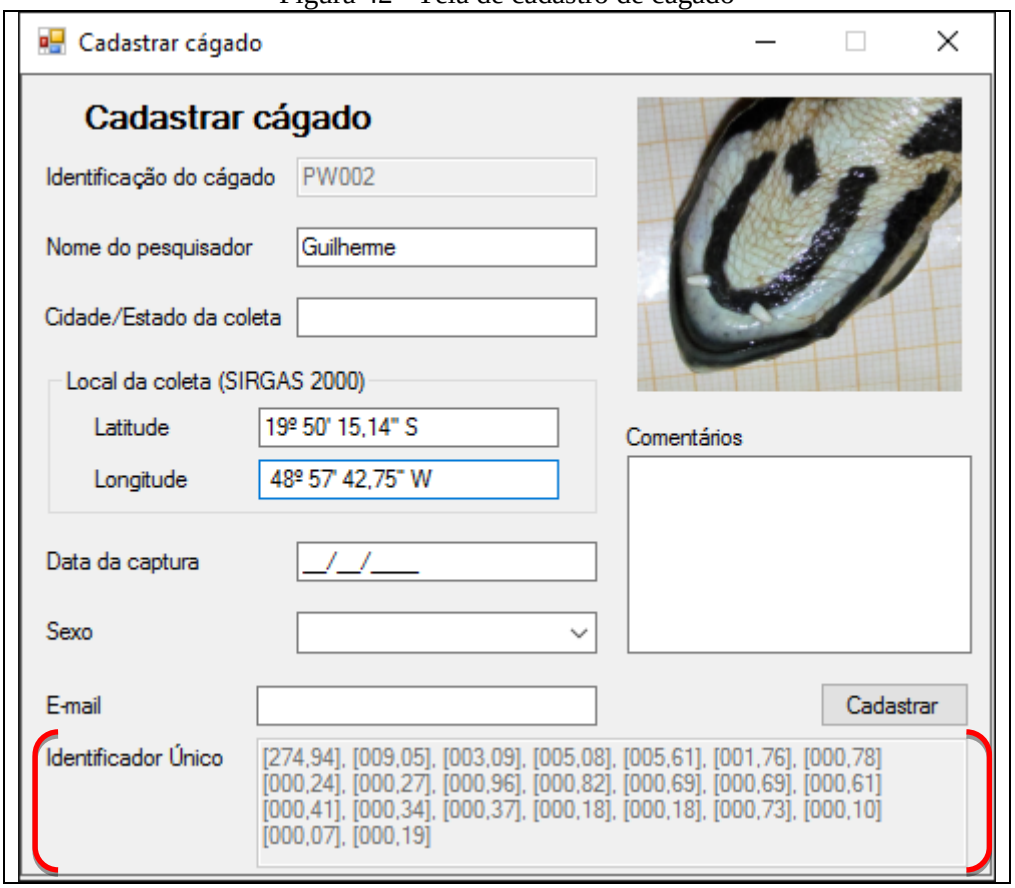
Figura 41 - Mensagem de cágado não encontrado



A partir desta mensagem, o usuário pode cancelar a operação clicando no botão `Não` ou cadastrar o cágado clicando no botão `Sim`. Confirmando a operação, o usuário será

redirecionado para a funcionalidade de cadastro do cágado. Nesta tela, o usuário poderá fazer a inserção de novos cágados no banco de dados para futuras comparações, conforme ilustra a Figura 42.

Figura 42 - Tela de cadastro de cágado



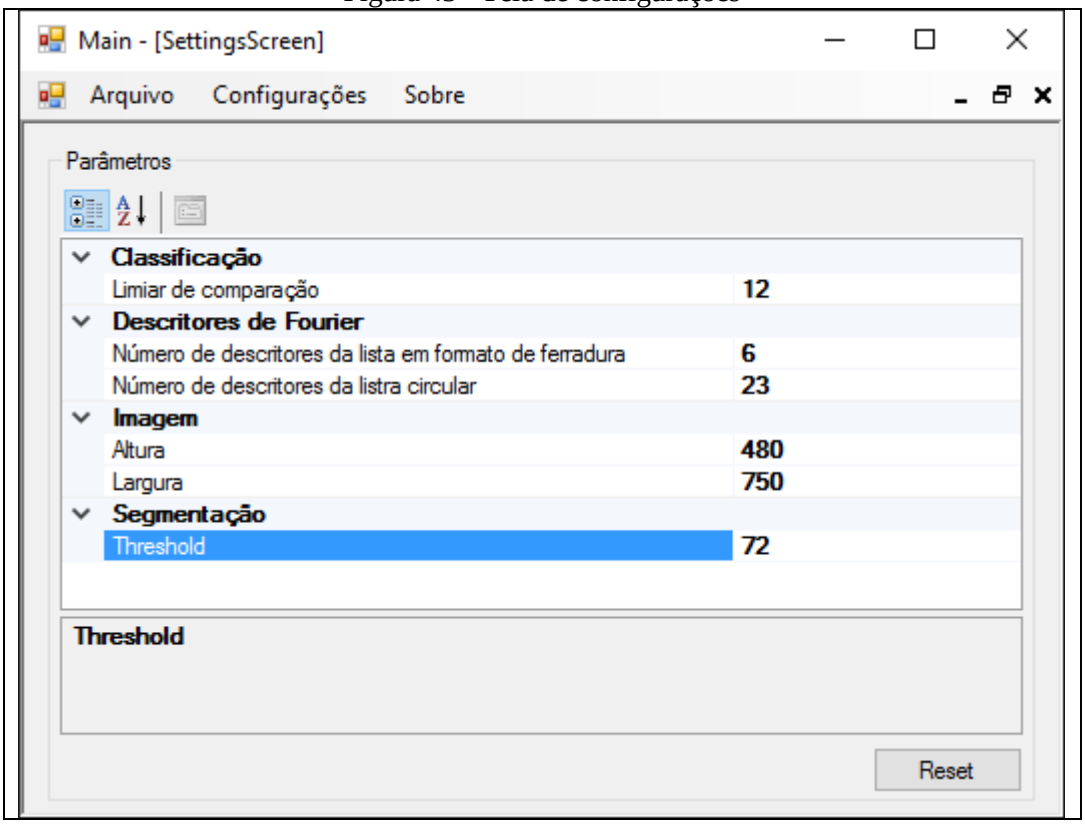
Ao ser acessar a tela de cadastro, os campos de `Identificação do Cágado` e `Imagem` são preenchidos automaticamente com as informações do processo de identificação (item em destaque na Figura 42). Os outros dados do cágado a serem incluídos no banco de dados como `Nome pesquisador`, `Cidade/Estado da coleta`, `Latitude`, `Longitude`, `Data da captura`, `Sexo (Masculino, Feminino, Juvenil, Indefinido)`, `E-mail` e `Comentários` devem ser informados pelo usuário.

Após o preenchimento de todas as informações, a inserção do cágado na base de dados pode ser efetivada através do botão `Cadastrar`. Uma mensagem de sucesso é apresentada no clique deste botão.

3.3.2.2 Alterar configurações

O usuário poderá alterar algumas configurações do protótipo. Esta tela pode ser acessada através da opção `Configurações`. Ao clicar nesta opção, é exibida a tela de configurações, conforme mostra a Figura 43.

Figura 43 - Tela de configurações



Os parâmetros mostrados em tela (limiares e números de descritores) são ajustados conforme o usuário vai alterando os seus valores. O botão `Reset` é responsável por restaurar os valores padrões utilizados inicialmente pelo protótipo.

3.4 RESULTADOS E DISCUSSÕES

Para validar a efetividade do protótipo, foram realizadas comparações utilizando a base de dados fornecida pelos biólogos André Santos e José Carlos Rocha. Esta base é composta por 53 imagens de 26 cágados diferentes. Estes números dizem respeito a base previamente tratada manualmente, onde foram mantidas apenas as imagens adequadas para a identificação. Sendo descartadas imagens de cágados que possuem listras conexas, imagens com falta de nitidez e/ou que apresentam alto nível de reflexo sobre as listras.

Na montagem manual da base, as imagens de cágados que possuem a listra em formato de ferradura conexa em outras listras são descartadas. Também são selecionadas apenas as regiões de interesse da imagem, ou seja, apenas a parte inferior da cabeça do cágado é mantida enquanto que o restante é descartado.

Com a base de dados devidamente tratada de forma manual, foram realizadas comparações intra-classe e inter-classe e o resultado destas podem observados na Tabela 1.

Nessa tabela são apresentados a quantidade de imagens de cada cágado, o número de comparações, o número de acertos para cada uma das duas modalidades de comparação e as taxas de acerto destas comparações.

Tabela 1 - Resultado das comparações intra-classe e inter-classe

Indivíduos	Nº amostras	Nº classificações intra-classe	Nº classificações intra-classes corretas	Percentual de acerto intra-classe	Nº classificações inter-classes	Nº classificações inter-classes corretas	Percentual de acerto inter-classe
PW01	1	-	-	-	52	50	96,15%
PW02	3	6	6	100,00%	147	147	100,00%
PW03	1	-	-	-	48	48	100,00%
PW04	1	-	-	-	47	31	65,96%
PW05	1	-	-	-	46	27	58,70%
PW06	1	-	-	-	45	45	100,00%
PW07	1	-	-	-	44	28	63,64%
PW08	1	-	-	-	43	43	100,00%
PW09	1	-	-	-	42	30	71,43%
PW10	4	12	10	83,33%	152	146	96,05%
PW11	3	6	6	100,00%	105	104	99,05%
PW12	6	30	28	93,33%	174	109	62,64%
PW13	1	-	-	-	28	25	89,29%
PW14	2	2	2	100,00%	52	31	59,62%
PW15	1	-	-	-	25	16	64,00%
PW17	3	6	6	100,00%	66	65	98,48%
PW18	2	2	2	100,00%	40	32	80,00%
PW19	4	12	10	83,33%	64	55	85,94%
PW20	1	-	-	-	15	11	73,33%
PW21	3	6	6	100,00%	36	35	97,22%
PW22	2	2	2	100,00%	20	20	100,00%
PW23	2	2	0	0,00%	16	16	100,00%
PW24	1	-	-	-	7	7	100,00%
PW25	3	6	0	0,00%	12	8	66,67%
PW26	1	-	-	-	3	3	100,00%
PW28	3	6	6	100,00%	0	0	-
Total	53	98	84	85,71%	1329	1132	85,17%

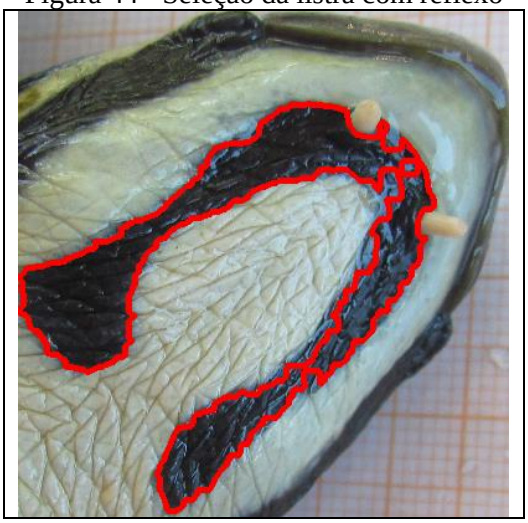
Nas comparações intra-classe a imagem de um cágado é comparada com todas as outras imagens do mesmo cágado. O objetivo deste tipo de comparação é comprovar a efetividade do protótipo no que diz respeito ao reconhecimento do mesmo cágado a partir de duas imagens distintas. Nas comparações inter-classe, as imagens de um cágado são comparadas com as imagens de outros cágados. Estas comparações têm por objetivo comprovar a precisão do protótipo no que diz respeito ao reconhecimento de cágados distintivos a partir de duas imagens.

A partir da Tabela 1, é possível observar que foram realizadas 98 comparações do tipo intra-classe, sendo que 84 delas foram classificadas de forma correta pelo protótipo, ou seja, o

protótipo indicou que os cágados das duas imagens eram os mesmos indivíduos. A partir destes resultados, conclui-se que o protótipo obteve uma taxa de acerto intra-classe de 85,21%.

Analisando as imagens processadas, pode-se identificar alguns motivos que corroboraram para a diminuição da taxa de acerto. O primeiro motivo é a presença de reflexo sobre a listra em formato de ferradura. Esta característica pode ser observada em algumas das imagens dos indivíduos `PW12, PW19 e PW23`, justificando a baixa taxa de acerto apresentada na Tabela 1. A Figura 44 mostra um exemplo da seleção da listra em formato de ferradura de uma imagem com reflexo.

Figura 44 - Seleção da listra com reflexo



A partir da Figura 44, pode-se observar que a presença de reflexo impossibilita que o protótipo faça a seleção correta da listra, dificultando a sua identificação durante as comparações. Ele, interfere na etapa de binarização, visto que nesta etapa é utilizada a luminosidade para definir a cor dos pixels.

Outra característica que dificultou a identificação dos cágados foi o ângulo em que a foto foi tirada. Esta característica pode ser observada em duas imagens do indivíduo `PW25`, conforme mostra a Figura 45.

Figura 45 - Imagens com inclinação

Conforme demonstrado na Figura 45, no momento em que o grau de inclinação da foto é muito alto, a listra em formato de ferradura sofre alterações no seu formato, dificultando a identificação do cágado. Tal problema acontece devido ao fato de que o formato desta listra é a principal característica utilizada na seleção da mesma, sendo assim o ideal é que a foto do cágado seja capturada de cima.

Outra característica que impossibilitou a identificação correta dos cágados por parte do protótipo foi a falta de nitidez/foco da imagem de entrada. Esta deficiência impacta no processo de remoção de ruídos, onde o filtro de erosão sobre a ferradura resulta no afinamento da mesma.

Por fim, pode-se observar na Tabela 1 que foram realizadas 1329 comparações inter-classe, sendo que 1132 foram identificadas corretamente, ou seja, o protótipo acusou que os cágados das duas imagens comparadas não eram os mesmos. A partir desses valores, pode-se afirmar que a taxa de acerto inter-classe do protótipo é de 85,17%.

Esta taxa de acerto não é a ideal, e novamente foram identificados alguns problemas que contribuem para a queda desta taxa. Estes problemas são os mesmos observados durante o processo de comparação intra-classe apresentados acima. Novamente a falta de nitidez da imagem, a inclinação da foto e a presença de reflexo na listra em formato de ferradura contribuíram para a queda da taxa de acerto.

4 CONCLUSÕES

A utilização de métodos não-invasivos na identificação individual de animais se mostra mais efetiva quando comparada a métodos mais invasivos visto que o nível de *stress* gerado ao animal é menor. Dado este cenário, torna-se de grande interesse a criação de ferramentas que permitem fazer a identificação de animais de modo não-invasivo.

Este trabalho propôs o desenvolvimento de um protótipo para reconhecimento de cágados da espécie *Phrynops williamsi* a partir de uma imagem informada pelo usuário. Entre os objetivos do trabalho tem-se a segmentação da imagem, a seleção da listra em formato da ferradura e da listra circular caso esta exista, discriminação da forma destas listras através dos descritores de forma e comparação desta com os cágados do banco de dados.

O desenvolvimento do protótipo foi realizado na IDE Visual Studio 2012 utilizando a linguagem C#. A interface foi implementada com a biblioteca Windows Forms. A biblioteca EmguCV foi utilizada no modulo de processamento de imagem. A persistência de dados foi realizada através da biblioteca SQLite.

Os resultados obtidos através dos testes do protótipo se mostraram satisfatórios. No que diz respeito as comparações intra-classe o protótipo obteve uma taxa de acerto de 85,71%. Já nas comparações inter-classe a taxa de acerto obtida foi de 85,17%.

Por fim, conclui-se que protótipo desenvolvido pode ser utilizado por pesquisadores e biólogos de campo que estudam essa espécie de cágado, auxiliando na identificação do mesmo através de um método não-invasivo. Além disso, ele também pode ser utilizado como base para outros trabalhos com foco na identificação de outras espécies de cágados e/ou tartarugas que venham a utilizar a imagem das mesmas como entrada.

4.1 LIMITAÇÕES

A primeira limitação apresentada pelo protótipo de identificação foi a não capacidade de identificar cágados que possuem a listra em formato de ferradura conexa com outras listras. Esta limitação ocorre no momento em que a listra em formato de ferradura é selecionada, visto que o perímetro e a área da mesma são as principais características utilizadas na seleção. Um exemplo de tal limitação é demonstrado na Figura 46.

Figura 46 - Listra em formato de ferradura não desconexa

Na Figura 46 tem-se um exemplo de um cágado em que a listra em formato de ferradura é conexa com a listra externa. Além desta, o protótipo apresenta as seguintes limitações:

a) a região de interesse (cabeça do cágado), não é selecionada pelo protótipo. É necessário a intervenção do usuário para realizar tal procedimento;
b) trabalha apenas com identificação de imagens de alto contraste das listras negras e da pele de cor clara;
c) funciona apenas na plataforma Desktop.

4.2 EXTENSÕES

Como sugestões de extensões para o protótipo propõem-se:

a) implementar a funcionalidade para seleção da região de interesse;
b) melhorar a técnica de seleção da componente, a fim de permitir o reconhecimento de cágados que possuam faixa em formato circular conexa à outras faixas;
c) realizar tratamento para remoção de reflexo das imagens;
d) permitir que o usuário possa cadastrar várias imagens para um mesmo cágado;
e) exibir uma lista dos 10 cágados mais relevantes ordenadas de modo descendente pelo nível de similaridade, permitido a classificação por parte do usuário;
f) ajustar o protótipo para que seja multiplataforma;
g) transformar o protótipo em um *App* para dispositivos móveis.

REFERÊNCIAS

ARAÚJO, Ana Paula Ulian de Araújo; BOSSOLAN, Nelma Regina Segnini. **Noções de Taxonomia e Classificação**: Introdução à Zoologia. 2006. 50 f. Licenciatura em Ciências Exatas – Instituto de Física de São Carlos, Universidade de São Paulo, [São Carlos]. Disponível em: <http://biologia.ifsc.usp.br/bio2/apostila/bio2_apostila_zoo_01.pdf>. Acesso em: 13 set. 2015.

BABOO, Capt Dr.S Santhosh; Vigneswari, A. R. J. Identification of Olive Ridley Turtle Using Feature Extraction. In: 2014 INTERNATIONAL CONFERENCE ON ITELLIGENT COMPUTING APPLICATIONS, [s.n.], 2014, Coimbatore. **Proceedings …** Coimbatore: Department of Computer Applications Bharathiar University, 2014. p. 39-72. Disponível em: <http://dx.doi.org/10.1109/ICICA.2014.23>. Acesso em: 27 jun. 2016.

BALESTRA, Rafael A. M. et al. Roteiro para Inventários e Monitoramentos de Quelônios Continentais. **Biodiversidade Brasileira**, [s.l.], v.16, n. 1, p. 114-152, ago. 2015. Disponível em: <http://www.icmbio.gov.br/revistaeletronica/index.php/BioBR/article/view/471/459>. Acesso em: 27 jun. 2016.

BEUGELING, Trevor; BRANZAN-ALBU, Alexandra. Computer Vision-Based Identification of Individual Turtles Using Characteristic Patterns of Their Plastrons. In: COMPUTER AND ROBOT VISION (CRV), 2014, Montreal. **Proceedings...** Montreal: IEEE, 2014. p. 203-210. Disponível em: <http://dx.doi.org/10.1109/CRV.2014.35>. Acesso em: 27 ago. 2015.

BURGHARDT, Tilo. **Visual Animal Biometrics**: Automatic detection and individual identification by coat pattern. 2008. 177 f. Dissertação (Doutorado em Filosofia) – Departamento de Ciência da Computação, Universidade de Bristol, Inglaterra. Disponível em: <http://www.bmva.org/thesis-archive/2008/2008-burghardt.pdf>. Acesso em: 17 set. 2015.

CASSANIGA, Arno W. **Planatarum**: API para reconhecimento de plantas. 2012. 103 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau. Disponível em: <http://dsc.inf.furb.br/tcc/index.php?cd=6&tcc=1511>. Acesso em: 29 ago. 2015.

COSTA, Luciano F.; CESAR JR., Roberto M. **Shape analysis and classification**: theory and practice. Boca Raton: CRC Press, 2000.

DODD JR., C. Kenneth. **Amphibian Ecology and Conservation**: A Handbook of Techniques (Techniques in Ecology & Conservation): New York, Oxford University Press, 2009.

EL-GHAZAL, Akrem; BASIR, Otman; BELKASIM, Saeid. Farthest point distance: A new shape signature for Fourier. **Signal Processing: Image Communication**, [s.l.], v. 24, n. 7, p. 572–586, Ago. 2009. Disponível em: <http://dx.doi.org/10.1016/j.image.2009.04.001>. Acesso em: 10 jun. 2016.

GONZALES, Rafael C.; WOODS, Richard E. **Digital image processing**. 3rd ed. Upper Saddle River: Pearson Prentice Hall, 2008.

HICKMAN, Cleveland P.; ROBERTS, Larry S.; LARSON, Allan. **Integrated Principles of Zoology**. 11th ed. Boston: McGraw-Hill Science/Engineering/Math, 2001.

KADIR, Abdul et al. Leaf classification using shape, color, and texture features. **International Journal of Computer Trends and Technology**, [S.l], v. 1, n. 3, p. 225-230, July/Aug. 2011. Disponível em: <http://arxiv.org/pdf/1401.4447>. Acesso em: 27 jun. 2016.

MATHWORKS, Inc. **Formatting and Annotation**: Add labels, adjust colors, define axis limits, apply lighting or transparency, set camera view. [s.l.], 2016. Disponível em: < http://www.mathworks.com/help/matlab/formatting-and-annotation.html>. Acesso em: 20 jun. 2016.

MCDIARMID, Roy W. et al. **Reptile Biodiversity**: Standard Methods for Inventory and Monitoring. London: University of California Press, 2012.

POUGH, F. Harvey; JANIS, Christine M.; HEISER, John B. **A vida dos vertebrados**. 4ª ed. São Paulo: Atheneu São Paulo, 2008.

NIXON, Mark S.; AGUADO, Alberto S. **Feature Extraction and Image Processing**. Woburn: Newnes, 2002.

RUSS, John C. **The Image Processing Handbook**. 2nd. Boca Raton: CRC Press, 1995.

SPIER, Edson F. et al. Registro de Phrynops williamsi (Rhodin & Mittermeier, 1983) no rio do Peixe, centro-oeste de Santa Catarina, Brasil. **Revista Brasileira de Biociências**, Porto Alegre, v. 12, n. 1, p. 56-57, Jan./Mar. 2014. Disponível em: <http://www.ufrgs.br/seerbio/ojs/index.php/rbb/article/view/2108>. Acesso em: 01 nov. 2015.

VOGT, R. C. et al. **Avaliação do Risco de Extinção de Phrynops williamsi Rhodin & Mittermeier, 1983 no Brasil**. Goiânia, 2015. Disponível em: <http://www.icmbio.gov.br/portal/biodiversidade/fauna-brasileira/estado-de-conservacao/7421-repteis-phrynops-williamsi-cagado-rajado.html>. Acesso em: 23 jun. 2016.

ZAHN, Charles T.; ROSKIES, Ralph Z. Fourier Descriptors for Plane Closed Curves. **IEEE Transaction in computers**, [S.l.], v. C-2, n. 3, p. 269-281, Mar. 1972. Disponível em: < http://dx.doi.org/10.1109/TC.1972.5008949>. Acesso em: 19 set. 2015.

APÊNDICE A – Detalhamento dos casos de uso

Nesta seção são apresentados os detalhamentos dos casos de uso, com descrição, pré-condições, fluxos principais, alternativos, de exceção e pós-condições.

No `UC01 – Identifica indivíduo` o usuário pode fazer a identificação de um cágado a partir da sua imagem. O detalhamento deste caso de uso é descrito no Quadro 22.

Quadro 22 - UC01 Identifica indivíduo

Número	01
Caso de uso	Identificar indivíduo
Ator principal	Usuário
Pré-condições	Usuário deve possuir a imagem do cágado no computador
Fluxo principal	1. Usuário seleciona a opção `Analisar Cágado`; 2. Sistema exibe a tela de análise; 3. Vai para o caso de uso `UC03 - Carregar Imagem`; 4. Usuário clica no botão `Analisar`; 5. Sistema imprime em tela a imagem em todas as etapas; 6. Sistema busca cágado da imagem no banco de dados; 7. Sistema encontra cágado e exibe as informações do mesmo em tela; 8. Fim do caso de uso.
Fluxo Alternativo	`A01 - Cágado não encontrado` No passo 7 do `fluxo principal`, o sistema não encontra o cágado na base de dados. 1. Sistema não encontra cágado na base de dados; 2. Sistema exibe a mensagem de confirmação “Deseja cadastrar cágado?”; 3. Usuário confirma a operação clicando no botão `Sim`; 4. Vai para o caso de uso `UC02 - Registrar indivíduo`; 5. Retorna para o passo 8 do `fluxo principal`.
Fluxo Alternativo	`A02 - Usuário opta por não cadastrar cágado` No passo 3 do fluxo alternativo `A01 - Cágado não encontrado`, o usuário opta por não realizar o cadastro do cágado. 1. Usuário não confirma a operação clicando no botão `Não`; 2. Sistema fechar a mensagem; 5. Retorna para o passo 8 do `fluxo principal`.
Fluxo de Exceção	`E01 - Nenhuma imagem selecionada` No passo 4 do `fluxo principal`, caso o usuário não tenha selecionado nenhuma imagem o sistema exibe a mensagem "Por favor, selecione uma imagem." e retorna para o passo 2 do `fluxo principal`.
Pós-condições	Cágado é identificado pelo sistema

No `UC02 – Registrar indivíduo` o usuário pode fazer a inserção de um cágado na base de dados. O detalhamento deste caso de uso é descrito no Quadro 23.

Quadro 23 - UC02 Registrar indivíduo

Número	02
Caso de uso	Registrar indivíduo
Ator principal	Usuário
Pré-condições	Usuário ter realizado caso de uso `UC01 - Identificar indivíduo`

Fluxo principal	1. Sistema exibe a tela de Cadastro de Cágados; 2. Sistema carrega os campos Imagem e Nome automaticamente a partir da tela de reconhecimento; 3. Usuário preenche os campos Nome do pesquisador, Cidade/Estado da coleta, Latitude, Longitude, Data da coleta, E-mail do pesquisador e Observações; 4. Usuário clica no botão Cadastrar; 5. Sistema válida se os campos preenchidos estão corretos; 6. Sistema realiza a inclusão do cágado; 7. Sistema exibe a mensagem de sucesso "Cágado cadastrado com sucesso."; 8. Sistema retorna para a tela de análise; 9. Fim do caso de uso.
Fluxo de Exceção	E01 - Usuário não informou os campos obrigatórios No passo 5 do fluxo principal, caso o usuário não tenha informado Nome do pesquisador, Cidade/Estado da coleta ou Data da coleta, o sistema exibe uma mensagem reportando a falta da informação e retorna para o passo 3 do fluxo principal.
Fluxo de Exceção	E02 - Usuário preenche data inválida No passo 5 do fluxo principal, caso o usuário tenha preenchido uma data inválida no campo Data da coleta, o sistema exibe a mensagem "Data da coleta inválida" e retorna para o passo 3 do fluxo principal.
Pós-condições	Cágado é cadastrado

No UC03 – Carregar Imagem o usuário pode fazer o carregamento/seleção da imagem do cágado do seu computador para a tela de identificação do protótipo. O detalhamento deste caso de uso é descrito no Quadro 24.

Quadro 24 - UC03 Carregar Imagem

Número	03
Caso de uso	Carregar Imagem
Ator principal	Usuário
Pré-condições	Usuário possuir a imagem do cágado no seu computador
Fluxo principal	1. Sistema exibe a janela para seleção de arquivos; 2. Usuário seleciona uma imagem e pressiona o botão Abrir; 3. Sistema fecha a janela de seleção e exibe o caminho da imagem no campo Imagem; 4. Fim do caso de uso.
Fluxo de Exceção	E01 - Formato de arquivo inválido No passo 2 do fluxo principal, caso o usuário selecione um arquivo inválido. 1. Usuário seleciona um arquivo com extensão diferente de *.jpg, *.png ou *.gif e pressiona o botão Abrir; 2. Sistema exibe a mensagem "Formato do arquivo é inválido."; 3. Retorna para o passo 1 do fluxo principal.
Pós-condições	Imagem do cágado é carregada em tela