UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

MOVEREMUS: UM PROTÓTIPO PARA CALCULAR O ÍNDICE FUNCIONAL DO CIÁTICO EM RATOS VIA PROCESSAMENTO DE IMAGENS

JONATHAN DE SOUZA

BLUMENAU
2016

JONATHAN DE SOUZA

MOVEREMUS: UM PROTÓTIPO PARA CALCULAR O ÍNDICE FUNCIONAL DO CIÁTICO EM RATOS VIA PROCESSAMENTO DE IMAGENS

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

MOVEREMUS: UM PROTÓTIPO PARA CALCULAR O ÍNDICE FUNCIONAL DO CIÁTICO EM RATOS VIA PROCESSAMENTO DE IMAGENS

Por

JONATHAN DE SOUZA

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, Doutor – FURB

Blumenau, 06 de julho de 2016

Dedico este trabalho a minha família, que sempre me apoiou e incentivou nos estudos.

AGRADECIMENTOS

À minha família por toda a formação que me deram e me levaram a ser a pessoa que sou hoje e por sempre acreditarem em mim.

Aos meus amigos por entenderem que ao fazer um trabalho como esse muito tempo é necessário e aos que contribuíram com ideias em alguns pontos do desenvolvimento do trabalho.

Ao meu orientador Aurélio Hoppe, por ter me ajudado desde a escolha do tema e tem me auxiliado desde então para concretizar esse trabalho. Por suas revisões nos meus textos e tempo dedicado para retirada de dúvidas e realização do experimento para coleta de imagens.

Procure ser um homem de valor, em vez de ser um homem de sucesso.

Albert Einstein

RESUMO

Este trabalho apresenta o desenvolvimento de um protótipo para auxiliar fisioterapeutas no cálculo do Índice Funcional do Ciático (IFC) em ratos. O protótipo isola o corpo do rato nas imagens e depois localiza as suas patas, calculando as suas dimensões. Em seu desenvolvimento, foram aplicadas técnicas de processamento de imagens para separação de canais dos modelos de cores RGB e HSV, segmentação, operadores morfológicos e detecção de contornos, tendo como intuito automatizar a busca da altura e largura das patas para cálculo do IFC. A partir dos experimentos realizados, obteve-se 35% de acerto nos valores das dimensões das patas traseiras. O protótipo obteve sucesso ao encontrar o corpo do rato e suas patas, mas falhou na definição da escala e dos tamanhos das patas ao processar imagens com baixa qualidade.

Palavras-chave: Fisioterapia. Rato. Patas. Índice funcional do ciático. Processamento de imagens.

ABSTRACT

This work presents the development of a prototype to assist physiotherapists in calculating the Sciatic Functional Index (SFI) in rats. The prototype isolates the body mouse on the images and then located their legs, calculating its dimensions. In its development, image for color separation models of channel processing techniques were applied RGB and HSV, segmentation, morphological operators and edge detection, with the aim to automate the search height and width of the legs for IFC's calculation. From experiments, it obtained 35% accuracy in dimension values of the hind paws. The prototype was successful to find the body of the mouse and your legs, but failed to define the scale and size of the legs to process images with low quality.

Key-words: Physiotherapy. Mouse. Paws. Sciatic functional index. Image processing.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

EE – Elemento Estruturante

HSV – *Hue Saturation Value*

IFC – Índice Funcional do Ciático

JNI – *Java Native Interface*

LNP – Lesão Nervosa Periférica

RF – Requisitos Funcionais

RGB – *Red Green Blue*

RNF – Requisitos Não Funcionais

ROI – *Region Of Interest*

UI – *User Interface*

UML – *Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

A fisioterapia é conhecida como uma ciência que estuda, previne e trata os distúrbios cinéticos funcionais intercorrentes em órgãos e sistemas do corpo humano. Tais distúrbios podem ser provenientes de alterações genéticas, traumas ou por doenças adquiridas (CONSELHO NACIONAL DE FISIOTERAPIA, 2015).

Segundo Possamai et al. (2012), dentre os vários ramos da fisioterapia existe a fisioterapia neurofuncional, que estuda tratamentos para Lesões Nervosas Periféricas (LNP). Em pesquisas recentes, estima-se que cerca de 1% a 2% dos pacientes atendidos nos prontos socorros, possuem esse tipo de lesão. Uma LNP se recupera com o tempo, quando as terminações nervosas afetadas regeneram seus axônios. Mas, apesar dessa capacidade de reparação, os resultados funcionais das recuperações de LNP em humanos são por muitas vezes desapontadoras (POSSAMAI et al., 2012).

No estado da arte, encontram-se muitos trabalhos que procuram avaliar novas possibilidades de recuperação de uma LNP, tais como os trabalhos de Franco et al. (2011), Monte-Raso, Barbieri e Mazzer (2006) e Costa, Camargo e André (2008). Suas abordagens, no geral se valem de experimentos realizados a partir de lesões do nervo ciático na perna de ratos, em que se esmaga o nervo ciático a fim de danificá-lo, mas sem causar o rompimento completo do mesmo.

Para avaliar os resultados alcançados, os pesquisadores normalmente medem o valor da recuperação motora durante o período de tratamento e, posteriormente, calculam o IFC, conforme proposto por Medinacelli, Freed e Wyatt (1982). Neste método, as pegadas dos ratos são capturadas analogicamente utilizando alguma superfície que possa registrá-las. Porém, segundo Costa, Camargo e André (2008), o problema desta prática é que nas primeiras semanas as pegadas possuem muitas irregularidades e seus valores para o IFC são duvidosos.

Entretanto, recentemente surgiram diversos trabalhos que buscam avaliar o experimento por meio da captura de dados de forma digital, tendo como objetivo alcançar maior confiabilidade e estabilidade dos resultados durante os experimentos realizados. Tais abordagens se utilizam principalmente do Image-J para realizar a extração manual de características das pegadas a partir de imagens digitais (COSTA, CAMARGO e ANDRÉ, 2008).

Diante desse cenário, este trabalho apresenta a criação de um protótipo voltado especificamente para análise digital de pegadas de ratos, identificando e calculando automaticamente as dimensões de altura e largura das patas traseiras.

1.1 OBJETIVOS

O objetivo deste trabalho é desenvolver um protótipo computacional que automatize parte do processo de cálculo do IFC em ratos via processamento de imagens.

Os objetivos específicos do trabalho são:

a) isolar o corpo do rato em relação ao restante da imagem capturada;
b) localizar e calcular as dimensões de largura e a altura das patas traseiras do rato.

1.2 ESTRUTURA

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 são descritos a arquitetura do trabalho através de diagramas, o detalhamento da implementação do protótipo e os resultados obtidos nos testes realizados. Por fim, são apresentadas as conclusões e limitações do trabalho, assim como sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Essa seção demonstra alguns dos principais conceitos utilizados nesse trabalho. A Seção 2.1 apresenta o conceito do IFC, apresentando a sua finalidade e o modo para obtenção do seu valor. A Seção 2.2 demonstra o conceito do esquema de cores HSV. Na Seção 2.3 aborda-se segmentação de imagens e o algoritmo de limiarização. A Seção 2.4 dedica-se a explanar o conceito da morfologia matemática e suas principais operações. Por fim, a Seção 2.5 apresenta os trabalhos correlatos.

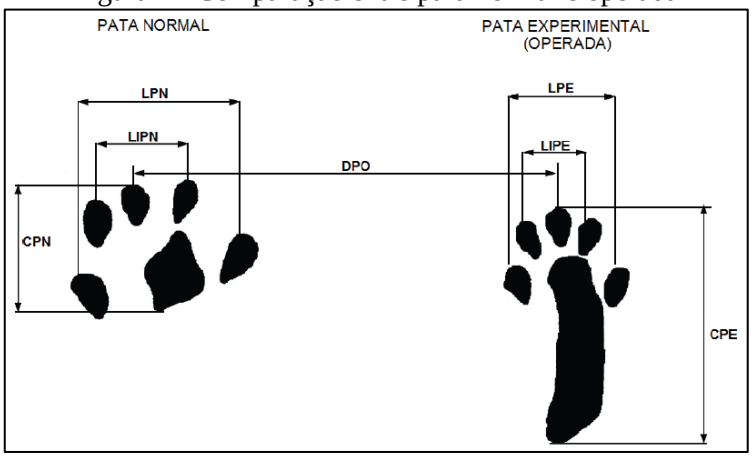
2.1 ÍNDICE FUNCIONAL DO CIÁTICO

Devido a sua distribuição e localização, os nervos são frequentemente traumatizados (MACHADO, 2000), deixando como consequências a perda ou diminuição da sensibilidade e de suas funções (MONTE-RASO, BARBIERI e MAZZER, 2006). Dentre esses traumas, existe a LNP, muito comum em acidentes de carro que causam perda total ou parcial da movimentação da perna devido a um trauma no nervo ciático (POSSAMAI et al., 2012).

Por causa disso, o ciático é um nervo que serve como objeto de estudo para análise de recuperação de uma LNP. Com o passar do tempo, os neurônios iniciam um processo de recuperação de seus axônios, na tentativa de reestabelecimento da comunicação nervosa (POSSAMAI et al., 2012). Esse processo de recuperação pode ser acelerado por meio da aplicação de técnicas de fisioterapia.

Segundo Possamai et al. (2012), foram desenvolvidos vários trabalhos na área de fisioterapia que apresentaram diferentes formas de estímulos para recuperação da lesão nervosa. Porém, o método o mais adotado para a realização da avaliação da recuperação é o IFC, conforme proposto por Medinacelli, Freed e Wyatt (1982). O IFC se baseia nas diferenças da pegada de uma pata saudável comparada a uma pata que sofreu lesão do ciático, conforme pode ser observado na Figura 1.

Figura 1 - Comparação entre para normal e operada



Fonte: Cavalcante (2011).

Medinacelli, Freed e Wyatt (1982) estabeleceram que para calcular o IFC, são necessários os valores das distâncias entre o 1º e 5º dedos e do 2º e 4º dedos, além do comprimento da pata. A fórmula do cálculo do IFC é apresentada no Quadro 1.

Quadro 1 - Fórmula do cálculo do IFC

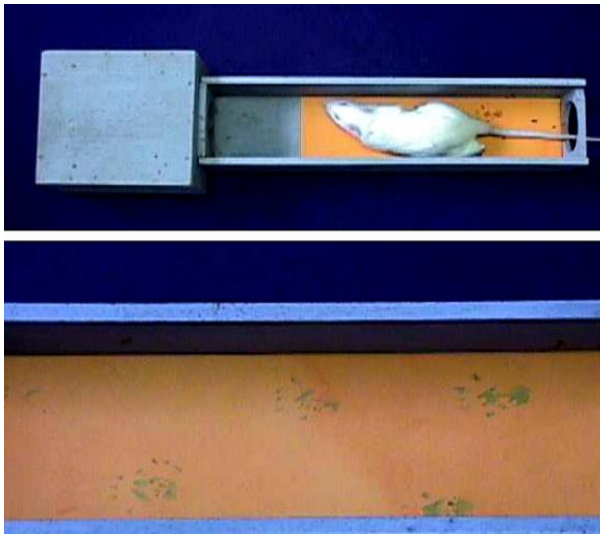
IFC =	-38,3	$\frac{(EPL-NPL)}{NPL}$	109,5	$\frac{(ETS-NTS)}{NTS}$	13,3	$\frac{(EIT-NIT)}{NIT}$	
EPL = comprimento da pegada da pata experimental; NPL = comprimento da pegada da pata normal; ETS = distância entre o primeiro e o quinto artelho da pata experimental; NTS = distância entre o primeiro e o quinto artelho da pata normal; EIT = distância entre o segundo e o quarto artelho da pata experimental; NIT = distância entre o segundo e o quarto artelho da pata normal.							

Fonte: adaptado de Franco et al. (2010).

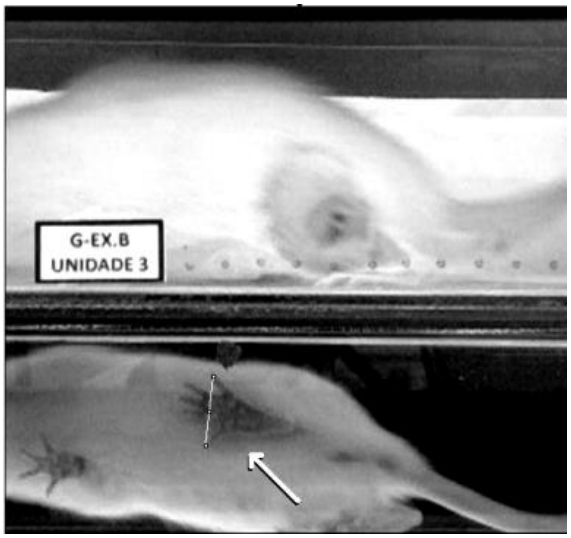
Segundo Franco et al. (2011), os valores obtidos pelo cálculo do IFC podem variar entre zero e cem negativo, sendo que valores próximos de zero indicam funcionamento normal do nervo ciático, enquanto valores próximos de cem negativo indicam maior perda da função motora.

Para coleta dos valores de entrada do cálculo do IFC existem atualmente duas técnicas muito utilizadas. No método mais tradicional, os ratos são colocados para andar sobre uma passarela com um papel marcado por tinta das pegadas (Figura 2a), conforme apresentado por Monte-Raso, Barbieri e Mazzer (2006). Como evolução dessa técnica, existe um processo em que a coleta é feita de forma digital (Figura 2b), em que o rato anda por uma passarela de vidro e tem as suas pegadas capturadas por uma câmera fotográfica (FRANCO et al., 2011).

Figura 2 – Tipos de coletas dos valores de entrada do IFC



(a) Passarela de papel



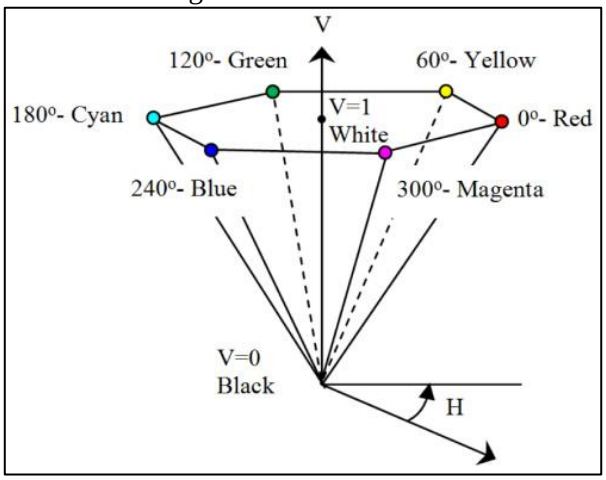
(b) Passarela de vidro

Segundo Monte-Raso, Barbieri e Mazzer (2006), nas duas primeiras semanas os valores calculados por meio do método tradicional não são confiáveis devido à má qualidade das pegadas (borrões e manchas), associada à dificuldade dos ratos em realizarem um movimento regular após a realização da LNP. Com a regeneração do nervo e recuperação funcional do animal, as pegadas se tornaram mais claras. No entanto, Franco et al. (2011) afirmaram que o método de coleta por meio de vídeos não apresenta tais empecilhos, pois é possível capturar as imagens/instantes em que as patas estão bem definidas.

2.2 MODELO DE COR HSV

O modelo de cores *Hue Saturation Value* (HSV), é formado canais *Hue* (Matiz), *Saturation* (Saturação) e *Value* (Intensidade). O modo de representação do modelo HSV é através de um cone hexagonal, conforme representado na Figura 3.

Figura 3 - Modelo HSV



Fonte: Yilmaztürk (2011).

O canal da intensidade é a escala de cinza, sendo representado pelo eixo central que varia de zero para a cor preta até um para a cor branca. A saturação é a profundidade ou pureza da cor, sendo medida como uma distância radial a partir do eixo central, variando de zero no centro até um na borda externa. A matiz é definida pelo ângulo ao redor do eixo central, sendo 0º para vermelho, 60º para amarelo, 120º para verde, 180º para ciano, 240º para azul e 300º para magenta (SURAL, QIAN e PRAMANIK, 2002).

Com o nível da saturação em zero e alterando o nível do eixo de valor temos uma variação de preto até branco, pelos tons cinza entre eles. Caso o valor da saturação seja variado entre uma determinada intensidade e matiz, a cor resultante varia entre um tom de cinza e a forma mais pura possível para a cor definida pela matiz (SURAL, QIAN e PRAMANIK, 2002). Com isso, quanto maior o valor da saturação, maior é a percepção da presença de uma cor para o olho humano.

Valores de intensidade próximos a zero possuem pouca percepção de cor mesmo com alto valor para a saturação, em qualquer nível da matiz (SURAL, QIAN e PRAMANIK, 2002). Com isso, valores mais altos de intensidade proporcionam uma maior percepção da variação de tons das cores, enquanto valores baixos resultam em cores próximas ao preto.

O valor radial da matiz indica qual cor está sendo variada pelos valores de saturação e intensidade. Ele inicia no vermelho nos seus níveis mais baixos, passando para amarelo, verde, ciano, azul e magenta até retorna para o vermelho nos mais altos níveis.

O modelo de cor HSV é muito utilizado em técnicas de processamento de imagens baseadas em propriedades do sistema visual humano. Tal fato se justifica pelos canais de matiz, saturação e intensidade estarem intimamente ligados com o modo que o olho humano percebe as cores (GONZALES e WOODS, 2000, p. 162).

2.3 SEGMENTAÇÃO

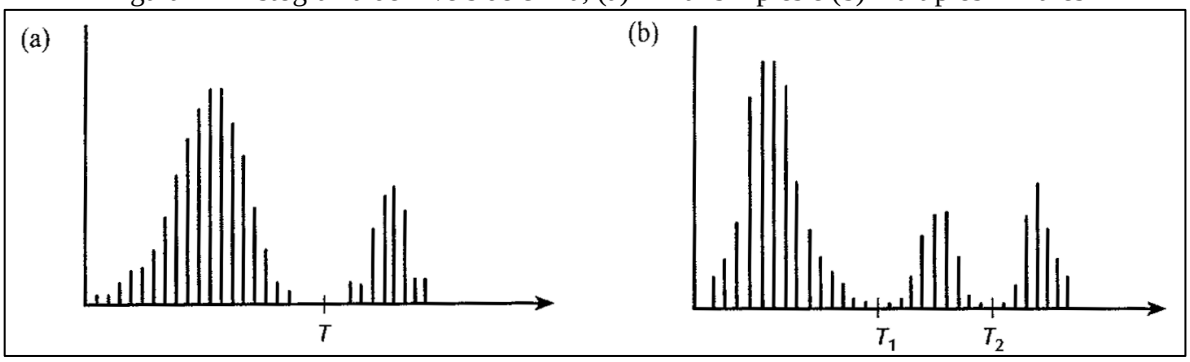
Uma imagem em níveis de cinza é composta por diversas regiões que representam diversas informações que podem ser consideradas (FACON, 1996, p. 261). Também segundo Facon (1996, p. 261), no processamento de imagens a segmentação é a tarefa responsável por realizar a extração dessas informações relevantes para determinado problema.

Conforme Facon (1996, p. 261), a segmentação é um dos grandes problemas do processamento de imagens. Isso se deve ao fato de haverem muitas informações inerentes misturadas ao objeto de estudo, além de diversos outros tipos de ruídos que atrapalham a busca pelas informações relevantes.

Segundo Gonzales e Woods (2000, p. 295), os algoritmos de segmentação para imagens monocromáticas são geralmente baseados nas seguintes propriedades básicas: descontinuidade e similaridade. Na primeira propriedade, a imagem é dividida levando em consideração alterações bruscas dos níveis de tons de cinza, já a segunda categoria procura identificar regiões similares em tons de cinza.

Dentre as técnicas de similaridade, existe a limiarização, que, segundo Gonzales e Woods (2000, p. 316), é uma das mais importantes abordagens para a segmentação de imagens. A limiarização busca encontrar regiões semelhantes de acordo com o seu nível de tons de cinza. Para isso são definidos limiares que definem a divisão entre esses grupos de pixels. Na Figura 4(a) é possível verificar um exemplo de dois grupos de pixels separados por um limiar T.

Figura 4 - Histograma de níveis de cinza, (a) limiar simples e (b) múltiplos limiares



Fonte: Gonzales e Woods (2000, p. 316).

Além da limiarização simples, na qual é definido apenas um limiar para divisão dos grupos, como exemplificado pela Figura 4(a), existe a limiarização multinível, como mostra a Figura 4(b), onde dois ou mais limiares são definidos para delimitar três ou mais agrupamentos de pixels. Segundo Gonzales e Woods (2000, p. 316), os limiares multiníveis são menos confiáveis do que um limiar simples, pois se torna mais complexo a determinação de vários limiares que possam isolar com efetividade as regiões de interesse. Nesses casos, um limiar único variável se torna a melhor opção.

O Quadro 2 apresenta o algoritmo que realiza a limiarização simples, também conhecida como *thresholding*. Na entrada ele recebe uma imagem em tons de cinza, e um valor de limiar. Para cada pixel da imagem, é feita uma avaliação para saber se o valor de sua intensidade é menor do que o limiar. Então o pixel recebe a cor preta, caso contrário é preenchido com a cor branca. O resultado da limiarização é uma imagem binarizada em preto em branco.

Quadro 2 - Algoritmo de limiarização

```
f = thresholding (f, X)
f: imagem de entrada
X: limiar de separação

para cada pixel p de f faça
   se f(p) ≤ X então
      f(p) ← 0
   senão
      f(p) ← 255
```

Fonte: adaptado de Klava (2006, p. 10).

A escolha do valor do limiar pode ser feita de maneira automática a partir da análise dos valores do histograma, gráfico com os valores dos níveis de cinza da imagem (KLAVA, 2006), conforme apresentado na Figura 4a. Nesse caso, é possível identificar facilmente a região de separação entre dois agrupamentos de pixels, assim o limiar é definido com esse valor o que resulta em uma separação bem sucedida desses dois possíveis objetos da imagem.

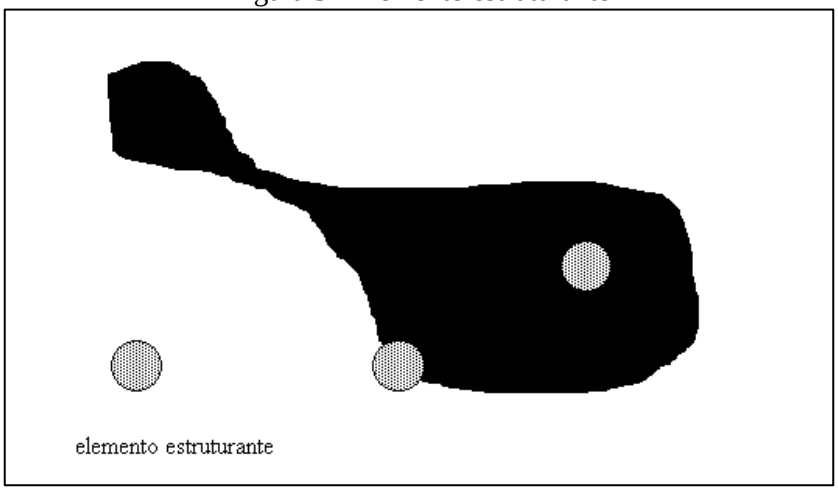
Mesmo uma definição feita pelo histograma pode acabar apresentando problemas, como no caso de imagens com muitos ruídos que tornam instável o gráfico dos níveis de cinza. Isso acaba por dificultar o processo para encontrar uma região que separe os objetos (KLAVA, 2006).

2.4 MORFOLOGIA MATEMÁTICA

A morfologia matemática é uma ferramenta que auxilia a extração de componentes úteis na imagem para a descrição de forma de uma região (GONZALES e WOODS, 2000, p. 369). Ainda segundo Gonzales e Woods (2000, p. 370), a linguagem da morfologia matemática é fortemente baseada na teoria dos conjuntos, fornecendo dessa forma uma abordagem poderosa para diversos problemas em relação ao processamento de imagens.

Segundo Facon (1996, p. 2), o princípio da morfologia matemática é extrair uma informação referente a geometria de um conjunto da imagem, aplicando uma transformação com outro conjunto denominado de Elemento Estruturante (EE). De acordo com sua forma e tamanho, é possível testar se o EE está ou não contido no conjunto desconhecido da imagem (FACON, 1996, p. 2), conforme a representação da Figura 5.

Figura 5 - Elemento estruturante



Fonte: Facon (1996).

A morfologia matemática apresenta diversas aplicações no processamento de imagens, como os operadores morfológicos que utilizam o EE para promover alterações na forma dos elementos existentes dentro da imagem. Segundo Facon (1996), existem dois operadores morfológicos básicos (dilatação e erosão), bem como suas combinações (abertura e fechamento).

2.4.1 Dilatação

A dilatação é definida no Quadro 3 conforme Facon (1996, p. 15). Na fórmula são apresentados dois conjuntos B e X, sendo B o conjunto do EE e X o conjunto original.

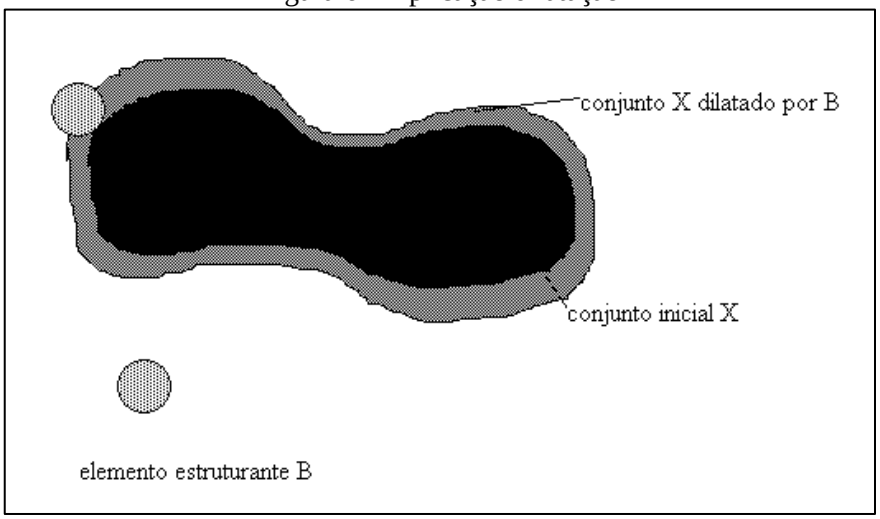
Quadro 3 - Definição dilatação

$$B = \{x \in X : B_x \cap X \neq \emptyset\}$$

Fonte: Facon (1996).

Ainda segundo Facon (1996, p. 15), a dilatação verifica se um ponto B_x possui intersecção com o conjunto X. Caso seja, verdade o ponto central o EE é fixado como um pixel relevante na imagem de retorno, adicionando essa nova região ao conjunto X original. Tal operação é demonstrada na Figura 6.

Figura 6 – Aplicação dilatação



Fonte: Facon (1996).

Por analogia, é possível dizer que a dilatação funciona como se o EE deslizasse pelo contorno do conjunto X, promovendo um crescimento da dessa região pelo seu ponto central. Com essa característica de expansão das regiões, ela possui os efeitos de preenchimento espaços vazios com tamanho inferior ao EE, além de conectar agrupamentos de pixels distintos que estão separados por uma distância inferior ao tamanho do EE.

2.4.2 Erosão

A erosão é definida no Quadro 4, conforme Facon (1996, p. 15). Nessa fórmula tem-se o conjunto original X e conjunto B pertencente ao EE.

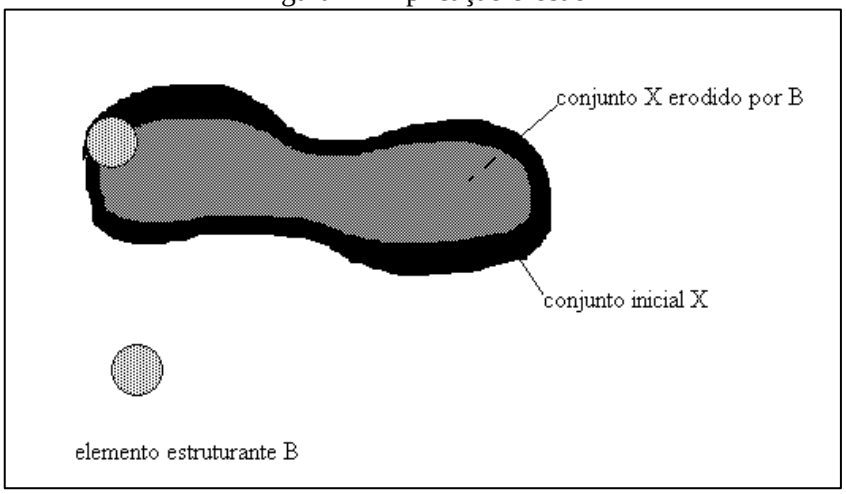
Quadro 4 - Definição erosão

$$B = \{x \in \mathcal{E} : B_x \subset X\}$$

Fonte: Facon (1996).

Segundo Facon (1996, p. 15), a erosão verifica o EE B_x está posicionado e centrado internamente ao pixel x de X. Caso seja verdade, o ponto central do EE é fixado como um pixel relevante na imagem de retorno, removendo essa região do conjunto X original. A operação descrita é demonstrada pela Figura 7.

Figura 7 – Aplicação erosão



Fonte: Facon (1996).

Pode se dizer que a erosão funciona como se o EE deslizasse internamente pelo conjunto X, removendo a região deste até o seu ponto central. Essa característica de diminuição dos agrupamentos de pixels da imagem faz com que pequenas regiões com tamanho inferior ao do EE sejam eliminadas, o que é útil na remoção de pequenos ruídos granulares na imagem. A erosão também possui as características de aumentar os furos presentes na imagem e de separar um conjunto de pixels que está ligado por uma região estreita com espessura inferior ao tamanho do EE.

2.4.3 Abertura

Segundo Facon (1996, p. 25), a operação de abertura é constituída em princípio pela sequência da aplicação da erosão e dilatação, nessa ordem e utilizando o mesmo EE. A definição da abertura é apresentada no Quadro 5. Nessa fórmula, temos o conjunto B que representa o EE e o conjunto original X.

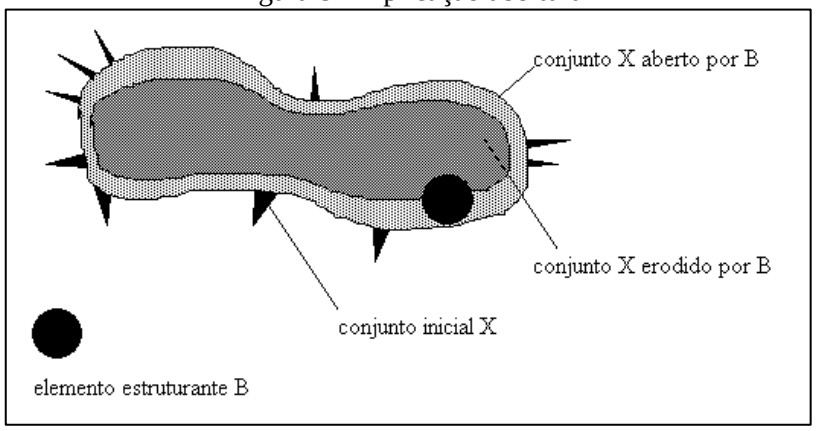
Quadro 5 - Definição abertura

$$B = (X \ominus \tilde{B}) \oplus B$$

Fonte: Facon (1996).

Conforme pode ser visto na Figura 8, o conjunto inicial X possuía regiões protuberantes de forma estreita que inicialmente são eliminadas pela erosão. Após a erosão, é aplicada uma dilatação que promove o crescimento do conjunto X ao seu tamanho original, mas sem os ruídos presentes no seu contorno inicialmente.

Figura 8 – Aplicação abertura



Fonte: Facon (1996).

De acordo com Facon (1996, p. 28), algumas características que surgem no resultado da abertura são: a separação de agrupamentos de pixels, pequenas partículas com tamanho inferior ao do EE, grande semelhança das formas resultantes com as originais e a redução de detalhes no conjunto aberto.

2.4.4 Fechamento

Segundo Facon (1996, p. 25), a operação de abertura é constituída em princípio pela sequência da aplicação da dilatação e erosão, nessa ordem e utilizando o mesmo EE. A definição do fechamento é apresentada no Quadro 6. Nessa fórmula, tem-se o conjunto B que representa o EE e o conjunto original X.

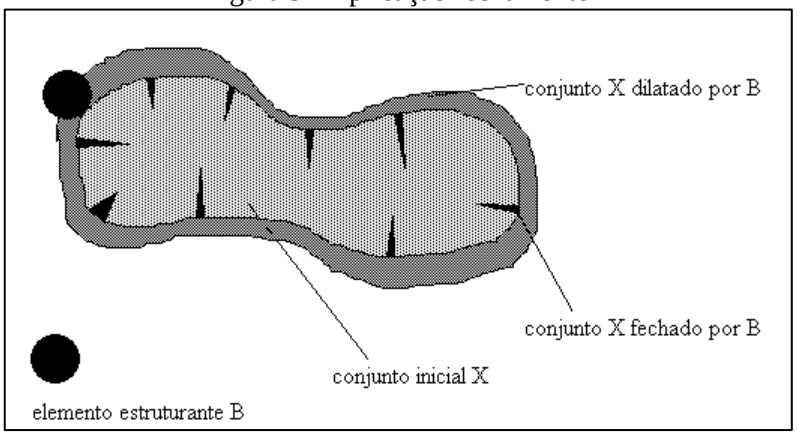
Quadro 6 - Definição fechamento

$$(X) = (X \oplus \tilde{B}) \ominus B$$

Fonte: Facon (1996).

A aplicação do fechamento é apresentada na Figura 9. Pode-se perceber que nessa imagem existiam pequenas fendas estreitas no conjunto X que foram removidas pela dilatação. Após a dilatação, o conjunto foi fechado com uma erosão que acabou por fazer o conjunto resultante possuir e mesma forma do conjunto X inicial.

Figura 9 - Aplicação fechamento



Fonte: Facon (1996).

De acordo com Facon (1996, p. 35), algumas das características resultantes do fechamento são: preenchimento de buracos com área inferior ao tamanho do EE, a junção de elementos distintos separados por uma distância menos que o tamanho da EE, as formas dos conjuntos originais são mantidas no conjunto fechado e a diminuição na quantidade de detalhes presentes na imagem fechada.

2.5 TRABALHOS CORRELATOS

Esta seção apresenta três trabalhos que possuem relação com o tema abordado neste trabalho. Na Seção 2.3.1 é apresentado o trabalho proposto por Monte-Raso, Barbieri e Mazzer (2006), que utiliza uma forma analógica de captura das pegadas de ratos para determinação do IFC. A Seção 2.3.2 descreve o trabalho de Franco et al. (2011), que realiza a coleta digital de informações durante a marcha de ratos para cálculo do IFC. A Seção 2.3.3 explica o trabalho de Yuan et al. (2006), que apresenta o reconhecimento de pegadas de ratos de diferentes espécies através de imagens.

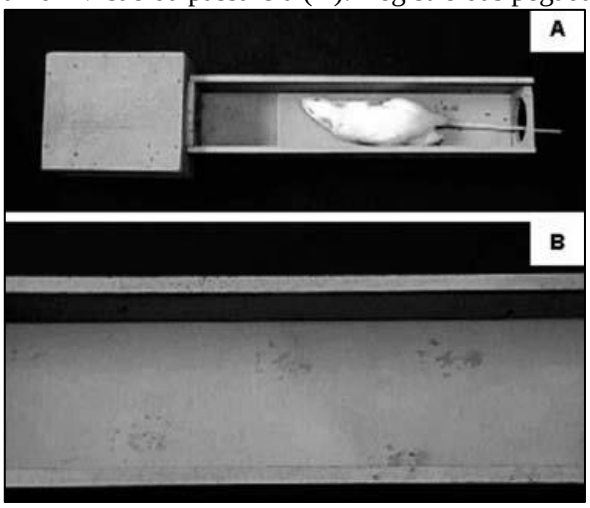
2.5.1 Índice funcional do ciático nas lesões por esmagamento do nervo ciático de ratos. Avaliação da reprodutibilidade do método entre examinadores

Monte-Raso, Barbieri e Mazzer (2006), criaram um procedimento para realização de aferição do valor IFC em ratos, após lesão do nervo ciático. Sua meta era verificar se tal método seria eficaz e viável para reprodução por outros estudiosos na área de lesões nervosas periféricas.

Para os experimentos foram usados cerca de 20 ratos Wistar adultos, que seriam submetidos a um procedimento cirúrgico no nervo ciático, para depois avaliar a sua recuperação. Com o intuito de averiguar sua recuperação, foi criada uma pista com uma folha de papel em sua base por onde o rato iria andar após ter as suas patas embebidas em solução

azul de bromofenol de acetona a 1%. Após esperar as pegadas secarem, as folhas de papel eram postas em scanner de alta resolução para digitalização de suas imagens. Os dados necessários para medição do IFC eram obtidos com um software onde em que marcados os pontos de interesse nas imagens das patas dos ratos e extraídos os valores das variáveis para o calculo da equação do IFC. A Figura 10 demonstra como são coletadas as pegadas utilizando este método de coleta.

Figura 10 - Visão da passarela (A). Registro das pegadas (B)



Fonte: Monte-Raso, Barbieri e Mazzer (2006).

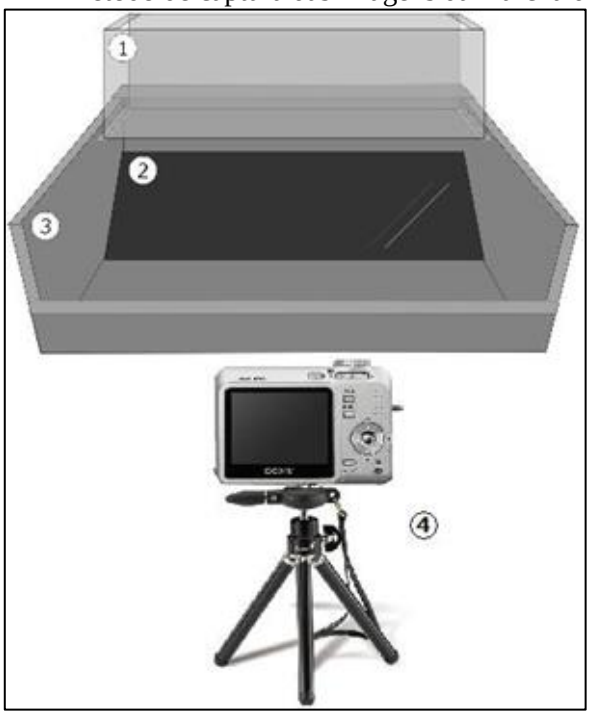
Tal método mostrou-se pouco eficaz nas duas primeiras semanas, pois os animais estavam em processo de recuperação da operação, o que por muitas vezes produzia pegadas com pouco destaque da pata a ser analisada. Sendo que, depois desse período inicial, os dados coletados são confiáveis e passíveis de reprodução.

2.5.2 Avaliação de método digital para análise do índice funcional do ciático em ratos

Franco et al. (2011) se deparam com a necessidade de encontrar uma forma mais simples e confiável para reproduzir o experimento da marcha de ratos para avaliação de seu IFC no período de recuperação de uma LNP. Com isso, seu estudo se voltou em descrever uma técnica da coleta digital de imagens utilizando câmeras digitais, para posterior avaliação no programa IMAGE-J.

Para criação desse novo método, foi utilizada uma passarela de vidro apoiada sobre uma caixa de madeira para suporte que possui um espelho afixado em ângulo de 45º para facilitar a captura de imagens. A frente desse espelho foi colocada uma câmera sobre um tripé para realização da captura das imagens do reflexo da parte inferior da passarela de vidro no espelho. A Figura 11 demonstra mais claramente os posicionamentos dos instrumentos acima citados.

Figura 11 - Método de captura das imagens da marcha de ratos



Fonte: Franco et al. (2011).

Na Figura 11, o item 1 representa a passarela de vidro, o item 2 o espelho, o item 3 a caixa de madeira e o item 4 a câmera com o tripé. Para extração dos frames do filme capturado pela câmera, foi utilizado o programa Microsoft® Windows® Movie Maker, para depois enviar essas imagens dos pés dos ratos ao programa IMAGE-J, para coleta dos valores dos dados necessários ao calculo do IFC.

A partir dessa técnica, pode-se obter uma boa consistência nas imagens capturadas em qualquer estágio do experimento. Diferentemente das formas analógicas, em que as pegadas dos ratos são registradas sobre algum papel utilizando tinta impregnada em suas patas e então, os valores são extraídos de maneira manual para determinar o IFC.

2.5.3 Entendendo pegadas de diferentes espécies de ratos

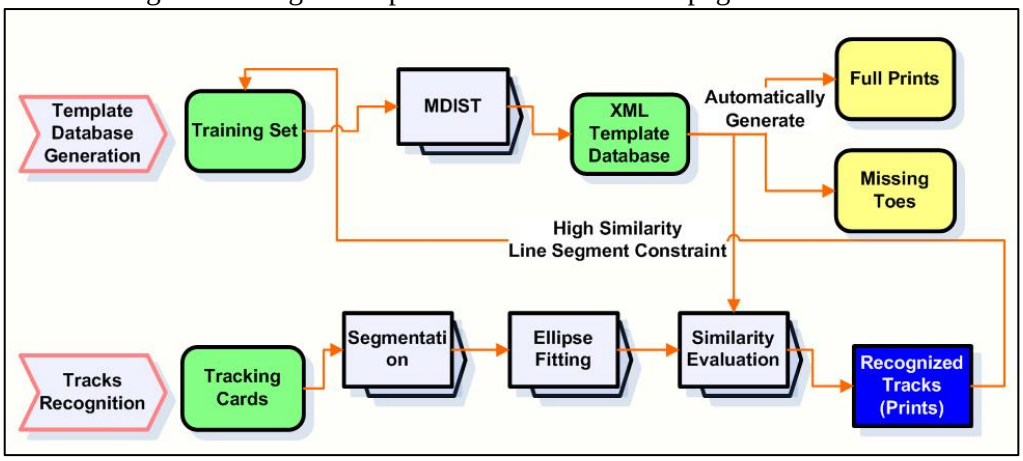
Yuan et al. (2006) propuseram um estudo que visava o reconhecimento de pegadas de ratos de diferentes espécies, sendo motivados pela necessidade de ajudar no controle populacional desses animais. A análise digital das imagens de pegadas dos ratos torna possível conferir uma melhor precisão na aferição da espécie de rato do qual ela pertence.

O método proposto começa com a coleta das pegadas dos ratos na natureza, utilizando armadilhas em formato de túnel com isca para atrair os ratos. Nesse túnel, eles andam sobre um cartão de rastreamento com tinta no fundo para registro de suas pegadas. Os cartões com as amostras coletadas são digitalizadas por um *scanner* e as imagens capturadas em tons de cinza são submetidas ao tratamento de um programa que executa o reconhecimento da pegada

em três etapas: aplicação de uma técnica comum de binarização de imagens, diferenciação e destaque com elipses dos artelhos e da almofada da pata e por fim uma função de avaliação linear é aplicada para realizar o reconhecimento dos padrões.

A terceira etapa utiliza uma base de padrões, em que cada nova pegada reconhecida é incluída num *dataset* para treinamento e posterior refinamento para as próximas análises. Na Figura 12 é demonstrado o fluxo do método proposto para o programa de reconhecimento das pegadas.

Figura 12 - Algoritmo para reconhecimento das pegadas dos ratos.



Fonte: Yuan et al. (2006).

A partir do trabalho de Yuan et al. (2006), foi possível constatar que havia uma infestação de ratos da espécie Norway numa pequena ilha da Nova Zelândia, exatamente como foi constatado por especialistas. Isso comprovou os resultados obtidos no estudo, onde foi constatada uma taxa de reconhecimentos positivos acima de 80%. Foi verificado que tais números poderiam ser melhorados com uma maior base de padrões para análise, mas isso acarretaria em maiores custos computacionais.

2.5.4 Comparativo entre os trabalhos correlatos

O Quadro 7 apresenta um comparativo entre as principais características dos trabalhos correlatos.

Quadro 7 - Características das trabalhos

Trabalhos/Características	Monte-Raso, Barbieri e Mazzer (2006)	Franco et al. (2011)	Yuan et al. (2006)
Origem das amostras	Pegadas eram captura das por um papel numa pista em laboratório	Registro de imagens fotográficas	Cartões com tinta em armadilhas na floresta
Modo de captura das imagens	Analógica	Digital	Analógica
Software utilizado	Proprietário, desenvolvido para o experimento	IMAGE-J	Proprietário, desenvolvido para o experimento
Utilização do IFC	Sim	Sim	Não
Intervenção humana na obtenção dos valores de entrada pelo programa	Sim	Sim	Não
Reconhecimento automático das patas	Não	Não	Sim

Com o Quadro 7, percebe-se que as abordagens dos trabalhos correlatos são distintos, no âmbito de Monte-Raso, Barbieri e Mazzer (2006) e Franco et al. (2011), utilizam programas para indicação dos pontos e valores de entrada, enquanto o trabalho de Yuan et al. (2006) realiza todo o processamento pelo próprio sistema.

Essas abordagens, se justificam pelo fato de Monte-Raso, Barbieri e Mazzer (2006) e Franco et al. (2011) necessitarem de valores precisos como entrada da equação do IFC, ao contrário do software desenvolvido por Yuan et al. (2006), que procurava reconhecer as formas das pegadas para a classificação de diversas espécies de ratos.

3 DESENVOLVIMENTO

Este capítulo apresenta as etapas para o desenvolvimento de um protótipo para busca das medidas das patas de ratos. Na seção 3.1 são listados os principais requisitos funcionais e não funcionais. A seção 3.2 apresenta as especificações do protótipo. A seção 3.3 explica os detalhes dos algoritmos e técnicas utilizadas no protótipo. Por fim a seção 3.4 demonstra os testes executados para validação do projeto, discutindo os seus resultados.

3.1 REQUISITOS

O protótipo desenvolvido possui os seguintes Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF):

a) permitir o carregamento da imagens das patas do rato (RF);
b) apresentar a segmentação do rato utilizando a técnica de limiarização (RF);
c) realizar o reconhecimento das patas traseiras da imagem (RF);
d) estabelecer a distância (largura) entre o primeiro e quinto artelhos das patas traseiras (RF);
e) estabelecer o comprimento das pegadas das patas traseiras (RF);
f) utilizar a linguagem de programação Java para implementar o processamento das dimensões das patas (RNF);
g) utilizar a biblioteca JavaCV para realizar o processamento das imagens das pegadas dos ratos (RNF).

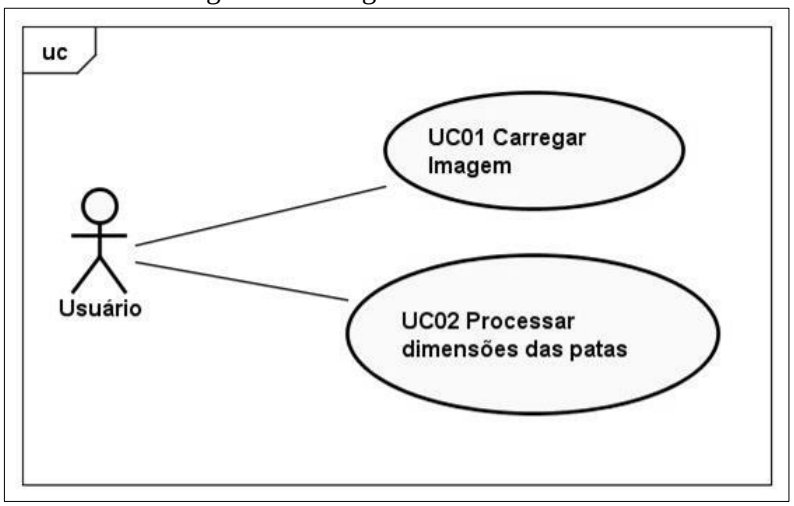
3.2 ESPECIFICAÇÃO

Esta seção apresenta os diagramas *Unified Modeling Language* (UML), utilizando a ferramenta Astah Community. A Seção 3.2.1 apresenta o diagrama de casos de uso. A Seção 3.2.2 descreve a classes criadas para o desenvolvimento do protótipo por meio de um diagrama de classes. Por fim, a Seção 3.2.3 apresenta dois diagramas de atividades que representam o fluxo de processamento do protótipo.

3.2.1 Diagrama de casos de uso

Nesta seção são apresentados os casos de uso do protótipo, que são ilustrados na Figura 13. Identificou-se apenas um ator, denominado `Usuário`, o qual utiliza todas as funcionalidades do protótipo.

Figura 13 - Diagrama de caso de uso

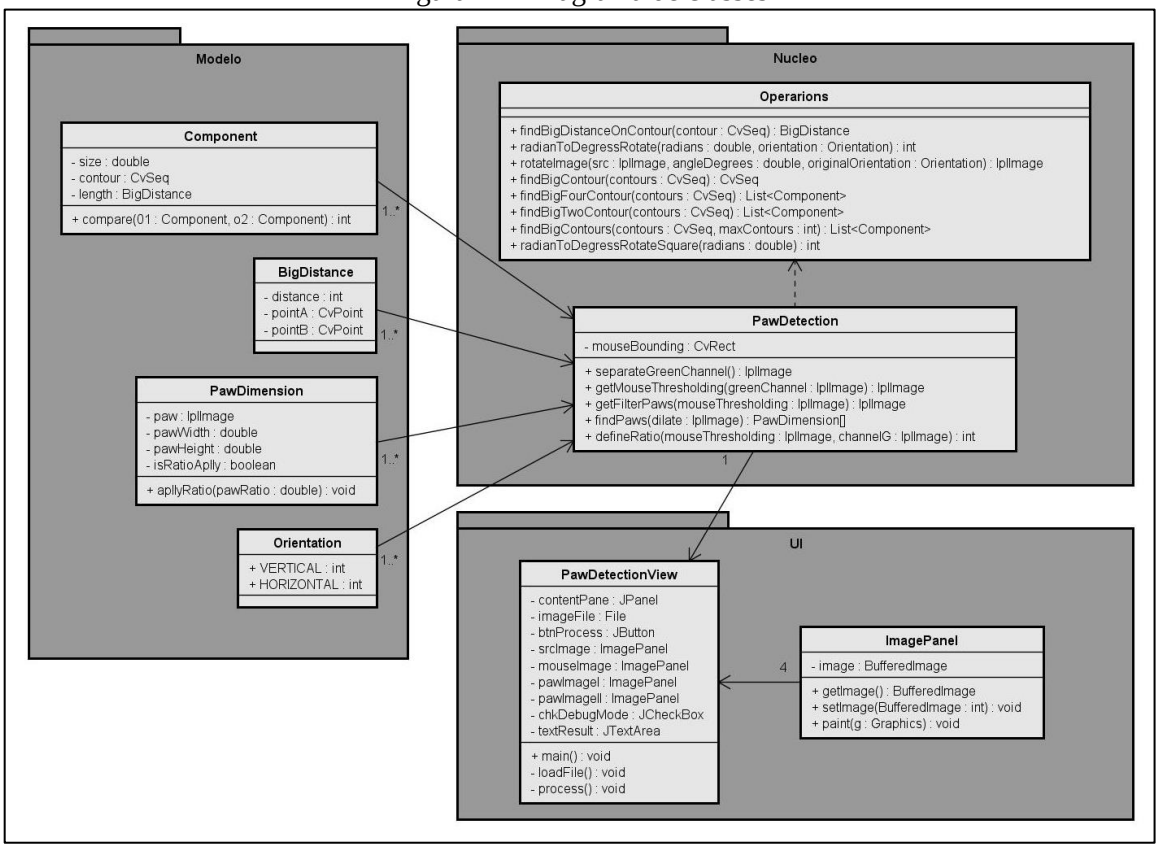


No `UC01 – Carregar Imagem` o protótipo deve exibir uma janela para que o usuário possa fazer a seleção da imagem do rato existente em seu computador. Após o carregamento da imagem, é executado o `UC02 – Processar dimensões das patas`, que é responsável por calcular as dimensões de altura e largura das patas do rato. O detalhamento dos casos de uso do protótipo está no Apêndice A.

3.2.2 Diagrama de Classes

Nesta seção é apresentado o diagrama de classes do protótipo. Para facilitar a visualização e o entendimento do relacionamento entre as classes, optou-se por agrupá-las em pacotes de acordo com sua especificidade. A Figura 14 exibe o diagrama de pacotes que compõem a protótipo, sendo eles: `modelo`, `núcleo` e *`User Interface`* `(UI)`. No pacote `modelo` encontram-se as classes que armazenam informações relevantes ao comprimento das patas e componentes de contornos. O pacote `núcleo` contém as principais funcionalidades do protótipo. Por fim, a camada de `UI` é responsável pela interação com o usuário.

Figura 14 - Diagrama de classes



Conforme pode ser visto no diagrama da Figura 14, o pacote `modelo` é composto por quatro classes: `Component`, `BigDistance`, `PawDimension` e `Orientation`. A classe `Component` tem a funcionalidade armazenar as informações utilizadas na manipulação de contornos. A `BigDistance` serviu para armazenar os dados dos cálculos de maior distância de um contorno. `PawDimension` agrupou as informações referentes ao cálculo das dimensões da pata. Por fim, a enumeração `Orientation` teve o propósito de indicar o alinhamento de um objeto.

O pacote `núcleo` possui as classes responsáveis por realizarem as principais funcionalidades do protótipo, sendo composto por apenas duas classes: `PawDetection` e `Operations`. A classe `PawDetection` possui as funcionalidades que realizam a detecção e o cálculo do tamanho das patas. Nela, têm-se os métodos: `separateGreenChannel`, responsável por separar as cores da imagem; `getMouseThresholding`, que busca a região do rato; `getFilterPaws`, que filtra as patas na imagem; `findPaws`, que localiza as patas; e `defineRatio`, que determina a escala da imagem.

A classe `Operations` encapsula os métodos secundários utilizados pela classe `PawDetection` no processo de detecção do tamanho da pata. Seus principais métodos são: `findBigDistanceOnContour`, que busca o maior contorno; `radianToDegressRotate`, para

descoberta do ângulo de rotação das patas; `radianToDegressRotateSquare`, que avalia o ângulo necessário para rotacionar o quadrado; e `otateImage`, que aplica a rotação. Outros métodos existentes são `findBigContour`, `findBigTwoContour`, `findBigFourContour` e `findBigContours` que fazem buscas nos maiores contornos da imagem.

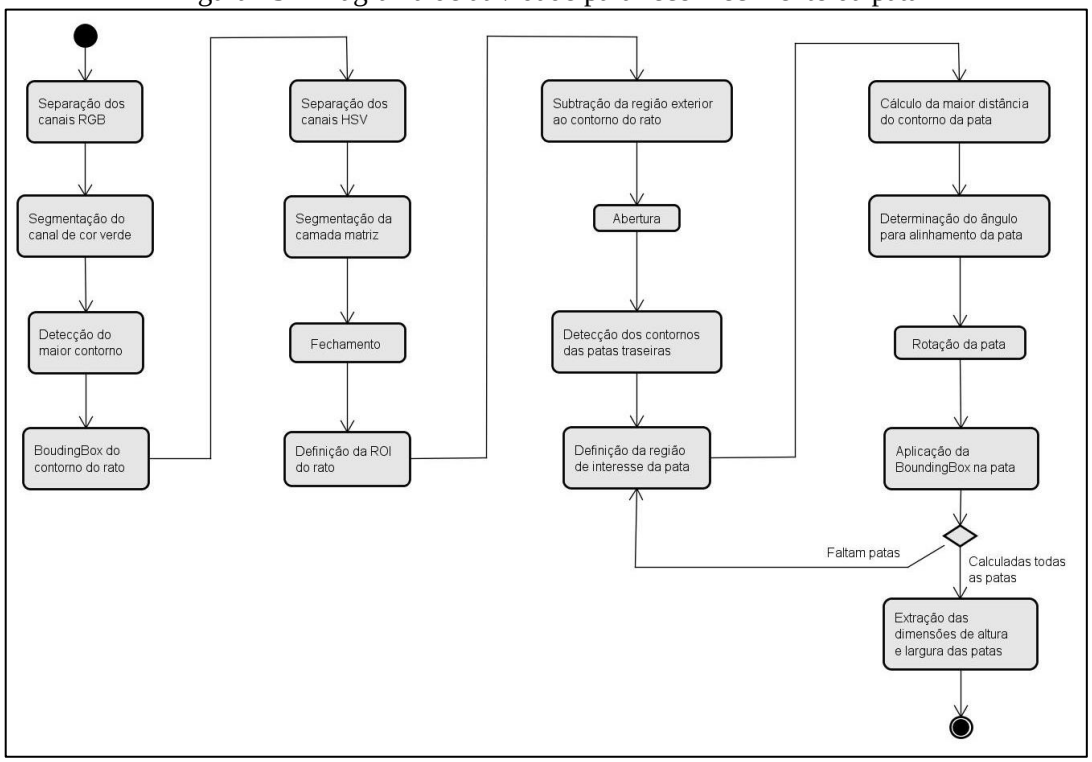
O terceiro pacote existente no diagrama contém as classes da interface com o usuário. A classe `PawDetectionView` representa a janela principal do protótipo, possuindo o menu superior, painéis para a imagem carregada, visualização da separação do corpo do rato e das patas traseiras e uma área de texto para mostrar os resultados obtidos no cálculo das dimensões das patas. A classe `ImagePanel` configurou os painéis de visualização do protótipo.

3.2.3 Diagramas de Atividades

O diagrama de atividades da Figura 15 mostra o fluxo de operação do protótipo para encontrar e definir as dimensões das patas do rato. Dessa forma, pode-se entender melhor a sequência das atividades realizadas pelo protótipo.

O primeiro passo se inicia com a separação dos canais de cores *Red Green Blue* (RGB) da imagem, sendo utilizado apenas o canal verde na etapa seguinte que é a limiarização. A partir da imagem segmentada, tenta-se encontrar o contorno correspondente ao corpo do rato, que na maioria das vezes é o maior contorno. Com o contorno do rato encontrado, determina-se uma `bounding box` para representar a localização do rato dentro da imagem.

Figura 15 - Diagrama de atividade para reconhecimento da pata



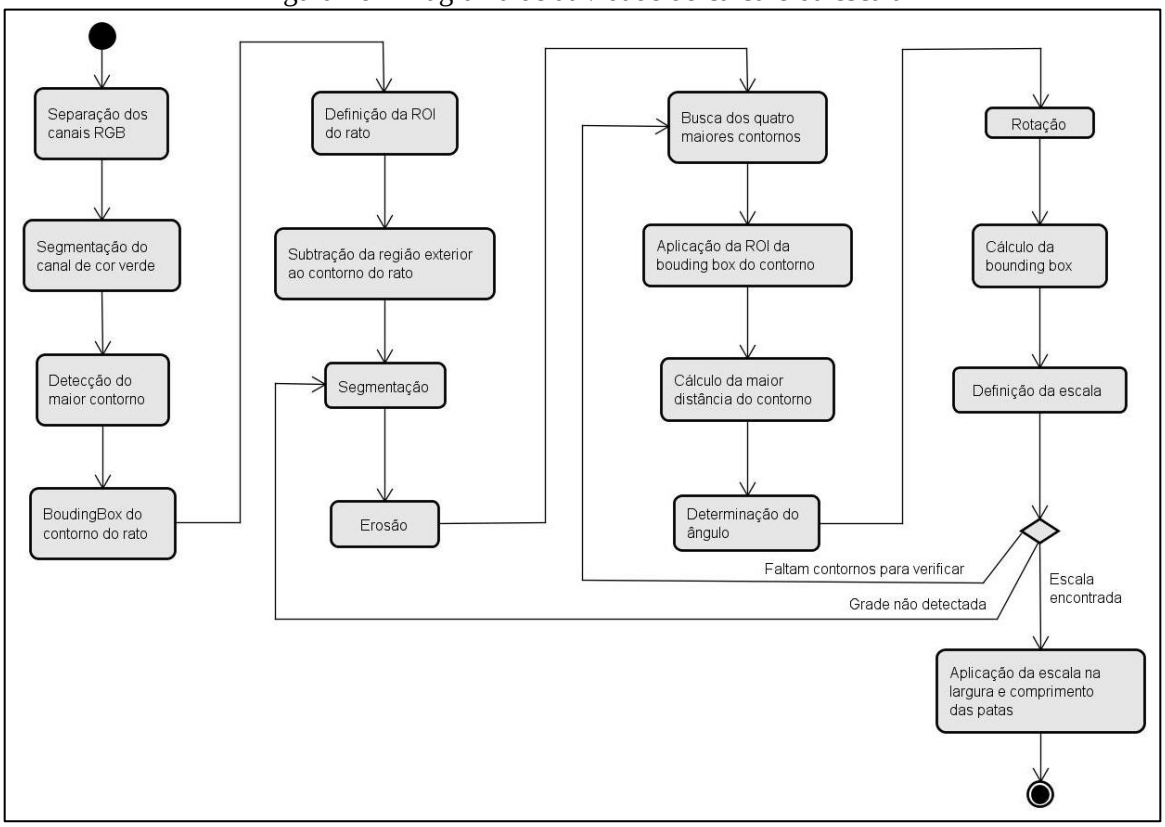
No passo seguinte, é iniciada a separação dos canais de cores do sistema *Hue Saturation Value* (HSV) com base na imagem original. Dessa separação, percebe-se que o canal de cores representando a matiz, realça a região das patas.

A partir desse ponto, a imagem possui as patas bem separadas, mas ainda com alguns ruídos. Esses ruídos acabaram sendo eliminados com aplicação da operação morfológica de abertura. Depois desta etapa, pode-se extrair os contornos e as regiões de interesse de cada uma das patas.

Para cada uma das patas, são repetidos os passos de cálculo da maior distância dentro do contorno, determinação do ângulo da pata e, por fim, a rotação da mesma. A partir disso, pode-se calcular a *bounding box* para obter as informações de altura e largura de cada pata.

Com as dimensões das patas traseiras calculadas, tenta-se definir o tamanho real delas. Na Figura 16 é apresentado o diagrama de atividades para a descoberta da escala da imagem. Inicialmente, são utilizados os mesmos passos do reconhecimento das patas, tais como a separação das camadas RGB, segmentação do canal verde, detecção do maior contorno e definição da *bounding box* do rato.

Figura 16 - Diagrama de atividade do cálculo da escala



Depois de estabelecer a *bounding box* que corresponde ao corpo do rato, ela foi utilizada na aplicação da ROI na imagem. Posteriormente, a etapa de subtração da região externa ao contorno do rato entra em um *loop*. No qual são realizadas a segmentação, erosão e busca dos quatro maiores contornos. Estes contornos representam os quadrados formados pelas linhas da grade existente na imagem.

Para cada contorno encontrado, aplica-se o seguinte conjunto de passos: definição da ROI pela *bounding box* do contorno, cálculo da maior distância, determinação do ângulo dos pontos da maior distância, rotação para alinhamento do quadrado, cálculo da *bouding box* mediante a rotação e a verificação do tamanho do quadrado para determinação da escala.

Caso o valor da escala não tenha sido obtido no passo anterior, o fluxo retorna ao início do *loop* após a subtração dos ruídos. Sendo repetidos esses passos até um limite máximo para o valor do limiar de segmentação da imagem. No fim, com a determinação da escala esse valor é aplicado junto às dimensões das patas encontradas, para então definir o tamanho das mesmas em centímetros.

3.3 IMPLEMENTAÇÃO

Neste capítulo são mostradas as técnicas, ferramentas e a operacionalidade da implementação. A Seção 3.3.1 apresenta o detalhamento das ferramentas e as técnicas utilizadas. A Seção 3.3.2 demonstra o processo operacional da construção do protótipo.

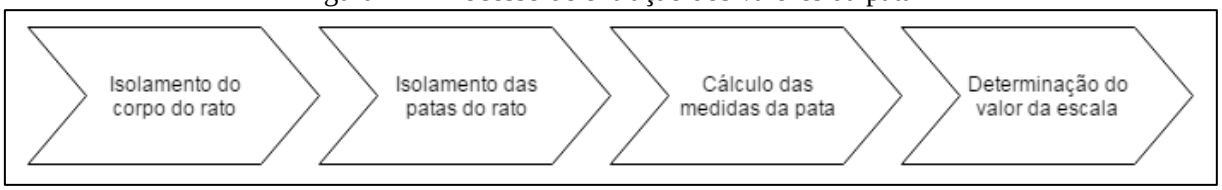
3.3.1 Técnicas e ferramentas utilizadas

A aplicação foi criada utilizando o ambiente de desenvolvimento Eclipse Luna com o *plug-in* Windown Builder para criação da interface gráfica, utilizando a linguagem de programação Java, na sua versão 8.

A tecnologia utilizada na implementação foi a biblioteca de visão computacional JavaCV. Ela disponibiliza as funções do OpenCV, feito em C/C++, para a linguagem Java, utilizando a tecnologia JNI presente no Java para acesso a DLL's nativas. Esta biblioteca facilita o carregamento de imagens, conversão para tons de cinza, separação de canais RGB, conversão para formato HSV, segmentação, operadores morfológicos, extração de contornos, separação da ROI, entre outras mais.

Conforme ilustra a Figura 17, o protótipo possui as seguintes etapas: isolamento do corpo do rato, isolamento das patas do rato, cálculo das medidas das patas, determinação do valor da escala.

Figura 17 - Processo de extração dos valores da pata



As próximas seções explicam o funcionamento completo de cada uma das etapas presentes na Figura 17. Sendo apresentada uma imagem de entrada, o código com a lógica de execução, a explicação do código e por fim, a imagem resultante.

3.3.1.1 Isolamento do corpo do rato

Esta etapa é responsável por receber uma imagem e procurar a região do corpo do rato. Para realizar esta identificação, foram realizados os seguintes passos: carregamento da imagem colorida, separação dos canais de cores RGB, segmentação do canal verde, detecção do maior contorno e cálculo da *bounding box*.

O primeiro passo da etapa de isolamento do corpo do rato é o carregamento da imagem em memória a partir do caminho informado pelo usuário. O Quadro 8 mostra que para abri-lo

é utilizado o método `cvLoadImage`.

Quadro 8 - Carga da imagem colorida

```
1. public static IplImage loadImage(String path) {
2.   File file =  new File(path);
3.   if (file.exists()) {
4.     return cvLoadImage(path, CV_LOAD_IMAGE_COLOR);
5.   }
6.   return null;
7. }
```

O resultado deste processamento pode ser visto na Figura 18. Nela, tem-se uma imagem do rato com um fundo escuro. Também, pode-se perceber a presença de uma grade, que será utilizada para calcular a escala da imagem.

Figura 18 - Amostra de imagem utilizada

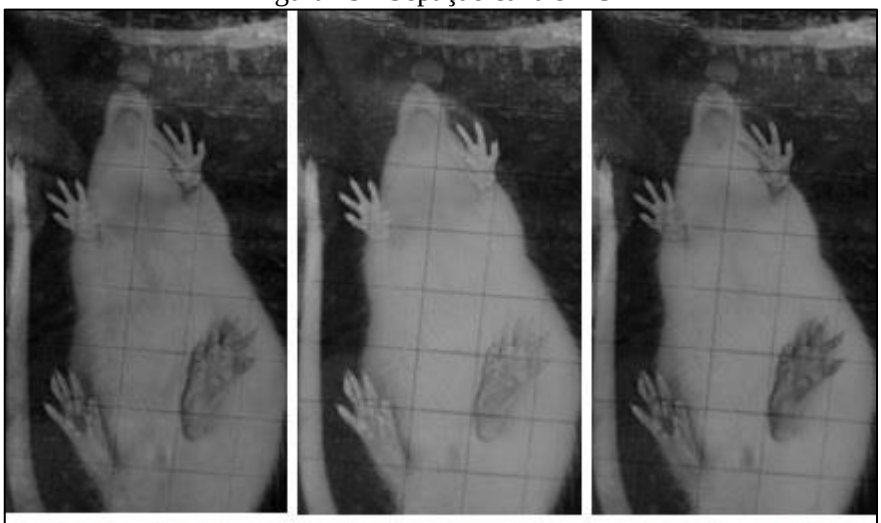
O Quadro 9 apresenta o trecho de código que separa cada um dos canais de cores da imagem. Por meio do método `cvCreateImage` são criadas três novas imagens, uma para cada canal RGB. O método `cvSplit` separa os canais de cores da imagem, criando novas imagens com os níveis de cinza de cada canal de cor.

Quadro 9 - Separação canais RGB

```
1. IplImage channelR = cvCreateImage(cvGetSize(src), IPL_DEPTH_8U, 1);
2. IplImage channelG = cvCreateImage(cvGetSize(src), IPL_DEPTH_8U, 1);
3. IplImage channelB = cvCreateImage(cvGetSize(src), IPL_DEPTH_8U, 1);
4. cvSplit(src, channelB, channelR, channelG, null);
```

Na Figura 19 é possível ver o resultado da separação dos canais RGB. A Figura 19(a) representa o canal azul, a Figura 19(b) o canal verde e a Figura 19(c) o canal vermelho.

Figura 19 - Sepação canais RGB



A partir da Figura 19, percebe-se que o canal verde representa o corpo do rato em tons mais claro do que o restante da imagem. Partindo desse pressuposto, realiza-se a segmentação da imagem, conforme mostra o código do Quadro 10.

Quadro 10 - Segmentação canal verde

```
1. CvScalar avg = cvAvg(greenChannel);
2. IplImage mouseThresholding =
              cvCreateImage(cvGetSize(greenChannel), 8, 1);
3. double thresh = avg.val(0);
4. cvThreshold(greenChannel, mouseThresholding, thresh, MAXVAL,
                                                    THRESH_BINARY);
```

O método `cvAvg` recebe a imagem correspondente ao canal verde e, retorna o valor da média das intensidades dos pixels da imagem. Este valor é utilizado pelo método `cvThreshold` para binarizar a imagem. A Figura 20 mostra o resultado da binarização a partir da segmentação do canal verde.

Figura 20 - Segmentação canal verde

A partir da Figura 20, pode-se perceber que o corpo do rato fica destacado do restante da imagem. Os únicos ruídos restantes são uma mancha no canto superior direito e o rabo na lateral esquerda. Também pode-se notar que parte as patas do rato ficam para fora do corpo. Porém, elas ainda pertencem ao mesma componente conexa. Contudo, pode-se considerar que a maior componente conexa é o corpo do rato. Os códigos dos Quadro 11 e Quadro 12 mostram o processo de busca pelos contornos na imagem segmentada.

Quadro 11 - Busca do contorno na imagem segmentada

```
1.  IplImage contourTemp =
                    cvCreateImage(cvGetSize(mouseThresholding), 8, 1);
2.  cvCopy(mouseThresholding, contourTemp);
3.  CvMemStorage storage = CvMemStorage.create();
4.  CvSeq contours = new CvContour(null);
5.  cvFindContours(contourTemp, storage, contours,
              Loader.sizeof(CvContour.class), CV_RETR_CCOMP,
              CV_CHAIN_APPROX_NONE, new CvPoint(0, 0));
6.  CvSeq bigContour = Operations.findBigContour(contours);
```

Para encontrar os contornos na imagem, primeiramente cria-se uma cópia da imagem segmentada e, posteriormente, utiliza-se o método `cvFindContours` para identificar todos os contornos existentes na imagem. O parâmetro `CV_RETR_CCOMP` serve para indicar o modo de extração dos contornos. A constante `CV_CHAIN_APPROX_NONE` indica o método que os pontos do contorno serão armazenados.

O Quadro 12 apresenta o método que identifica o maior contorno existente em uma sequência de contornos. A classe `CvSeq` funciona como uma sequência de ponteiros que apontam para a próxima posição, semelhante ao funcionamento de uma lista encadeada. A variável `ptr` armazena a referência para o contorno atual, a `bigArea` armazena a área do maior contorno, enquanto a `bigContour` guarda a referência do maior contorno encontrado.

Quadro 12 - Método de busca do maior contorno

```
1. public static CvSeq findBigContour(CvSeq contours) {
2.         CvSeq ptr = new CvSeq();
3.         double bigArea = 0;
4.         CvSeq bigContour = null;
5.         for (ptr = contours; ptr != null && ptr.address() != 0; ptr =
                                                          ptr.h_next()) {
6.             double size = cvContourArea(ptr);
7.             if (size > bigArea) {
8.                  bigArea = size;
9.                  bigContour = ptr;
10.            }
11.        }
12.        return bigContour;
13.}
```

Para cada contorno é calculada a área através do método `cvContourArea`. A área do contorno será comparada para verificar se ela representa o maior contorno, alterando o valor

do tamanho (`bigArea`) e a referência do maior contorno (`bigContour`), caso seja necessário. Ao final, o método retornará o maior contorno encontrado.

Ao encontrar o contorno do corpo do rato, tenta-se reduzir a área a ser processada. Para isso, define-se uma *bounding box* com base no conjunto de pontos que compõem o contorno. O Quadro 13 demonstra a aplicação do método `cvBoundingRect` para extração do retângulo que corresponde aos limites da `bounding box` e, consecutivamente onde o rato se localiza.

Quadro 13 - Extração de bounding box do rato

```
1.  CvRect mouseBounding = cvBoundingRect(bigContour);
```

A imagem representada na Figura 21 demonstra, com um retângulo vermelho, a `bounding box` que delimita o corpo do rato. Nela, também é possível perceber que a região foi restringida a área onde o rato se encontra.

Figura 21 - Bounding box corpo do rato

3.3.1.2 Isolamento das patas do rato

Se a separação dos canais RGB foi extremamente útil para determinar a área correspondente ao rato. Desta etapa, é realizada a separação dos canais utilizando o sistema de cores HSV (matiz, saturação e brilho), cujo intuito é identificar a região onde se encontram as patas do rato.

Para realizar a conversão do sistema de cores RGB para o HSV foi utilizado o método `cvCvtColor`. Este método recebe como parâmetro a imagem original (`src`), a imagem de destino (`hsv`) e o código que identifica o tipo de conversão a ser realizada (`COLOR_BGR2HSV`), conforme mostra o Quadro 14.

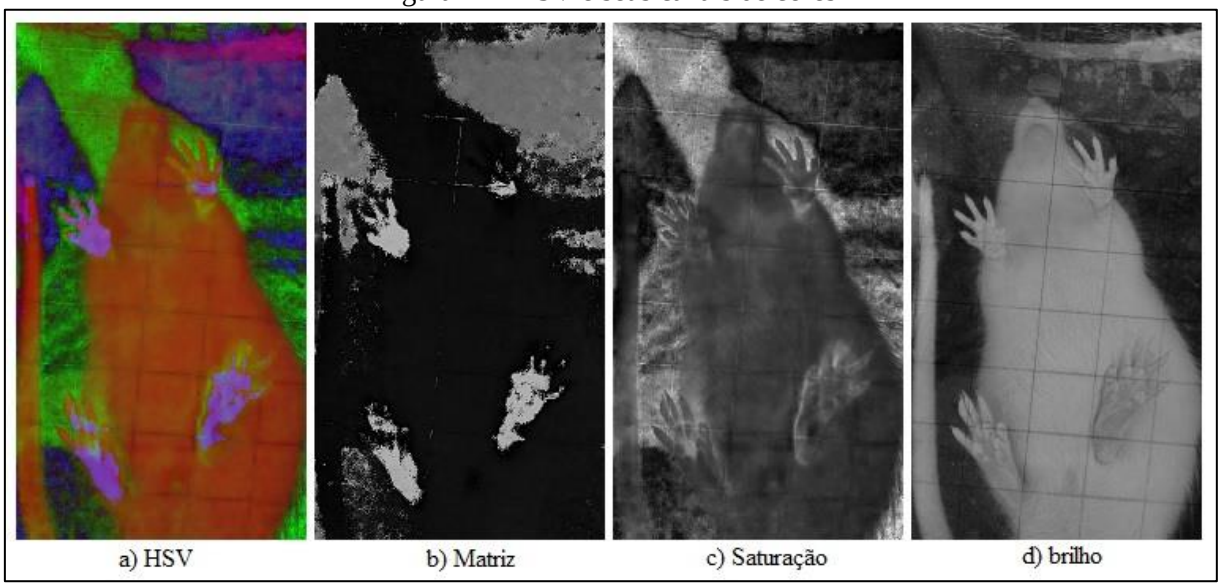
Quadro 14 - Conversão HSV e separação canais

```
1.  IplImage hsv = cvCreateImage(cvGetSize(src), 8, 3);
2.  cvCvtColor(src, hsv, COLOR_BGR2HSV);
3.
4.  IplImage h = cvCreateImage(cvGetSize(src), IPL_DEPTH_8U, 1);
5.  IplImage s = cvCreateImage(cvGetSize(src), IPL_DEPTH_8U, 1);
6.  IplImage v = cvCreateImage(cvGetSize(src), IPL_DEPTH_8U, 1);
7.  cvSplit(hsv, h, s, v, null);
```

Após realizar a conversão para o formato HSV, o método `cvSplit` é utilizado para separar os 3 canais. Para isso, foram criadas 3 imagens uma para cada canal, a matiz foi armazenada na variável `h`, a saturação em `s` e por fim o brilho na variável `v`.

A Figura 22(a) mostra a imagem convertida para o formato HSV, onde a Figura 22(b) mostra o canal matiz, a Figura 22(c) o canal saturação e a Figura 22(d) o canal brilho.

Figura 22 - HSV e seus canais de cores



A partir da separação dos canais, é possível observar que o canal matiz realça melhor as patas do rato. Também, pode-se receber que os ruídos que podem atrapalhar o processo de localização das patas ficam, em sua grande maioria, concentrados na região circundante ao rato.

Outra constatação importante é que as patas possuem um tom mais claro que o restante da imagem. Esta característica permite realizar uma segmentação por limiarização, em que, o valor do limiar é definido mediante análise da faixa de tons das cores existentes nas regiões das patas.

O trecho de código do Quadro 15 demonstra o processo de limiarização das patas. A constante `POW_THRESHOLDING` define o valor do limiar para a separação dos segmentos da amostra, enquanto a constante `MAXVAL` define o valor máximo que os pixels com valor superior ao limiar irão receber. Já o método `cvThreshold`, realiza a binarização da imagem.

Nele, o parâmetro `THRESH_BINARY` é o responsável por definir que os pixels com valor superior ao limiar receberam o valor máximo, ou seja, serão considerados como pixels brancos. Os abaixo do limiar, foram considerados pixels de cor preta.

Quadro 15 - Segmentação canal matiz

```
1.private static final int POW_THRESHOLDING = 150;
2.private static final double MAXVAL = 255;

3.IplImage pawThresholding = cvCreateImage(cvGetSize(src), 8, 1);
4.cvThreshold(h, pawThresholding, POW_THRESHOLDING, MAXVAL, THRESH_BINARY);
```

A Figura 23 mostra o resultado da limiarização, em que as patas e alguns ruídos externos ao rato também ficam destacados. As patas traseiras representam as maiores áreas brancas. Também pode-se perceber que a imagem ainda possui alguns ruídos e que algumas partes das patas acabaram ficando separadas, principalmente as áreas dos dedos. Para tentar agrupar os pedaços desconexos e eliminar os ruídos, foi aplicada a operação morfológica de fechamento. Esta operação consiste em aplicar operações morfológicas de dilatação e em seguida a de erosão. Após a aplicação da dilatação, algumas regiões que estavam separadas se juntaram devido ao crescimento de suas regiões. Já a aplicação da erosão, com o mesmo elemento estruturante utilizado na dilatação, reestabeleceu o tamanho original das patas.

Figura 23 - Limiarização patas

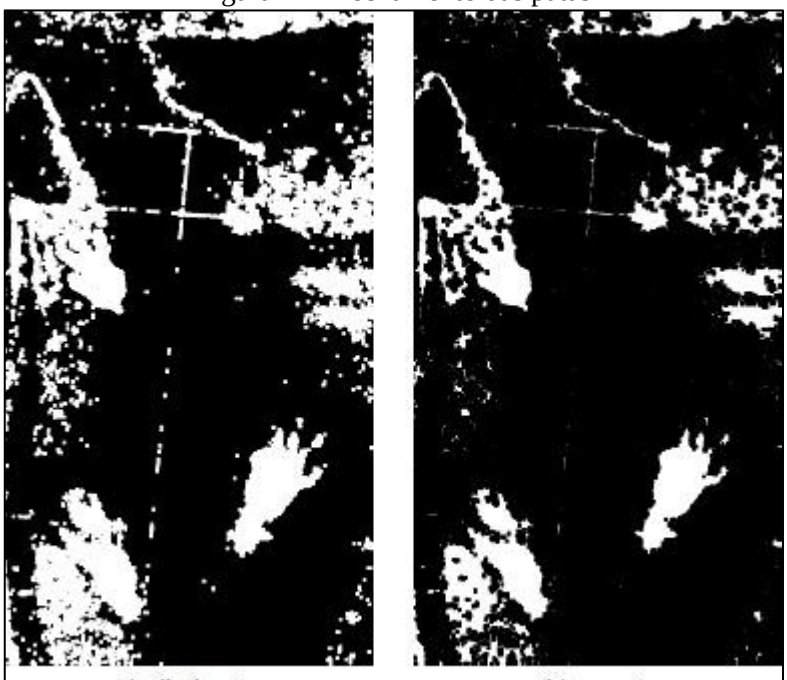
O Quadro 16 mostra o código responsável por aplicar as operações de dilatação e erosão.

Quadro 16 - Fechamento patas

```
1. IplConvKernel element = cvCreateStructuringElementEx(17, 17, 0, 0,
                                                        MORPH_RECT);
2.  IplImage dilate = cvCreateImage(cvGetSize(pawThresholding), 8, 1);
3.  IplImage erode = cvCreateImage(cvGetSize(pawThresholding), 8, 1);
4.  cvDilate(pawThresholding, dilate, element, 1);
5.  cvErode(dilate, erode, element, 1);
```

O método `cvCreateStructuringElementEx` cria o elemento estruturante a ser utilizado na convolução das operações morfológicas. O tamanho utilizado como elemento foi 17. Após definir o tamanho do elemento estruturante, são criadas duas novas imagens que receberem os resultados das operações morfológicas. Para aplicação da dilatação sobre a imagem das patas segmentadas foi utilizado o método `cvDilate`. Por fim, foi aplicada a operação morfológica de erosão na imagem dilatada, utilizando o método `cvErode`. A Figura 24 mostra os resultados obtidos ao aplicar a operação morfológica de fechamento. A Figura 24a mostra o resultado da operação de dilatação e a Figura 24b o resultado da operação de erosão.

Figura 24 - Fechamento das patas



A partir da Figura 24, pode-se perceber que ainda existem ruídos externos ao corpo do rato. Para removê-los, realizou-se a subtração da região correspondente a área do corpo do rato, conforme mostra o código do Quadro 17.

Quadro 17 - Subtração do corpo do rato

```
1. cvSetImageROI(erode, mouseBounding);
2. cvSetImageROI(mouseThresholding, mouseBounding);
3. IplImage sub = cvCreateImage(cvGetSize(erode), 8, 1);
4. cvNot(erode, erode);
5. cvSub(mouseThresholding, erode, sub);
```

Antes de fazer a subtração, a imagem teve sua cor invertida através do método `cvNot`. Essa inversão foi necessária para que restassem apenas as regiões das patas. A Figura 25 apresenta o resultado da inversão das cores.

Figura 25 - Cores das patas invertidas

A Figura 26 mostra o resultado da subtração, onde apenas as regiões pretas continuaram presentes. Nela, também percebe-se que restaram apenas as patas e alguns ruídos dentro da região do corpo do rato.

Figura 26 - Subtração das patas

Para limpar os ruídos internos, foi aplicada a operação morfológica de abertura, que é caracterizada pela aplicação da operação de erosão seguida de uma dilatação. Ambas utilizando o mesmo elemento estruturante. O Quadro 18 mostra o trecho de código responsável pela operação de abertura onde, utilizou-se o método `cvErode` sobre a imagem resultante da subtração anterior e, posteriormente o método `cvDilate` sobre a imagem erodida.

Quadro 18 - Abertura patas

```
1. erode = cvCreateImage(cvGetSize(sub), 8, 1);
2. cvErode(sub, erode, element, 1);
3.
4. dilate = cvCreateImage(cvGetSize(sub), 8, 1);
5. cvDilate(erode, dilate, element, 1);
```

Na Figura 27 pode-se verificar que a aplicação da operação de abertura removeu com sucesso os ruídos que estavam dentro da região do corpo do rato, preservando apenas as patas.

Figura 27 - Abertura Patas

Com as patas do rato bem destacadas, o próximo passo é encontrar os contornos correspondem às patas traseiras, que são os maiores elementos presentes na imagem. Nos Quadro 19 e Quadro 20 são apresentados os códigos responsáveis por detectar os dois maiores contornos.

Quadro 19- Detecção dos contornos patas

```
1. contourTemp = cvCreateImage(cvGetSize(dilate), 8, 1);
2. cvCopy(dilate, contourTemp);
3. storage = CvMemStorage.create();
4. contours = new CvContour(null);
5. cvFindContours(contourTemp, storage, contours,
                  Loader.sizeof(CvContour.class), CV_RETR_CCOMP,
                  CV_CHAIN_APPROX_NONE, new CvPoint(0, 0));
6.
7. List<Component> paws = findBigTwoContour(contours);
8. public static List<Component> findBigTwoContour(CvSeq contours) {
9.    return findBigContours(contours, 2);
10.}
```

O método `cvFindContours` realiza a busca dos contornos na imagem. Seu parâmetro `CV_RETR_CCOMP` serve para indicar o modo de extração dos contornos. A constante `CV_CHAIN_APPROX_NONE` indica que todos os pontos do contorno são retornados, sem nenhum tipo de compressão. O último valor indica que o *offset* inicial para inicio da varredura na imagem, começa no seu ponto de origem. No Quadro 20 encontra-se o código que busca os maiores contornos.

Quadro 20 - Busca dos maiores contornos

```
1. public static List<Component> findBigContours(CvSeq contours, int
                                                          maxContours) {
2.    CvSeq ptr = null;
3.    List<Component> bigPawList = new ArrayList<Component>();
4.    for (ptr = contours; ptr != null; ptr = ptr.h_next()) {
5.        double size = cvContourArea(ptr);
6.        Component newPaw = new Component(size, ptr);
7.        boolean isBig = bigPawList.size() == 0;
8.        for (Component paw : bigPawList) {
9.            if (newPaw.getSize() > paw.getSize()) {
10.               isBig = true;
11.           }
12.       }
13.       if (isBig) {
14.           if (bigPawList.size() >= maxContours) {
15.               bigPawList.remove(maxContours - 1);
16.           }
17.           bigPawList.add(newPaw);
18.           bigPawList.sort(newPaw);
19.       }
20.   }
21.   return bigPawList;
22. }
```

O método `findBigContours` recebe a sequência inteira de contornos encontrados e quantos serão considerados (maiores contornos). A lista `bigPawList` armazena os maiores contornos encontrados na varredura de contornos. Na linha 5 é calculado o tamanho dos contornos através do método `cvContourArea`. Na linha 6, é criado o objeto `newPaw` para armazenar o tamanho e o contorno das maiores patas encontradas. A variável `isBig`, controla se o contorno analisado ficou entre os maiores contornos que serão considerados.

Caso o contorno analisado seja um dos maiores, o menor deles será removido da lista. Quando um novo contorno é adicionado lista, ela é reordenada de forma decrescente pelo tamanho dos contornos. Ao término da análise de todos os contornos, a lista com os maiores contornos é então retornada. Na Figura 28 são destacadas as regiões dos dois maiores contornos encontrados. Ao qual, coloridos em vermelho para facilitar sua visualização.

Figura 28 - Destaque patas traseiras

3.3.1.3 Cálculo das medidas das patas

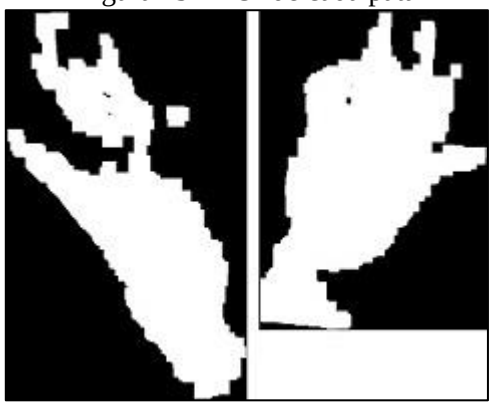
Esta etapa é responsável por estimar a altura e largura das duas patas traseiras. Para realizar a extração dessas medidas, criou-se uma *bounding box* a partir dos limites do contorno da pata. Se fossem utilizados os contornos encontrados anteriormente, as medidas das patas não seriam verdadeiras. Isso se deve ao fato delas não estarem bem alinhadas verticalmente, o que resultaria em falsos valores de altura e largura, conforme apresenta o Quadro 21.

Quadro 21 - Região de interesse da pata

```
1. IplImage pata = cvCreateImage(cvGetSize(dilate), 8, 1);
2. cvCopy(dilate, pata);
3. CvRect boundingBox = paw.getBoundingBox();
4.
5. int xBBox = boundingBox.x();
6. int yBBox = boundingBox.y();
7. CvPoint offesetPoint = new CvPoint(xBBox, yBBox);
8. CvPoint endPoint = new CvPoint(xBBox + boundingBox.width(), yBBox +
                                  boundingBox.height());
9. cvRectangle( pata, offesetPoint, endPoint,
                        CvScalar.BLACK, 2, CV_AA, 0);
10.cvSetImageROI(pata, boundingBox);
```

A Figura 29 mostra o resultado da imagem das patas após a aplicação da ROI. Cada uma das patas foi isolada, mas ainda não estão alinhadas verticalmente para realização do cálculo da altura e largura.

Figura 29 - ROI de cada pata



Para realizar o alinhamento das patas, primeiramente é necessário descobrir a orientação delas. Para encontrar a orientação é verificada a maior distância entre os pontos que a compõem o contorno da pata. Esta distância é estabelecida entre o calcanhar e a ponta do terceiro dedo. O Quadro 22 apresenta o código que busca a maior distância de cada pata. Para isso, é extraído o contorno dela. Esse contorno é utilizado como entrada para o método `findBigDistanceOnContour`, que calcula a maior distância. Nesse método, existe o objeto `pawLength`, que armazena as informações de comprimento além dos pontos de origem e destino da reta que forma a maior distância do contorno.

Quadro 22 - Busca da maior distância da pata

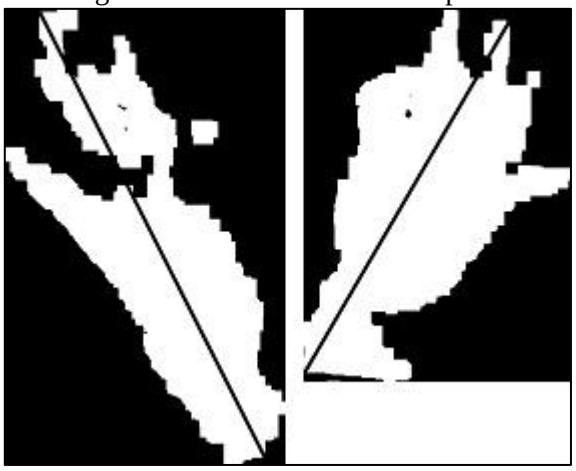
```
1. CvSeq contour = paw.getContour();
2. BigDistance pawLength = findBigDistanceOnContour(contour);
3.
4. public static BigDistance findBigDistanceOnContour(CvSeq contour) {
5.    BigDistance pawLength = null;
6.    int countourSize = contour.total();
7.    for (int j = 0; j < countourSize; j++) {
8.       CvPoint pointA = new CvPoint(cvGetSeqElem(contour, j));
9.       for (int k = j; k < countourSize; k++) {
10.         CvPoint pointB = new CvPoint(cvGetSeqElem(contour, k));
11.
12.         double distance = Math.sqrt(Math.pow(pointA.x() - pointB.x(),
                                2)+ Math.pow(pointA.y() - pointB.y(), 2));
13.         if (pawLength == null || distance > pawLength.getDistance()) {
14.               pawLength = new BigDistance(distance, pointA, pointB);
15.         }
16.      }
17.   }
18.   return pawLength;
19.}
```

Na linha 6 foi utilizado o método `total()` para descobrir a quantidade de contornos existentes. Esse valor é utilizado nos dois blocos `for` para navegação entre todos os pontos

distintos do contorno. O primeiro `for` busca um ponto nomeado de `pointA`, enquanto o segundo `for` busca um segundo ponto nomeado de `pointB`. Com os dois pontos encontrados, é feito o cálculo da distância euclidiana entre eles. Caso esteja na primeira interação ou a nova distância seja maior do que a já encontrada previamente, essas novas informações são armazenadas no objeto `pawLength`. Ao final da execução do método `findBigDistanceOnContour` essa informação é retornada.

Como pode ser visto na Figura 30, os dois pontos mais distantes formam uma linha que cruza a pata. Com base nessa reta, é obtido o valor da altura da pata, a partir do qual pode-se calcular o ângulo necessário para alinhar verticalmente a imagem.

Figura 30 - Maior distância das patas



No Quadro 23, encontra-se o a fórmula para descobrir o ângulo entre os dois pontos da reta. Primeiramente, é calculada a diferença entre os valores de *x* dos dois pontos da reta, depois é feito o mesmo para os valores de *y*. Então, é chamado o método `atan2` para calcular o arco da tangente da divisão da diferença de y pela diferença de y. O valor retornado está no formato de radianos.

Quadro 23 - Ângulos entre os pontos

```
1. double x = pawLength.getPointA().x() - pawLength.getPointB().x();
2. double y = pawLength.getPointA().y() - pawLength.getPointB().y();
3. double theta = Math.atan2(y, x);
4. Orientation orientation = Math.abs(y) > Math.abs(x) ?
                        Orientation.VERTICAL : Orientation.HORIZONTAL;
5. double angle = radianToDegressRotate(theta, orientation);
```

Depois de encontrar o valor do ângulo entre os dois pontos, é necessário descobrir o valor da rotação a ser aplicada para alinhar a pata verticalmente. Primeiramente, é descoberto se a pata encontrava-se próxima de um alinhamento horizontal ou vertical. Para isso, foram comparados os valores absolutos das diferenças entre *y* e *x*, sendo que para descobrir o valor necessário dessa rotação foi criada a função `radianToDegressRotate`. Nesse método, são

informados como valores de entrada o ângulo dos pontos em radianos e a orientação do posicionamento das patas.

No Quadro 24 encontra-se o código responsável por calcular o valor da rotação em graus para o alinhamento da pata. O valor é convertido de radianos para graus, sendo armazenado o resultado na variável `degress`. Caso a orientação tenha sido mais vertical, foi calculado diretamente o valor necessário para o alinhamento vertical. No caso de uma orientação mais horizontal, primeiro se descoberto o valor para realizar o alinhamento horizontal e posteriormente, para fazer o alinhamento vertical. No retorno do método tem-se o valor em graus para alinhar verticalmente a imagem.

Quadro 24 - Ângulo para rotação da pata

```
1. private static int radianToDegressRotate(double radians, Orientation
orientation) {
2.    int degress = (int) (radians * (180 / Math.PI));
3.    if (orientation == Orientation.VERTICAL) {
4.          if (degress > 90) {
5.                return 90 - degress;
6.          } else {
7.                return 270 - degress;
8.          }
9.    } else {
10.         return (180 - degress) - 90;
11.   }
12.}
```

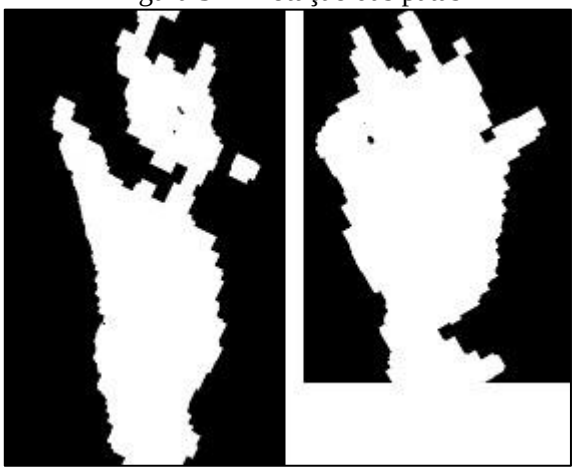
Após descobrir o valor de rotação para fazer o alinhamento da imagem, o próximo passo é aplicar rotação. O Quadro 25 apresenta o código responsável para realizar essa rotação na imagem.

Quadro 25 - Rotação da pata

```
1. public static IplImage rotateImage(IplImage src, double angleDegrees,
Orientation originalOrientation) {
2.    CvMat M = CvMat.create(2, 3, CV_32F);
3.    int w = src.roi().width();
4.    int h = src.roi().height();
5.    double angleRadians = angleDegrees * (Math.PI / 180.0f);
7.    M.put(0, (float) (Math.cos(angleRadians)));
8.    M.put(1, (float) (Math.sin(angleRadians)));
9.    M.put(2, w * 0.5f);
10.   M.put(3, -(float) (Math.sin(angleRadians)));
11.   M.put(4, (float) (Math.cos(angleRadians)));
12.   M.put(5, h * 0.5f);
13.
14.   CvSize sizeRotated = new CvSize();
15.   if (originalOrientation == Orientation.VERTICAL) {
16.         sizeRotated.width(Math.round(w));
17.         sizeRotated.height(Math.round(h));
18.   } else {
19.         sizeRotated.width(Math.round(h));
20.         sizeRotated.height(Math.round(w));
21.   }
23.   IplImage imageRotated = cvCreateImage(sizeRotated, src.depth(),
                                            src.nChannels());
24.   cvGetQuadrangleSubPix(src, imageRotated, M);
26.   return imageRotated;
27.}
```

O método `cvGetQuadrangleSubPix` realiza a rotação da imagem inicial utilizando a matriz de transformação (`M`). O resultado da rotação é armazenado na imagem `imageRotated`, sendo retornada pelo método. Na Figura 31 são apresentados os resultados da rotação das patas.

Figura 31 - Rotação das patas



Com as imagens das patas alinhadas verticalmente, é possível verificar a largura das patas, conforme mostra o Quadro 26.

Quadro 26 - Largura da pata

```
1. storage = CvMemStorage.create();
2. contours = new CvSeq(null);
3. contourTemp = cvCreateImage(cvGetSize(rotatedPaw), 8, 1);
4. cvCopy(rotatedPaw, contourTemp);
5. cvFindContours(contourTemp, storage, contours,
      Loader.sizeof(CvContour.class), CV_RETR_CCOMP,
      CV_CHAIN_APPROX_NONE, new CvPoint(0, 0));
6. CvSeq alignedPaw = findBigContour(contours);
7. int pawWidth = cvBoundingRect(alignedPaw).width();
```

Primeiramente, é feita uma busca pelos contornos da pata utilizando o método `cvFindContours`. Na linha 6 é feita a busca pelo maior contorno. Na linha 7 é estabelecida a *bounding box* da pata através do método `cvBoundingRect`, em que as distâncias entre as extremidades representam a largura e altura da pata do rato. Porém, estes valores estão em pixels, sendo necessário realizar a conversão para centímetros.

3.3.1.4 Determinação do valor da escala

Para descobrir o tamanho real das patas foi necessário realizar a medição da escala da imagem. Para auxiliar desta tarefa, as imagens coletadas possuem uma grade com linhas espaçadas por 2cm. Esse valor é tanto para a distância vertical quanto para a horizontal.

Para descobrir o valor da escala, foram verificados quantos pixels separavam uma linha da outra. Porém, as linhas tinham pouco destaque na imagem, o que acabou dificultando o reconhecimento das mesmas ao calcular as distâncias em pixels. Para estabelecer a quantidade de pixels entre 2 linhas, foi aplicada uma operação de erosão na imagem para que as regiões do rato fossem separadas mais claramente pelas linhas. As linhas destacadas pela erosão formam os quadrados utilizados para obter o valor da relação da escala.

Primeiramente, retirou-se os ruídos externos ao corpo do rato, para ressaltar e facilitar o reconhecimento dos quadrados formados pelas linhas da amostra. No Quadro 27 é apresentado o código que realiza a filtragem dos ruídos externos da imagem pela subtração das imagens.

Quadro 27 - Subtração ruídos escala

```
1. IplImage filtered = cvCreateImage(cvGetSize(mouseThresholding), 8, 1);
2. cvNot(mouseThresholding, mouseThresholding);
3. cvSub(channelG, mouseThresholding, filtered);
```

A Figura 32 mostra o resultado da subtração dos ruídos externos ao corpo do rato. Nela, as linhas do rato possuem um pequeno destaque mais escuro do que o branco que representa o rato.

Figura 32 - Filtragem ruídos externo do rato

Após a eliminação dos ruídos, foi possível aplicar uma limiarização para começar a isolar os quadrados (grade) existentes na imagem. Inicialmente, foi definido um valor base para o limiar, que é incrementado a cada interação ao qual não foi possível estabelecer o valor da proporção. Isso foi feito, pois nem todas as vezes o limiar definido conseguia separar com sucesso os quadrados de uma imagem. O Quadro 28 demonstra o processo da iteração de incremento do limiar. O valor inicial definido para o limiar foi de `129`, incrementando em `5` a cada nova interação. O critério para sair dessa interação é encontrar a escala definida na variável `pixelsForSquare` ou o valor do limiar ultrapassar `160`. A cada interação aplicou-se uma segmentação. A Figura 33 mostra o resultado deste processo.

Quadro 28- Limiarização linhas da grade

```
1. for (thresh = 129; pixelsForSquare == 0 || thresh > 160; thresh += 5) {
2.     IplImage filteredThresh = cvCreateImage(cvGetSize(filtered), 8, 1);
3.     cvThreshold(filtered, filteredThresholding, thresh, MAXVAL,
4.     THRESH_BINARY);
5.     ...
6. }
```

Figura 33 - Limiarização das linhas da grade

Após a limiarização, ainda existem regiões em que as linhas estão muito estreitas ou até mesmo apagadas. Para melhorar a espessura das linhas que delimitam os quadrados, foi aplicada a operação morfológica de erosão. O Quadro 29 demonstra o código que aplica a erosão na imagem. O valor do elemento estruturante utilizado na erosão é 7.

Quadro 29 - Aplicação de erosão nas linhas da grade

```
1. IplConvKernel element = cvCreateStructuringElementEx(7, 7, 0, 0,
MORPH_RECT);
2. IplImage erode = cvCreateImage(cvGetSize(filteredThresh), 8, 1);
3. cvErode(filteredThresh, erode, element, 1);
```

A Figura 34 apresenta o resultado da aplicação da operação de erosão que foi utilizada para aumentar a espessura das linhas da grade. Pode-se perceber que os quadrados entre as linhas da grade tiveram uma melhora, tornando mais precisa a detecção dos contornos.

Figura 34 - Erosão das linhas da grade

Conforme mostra o Quadro 30, primeiro localiza-se o maior contorno na imagem a fim de se estabelecer a ROI onde o rato se encontra. Isso foi feito para limitar a região onde se faziam presentes as linhas da grade.

Quadro 30 - Isolamento das linhas no rato

```
1. cvFindContours(filteredTemp, storage, contours,
2. Loader.sizeof(CvContour.class), CV_RETR_CCOMP,
3. CV_CHAIN_APPROX_NONE, new CvPoint(0, 0));
4. CvSeq bigContour = MaiorComponenteSaturacao.findBigContour(contours);
5. CvRect mouseBounding = cvBoundingRect(bigContour);
6. cvSetImageROI(erode, mouseBounding);
```

Com a região de interesse identificada, o passo seguinte é buscar os contornos. Desta vez, conforme o Quadro 31, foram buscados os 4 maiores contornos, para garantir que se os quadrados tivessem sido isolados com sucesso, um ou dois deles serão identificados.

Quadro 31 - Busca dos quatro maiores componentes

```
1. cvFindContours(squares, storage, contours,
2. Loader.sizeof(CvContour.class), CV_RETR_CCOMP,
3. CV_CHAIN_APPROX_NONE, new CvPoint(0, 0));
4. List<Component> possibleSquares =
                  MaiorComponenteSaturacao.findBigFourContour(contours);
```

Com os 4 maiores contornos encontrados, surgiu à necessidade de descobrir se algum deles possuía lados iguais. Diante disso, precisava-se verificar se todos os lados tinham o mesmo tamanho, característica para serem classificados como quadrados. Para realizar o alinhamento dos objetos foi buscada a maior distância entre os pontos do contorno. O Quadro 32 apresenta o código que realiza a busca pela diagonal do quadrado. Para isso, itera-se sobre cada um dos quatro maiores contornos, verificando a maior distância entre os pontos. Essa informação foi utilizada para aplicar a rotação na região do quadrado.

Quadro 32 - Busca diagonal do quadrado

```
1. for (int i = 0, amount = possibleSquares.size(); i < amount; i++) {
2.    Component component = possibleSquares.get(i);
3.    CvSeq contour = component.getContour();
4.    BigDistance bigDistance =
                MaiorComponenteSaturacao.findBigDistanceOnContour(contour);
```

Descoberta a diagonal do possível quadrado, é preciso calcular o ângulo necessário para alinhá-lo em 45º. Com este ângulo, o quadrado fica com suas laterais paralelas aos limites da imagem. Assim, pode-se verificar se a *bonding box* do objeto possui os quatros lados iguais. No Quadro 33 é apresentado o código responsável por descobrir o ângulo de rotação do possível quadrado.

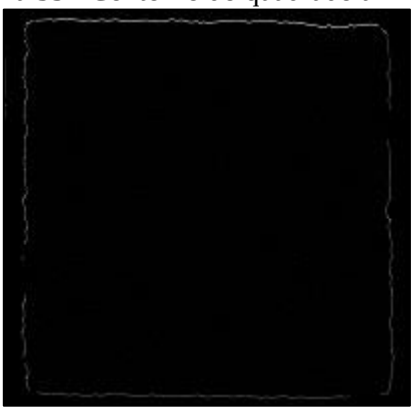
Quadro 33 - Alinhamento do possível quadrado

```
1. CvPoint pointA = bigDistance.getPointA();
2. CvPoint pointB = bigDistance.getPointB();
3. double x = pointA.x() - pointB.x();
4. double y = pointA.y() - pointB.y();
5. double theta = Math.atan2(y, x);
6. double angle = radianToDegressRotateSquare(theta);
7.
8. CvRect boundingBox = cvBoundingRect(contour);
9. cvRectangle(squares, new CvPoint(boundingBox.x(), boundingBox.y()), new
CvPoint(boundingBox.x() + boundingBox.width(), boundingBox.y() +
boundingBox.height()), CV_RGB(255, 0, 0), 2, CV_AA, 0);
10. cvSetImageROI(squares, boundingBox);
11. IplImage rotatedPossibleSquare =
MaiorComponenteSaturacao.rotateImage(squares, angle,
Orientation.VERTICAL);
```

O método `radianToDegressRotateSquare` busca o ângulo de rotação. Esse método recebe o ângulo em radianos e devolve o ângulo em graus para rotação de alinhamento da diagonal em 45º. Nas linhas 8 e 9 é definida uma *bounding box*, inserindo um retângulo nos limites da mesma para evitar a distorção da imagem ao ser rotacionada. Na linha 10 é definida a região de interesse do possível quadrado para então aplicar a rotação a partir do ângulo que alinha a diagonal em 45º.

A Figura 35 demonstra o resultado da rotação do quadrado. Com as suas laterais alinhadas, se torna possível extrair as medidas para confirmar se o objeto em questão é um quadrado ou não.

Figura 35 - Contorno do quadrado alinhado



Com a imagem do quadrado alinhada, busca-se as medidas para verificar se é um dos quadrados da grade. Se for, define-se o tamanho para a descoberta da escala. O Quadro 34 demonstra a definição da escala.

Quadro 34 - Definição escala

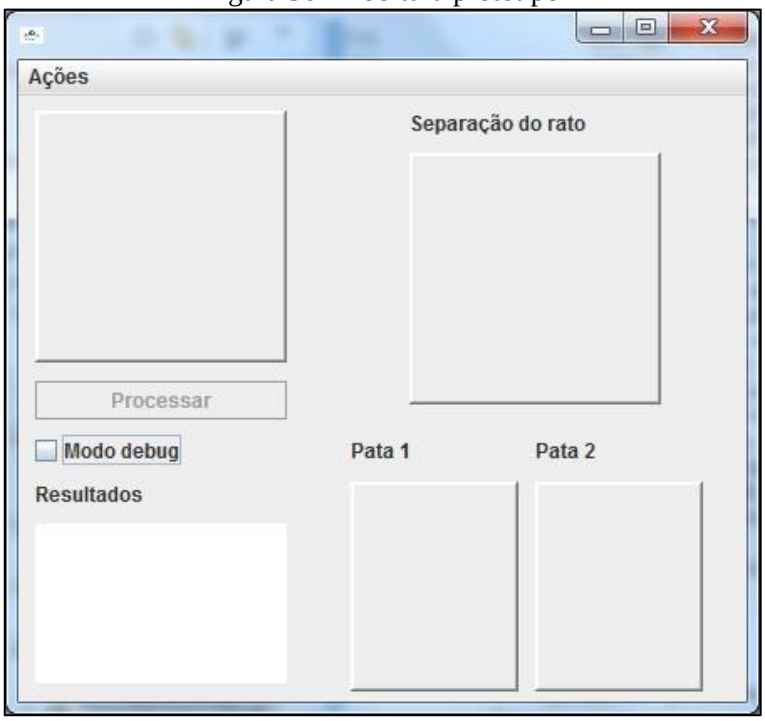
```
1. CvSeq square = MaiorComponenteSaturacao.findBigContour(contours);
2. if (square != null) {
3.    CvRect boudingSquare = cvBoundingRect(square);
4.    int width = boudingSquare.width();
5.    int height = boudingSquare.height();
6.    int relationDifference = Math.abs(width - height);
7.    int proportion = (width + height) / 2;
8.    if (relationDifference < 13 && size > (300 * 300)
                    && size < (500 * 500)) {
9.          if (pixelsForSquare == 0 || proportion < pixelsForSquare) {
10.               pixelsForSquare = proportion;
11.         }
12    }
13.}
```

Primeiramente, busca-se o maior contorno e se calcula o contorno da *bounding box*. Com essa informação, é verificado se as medidas dos lados são iguais, tendo uma tolerância de até 10 pixels. Por fim, a variável `pixelsForSquare` armazena a quantidade aproximada de pixels a cada 2 centímetros. Esta informação é utilizada para converter a altura e largura das patas de pixels para centímetros.

3.3.2 Operacionalidade da implementação

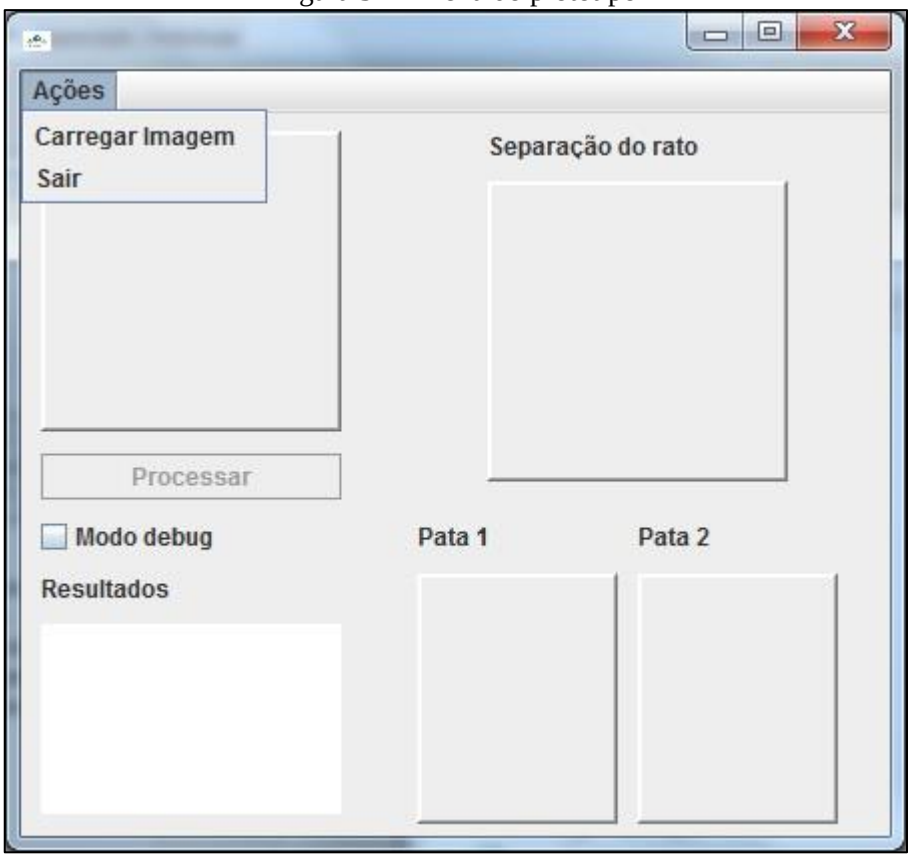
O protótipo possui um menu de ações, um painel no canto superior esquerdo para visualização da imagem escolhida. Abaixo desse painel, tem-se o botão para iniciar o processamento da imagem. Existe também a opção de `modo debug` que serve para a geração das imagens intermediárias do processamento no local de execução da aplicação. A Figura 36 mostra a tela principal do protótipo, onde pode-se ver visto os itens descritos anteriormente.

Figura 36 - Abertura protótipo



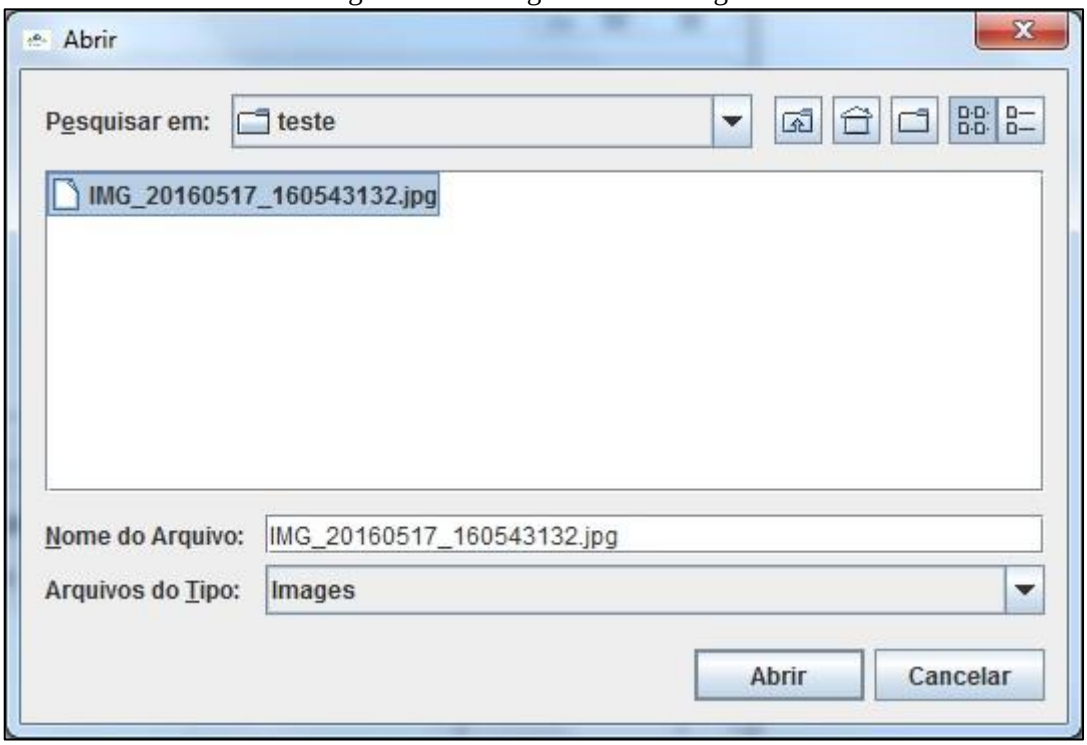
Para carregar uma imagem, o usuário deve acessar o item `Carregar Imagem` do menu `Ações`, conforme mostra a Figura 37.

Figura 37 – Menu do protótipo



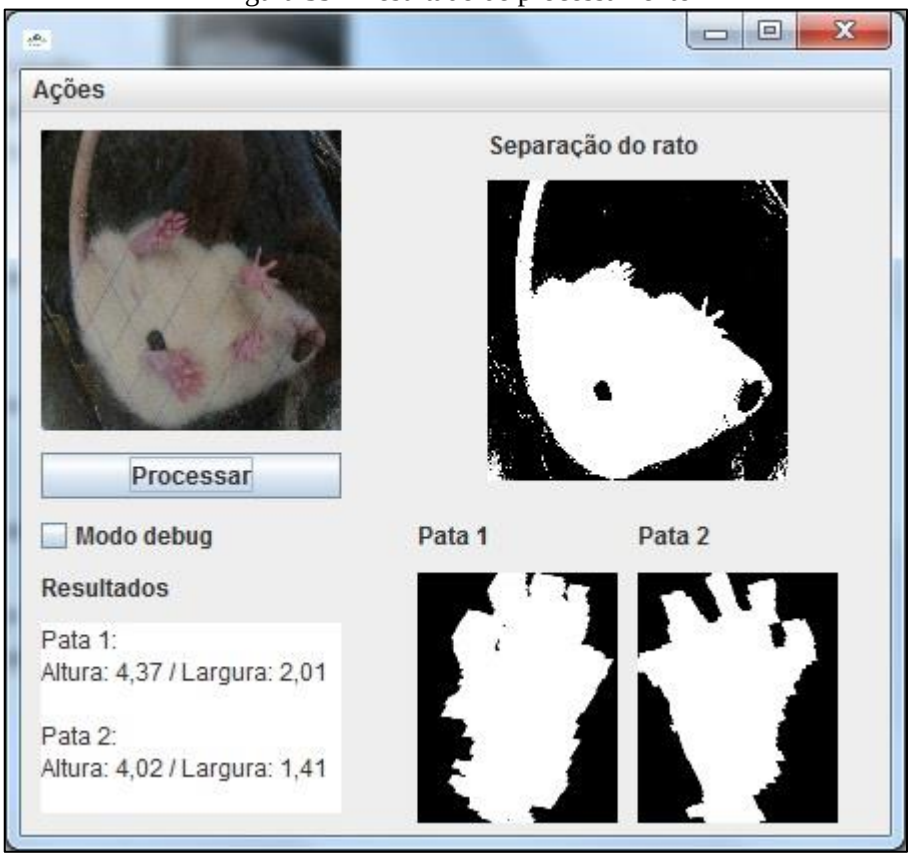
Para facilitar o processo de seleção do arquivo correspondente a imagem do rato, ao clicar no item `Carregar Imagem` é aberta uma janela de exploração de pastas e arquivos. Esta tela é demonstrada pela Figura 38.

Figura 38 - Carregamento da imagem



Ao abrir uma imagem, o botão `Processar` será habilitado. Se o usuário clicar nele, inicia-se o processamento da imagem, apresentando os resultados obtidos, conforme mostra a Figura 39.

Figura 39 - Resultado de processamento



As dimensões de altura e largura de cada uma das patas são apresentadas na caixa de texto `Resultados`, no canto inferior esquerdo do protótipo. Na região superior direita é apresentado o painel `Separação do rato`, que representa a imagem segmentada do rato. Por fim, na região inferior direita existem dois painéis, `Pata 1` e `Pata 2`. Esses painéis apresentam as patas encontradas a partir do seu processo de localização.

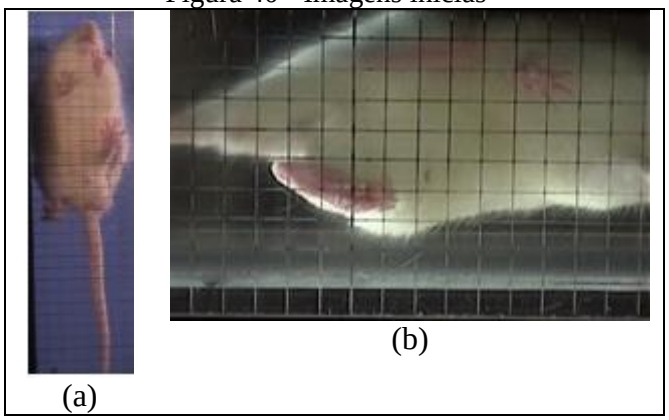
3.4 RESULTADOS E DISCUSSÕES

Esse capítulo apresenta os resultados obtidos a partir da execução do protótipo. A Seção 3.4.1 mostra o processo de montagem da base de testes. A Seção 3.4.2 demonstra os resultados obtidos na etapa de isolamento do corpo do rato. Na Seção 3.4.3 têm-se os resultados referentes à busca das patas do rato. A Seção 3.4.4 apresenta os resultados do cálculo da escala. Por fim, na Seção 3.4.5 encontram-se os resultados das medidas de altura e largura das patas.

3.4.1 Montagem da base de testes

As imagens utilizadas nos primeiros testes não possuíam um ambiente controlado, o que dificultava a identificação do rato e consecutivamente as patas. A Figura 40 mostra duas imagens desse conjunto inicial de imagens.

Figura 40 - Imagens inicias

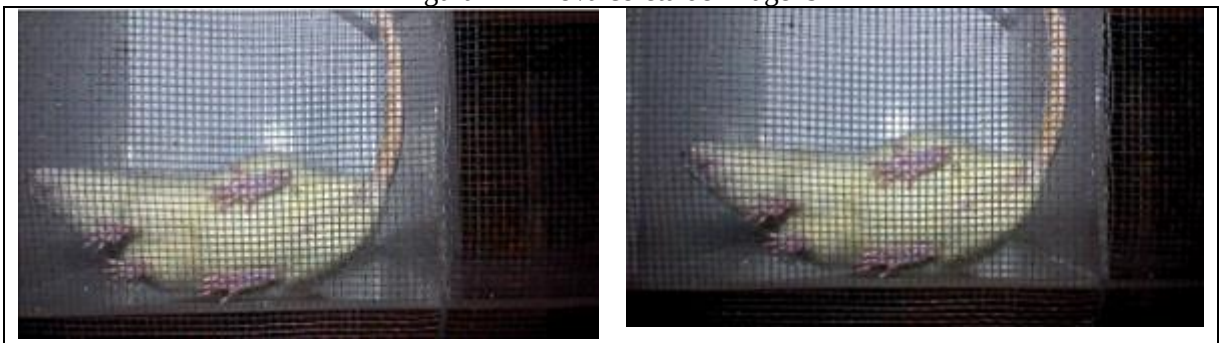


(a) (b)

A partir da Figura 40, pode-se perceber que a iluminação e o fundo com cores distintas prejudicavam os resultados. Dessa forma, este conjunto de imagens foi descartado. Fato que resultou na montagem de uma base própria, na qual foram realizadas 3 tentativas de coleta.

Primeiramente, foi realizado um experimento em que os ratos foram postos sobre uma grade de metal trançado suspensa sobre uma base de madeira, possibilitando a captura de imagens da parte inferior do rato. Para limitar a movimentação do rato sobre a grade, foi utilizada uma caixa de acrílico, aberta nas partes superior e inferior. A Figura 41 mostra duas imagens dessa coleta. A intenção dessa coleta foi padronizar o fundo da imagem com a cor branca. As fotos foram tiradas em um ambiente escuro, tendo apenas o auxílio da lanterna de um celular para iluminar o ambiente.

Figura 41 - Nova coleta de imagens



(a) (b)

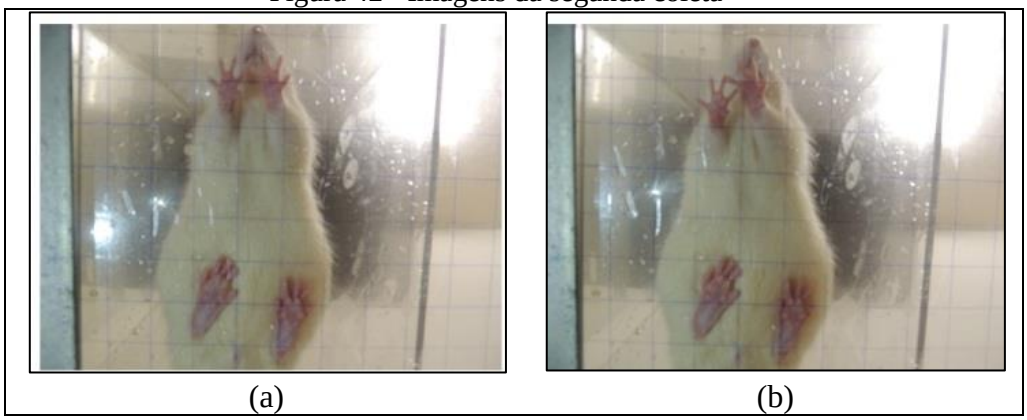
Pode-se perceber, na Figura 41, que a iluminação não obteve um padrão uniforme, havendo um padrão radial em sua incidência. O fundo branco acabou se provando uma escolha que não melhorou os resultados, pois ele acabava se confundindo com a cor clara da pelagem do rato. Além disso, como a grade possuía uma espessura maior do que o necessário, ela acabou sendo mais uma fonte de ruído do que uma forma de obtenção da escala.

Com os avanços praticamente nulos em relação à qualidade das imagens, realizou-se uma segunda coleta. Dessa vez, foram realizadas algumas melhorias no modo de coleta. A grade que havia sido um dos maiores problemas na primeira coleta foi substituída por uma

base de acrílico, que por sua vez tinha colada uma camada de papel *contact* com a grade marcada com caneta.

Com relação ao ambiente, foi optado por utilizar mais iluminação e ao invés de uma folha branca, uma pasta escura foi colocada por cima. Conforme mostra a Figura 42, pode-se perceber que os resultados não saíram como o esperado. O fundo mais escuro posto em cima do acrílico não foi o suficiente para cobrir todo o contorno do rato, sendo assim, não se obteve o contraste desejado entre o fundo e o corpo do rato.

Figura 42 - Imagens da segunda coleta

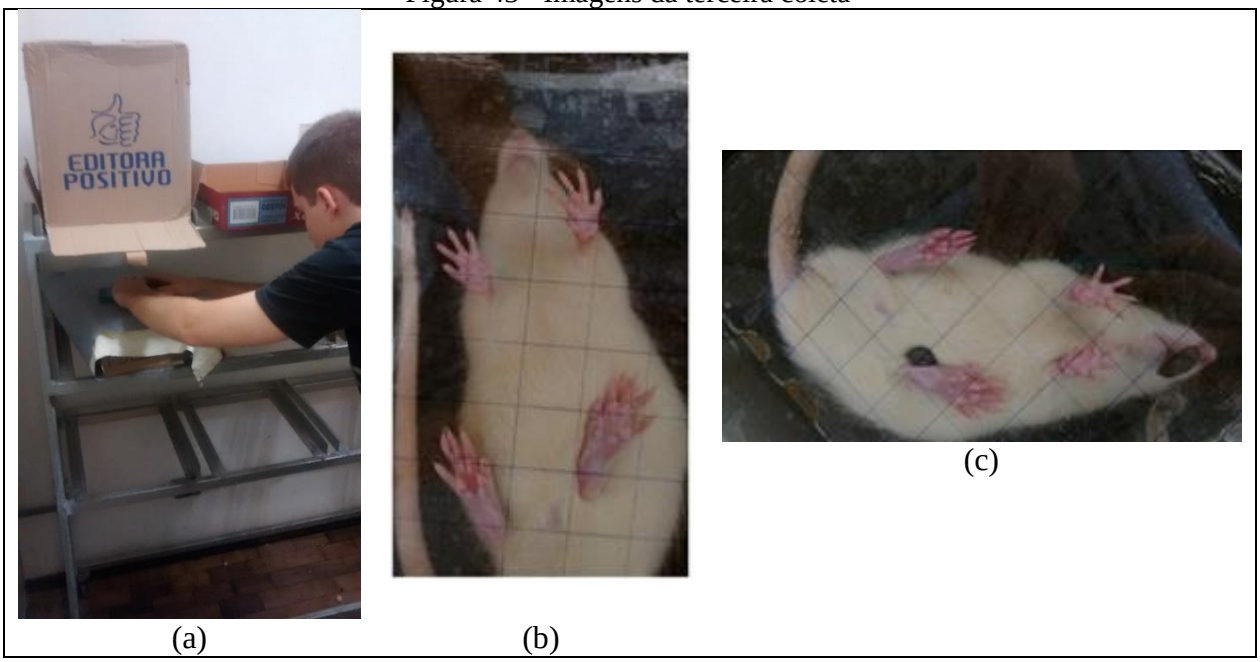


(a) (b)

Pode-se perceber que houve excesso de iluminação no ambiente, causando com isso muitos reflexos e brilhos indesejados na placa de acrílico inferior. Mesmo com tais problemas, esse experimento foi válido para indicar que o ambiente precisava ser mais escuro. Outro fator positivo foi perceber que as linhas estavam mais finas devido à utilização do traço de caneta sobre o papel *contact*. Porém, os resultados do processamento de imagens ainda não se mostraram bons.

Na terceira e última coleta, foi utilizada uma caixa de papelão para cobrir a estrutura de acrílico. O ambiente continuou sendo iluminado por uma lâmpada superior e a base inferior com o acrílico era iluminada pela refração da luz por essa base. Com essa caixa, teve-se uma base iluminada de maneira mais uniforme e um fundo mais escuro, aumentando o contraste em relação ao corpo do rato, facilitando o reconhecimento da região do rato, conforme mostra a Figura 43.

Figura 43 - Imagens da terceira coleta



(a) (b) (c)

Apesar das imagens coletadas terem sido retiradas em um ambiente mais controlado, nem todas as imagens ficaram com boa qualidade, sendo necessário descarta-las. Isso se deu pelo fato de que, no ambiente mais escuro, o rato ficou mais agitado. Assim, muitas fotos não conseguiram destacar as patas como era necessário. Por fim, foram selecionadas 10 imagens que conseguiram uma exposição razoável rato e das suas patas.

3.4.2 Experimento 01: Isolamento do corpo do rato

Neste experimento, objetivo é verificar se o protótipo consegue extrair corretamente a região correspondente ao corpo do rato. A Tabela 1 descreve os resultados obtidos na etapa de isolamento do corpo do mesmo.

Tabela 1 - Resultados do isolamento do corpo do rato

Quantidade de imagens	Imagens que o rato foi isolado completamente	Percentual de imagens com o isolamento correto
10	8	80%

Percebe-se que o protótipo obteve 80% de acerto. Sendo que, das imagens que apresentaram problemas, o rabo do rato acabou ficando muito junto ao acrílico utilizado como barreira, conforme mostra a Figura 44.

Figura 44 - Imagem com ruído junto ao corpo do rato

Imagem original Imagem limiarizada

3.4.3 Experimento 02: Isolamento das patas do rato

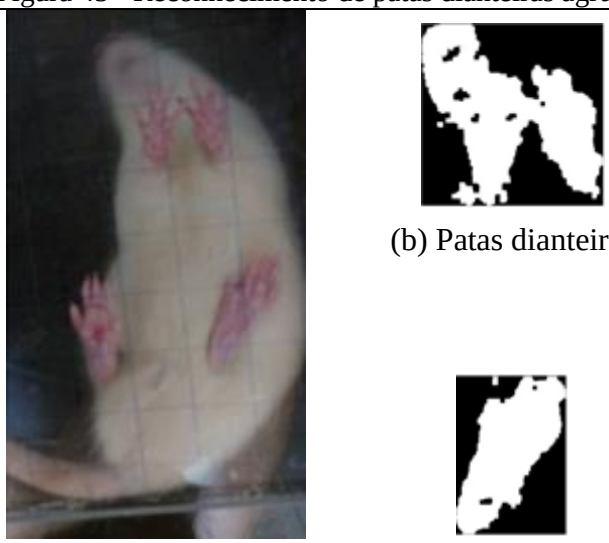
Este experimento tinha por objetivo validar a busca pelas patas traseiras dos ratos. A Tabela 2 mostra os resultados obtidos, onde as colunas indicam a quantidade de imagens usadas, quantidade com nenhuma pata reconhecida, quantidade com apenas uma pata e por fim, a quantidade com as duas patas traseiras encontradas.

Tabela 2 - Resultados do isolamento das patas traseiras

Quantidade de imagens	Nenhuma pata reconhecida	Apenas uma pata reconhecida	Ambas as pata traseiras reconhecidas	Percentual de imagens com correto reconhecimento das duas patas
10	1	4	5	50%

Os principais motivos para a taxa de acerto ser de 50% foram: imagens em que apenas uma parte da pata era encontrada, imagem com esquema de cores muito detalhadas e em alguns casos as patas dianteiras estavam juntas. A Figura 45 apresenta um caso em que as patas dianteiras ficaram agrupadas.

Figura 45 - Reconhecimento de patas dianteiras agrupadas



(a) Imagem original (b) Patas dianteiras (c) Pata traseira direita

A Figura 45a representa a imagem original, onde se pode verificar que as patas dianteiras estão sobrepostas. Essa proximidade levou-as a serem agrupadas como uma única pata. Como o objetivo é identificar apenas o tamanho das patas traseiras, essas patas (dianteiras), foram identificadas de forma incorreta como uma pata traseira, conforme mostra a Figura 45b. Sendo que apenas a pata traseira direita (Figura 45c), foi corretamente encontrada.

3.4.4 Experimento 03: Cálculo da escala da imagem

Este experimento visa verificar se a conversão do tamanho das patas, em pixels, foi feita corretamente para escala de centímetros. Essa escala foi extraída com base nas linhas de referência que formavam uma grade na imagem. Aconteceu que as linhas se mostraram estreitas demais para serem identificadas com facilidade. Além desse problema, foi identificado que em imagens desfocadas não era possível aferir a escala da amostra.

Na Tabela 3 são apresentados os resultados obtidos na definição da escala das 10 amostras analisadas. Das amostras que não se conseguiu estabelecer a escala, constatou-se que a causa do problema era a falta de foco de algumas delas, atrapalhando a definição dos contornos das linhas de referência. Cinco (5) imagens não obtiveram o valor da escala, 1 imagem teve a escala definida com o valor incorreto e as outras 4 foram a que obtiveram valores próximos da escala real.

Tabela 3 - Resultados da definição da escala

Imagens utilizadas	Não foi possível definir a escala	Escala foi definida incorretamente	Escala definida próxima do valor real	Percentual de escalas com valores próximos
10	5	1	4	40%

A escala foi definida a partir da quantidade de pixels existentes em 2 centímetros. Nesses testes, foram aferidas as diferenças entre os valores reais e os obtidos, sendo que uma diferença de até 20 pixels é considerada com um valor próximo do real. A Tabela 4 apresenta os valores obtidos na definição da escala de todas as amostras existentes na base de imagens.

Tabela 4 - Resultado de todos os valores de definição da escala

Amostra	Valor real da escala	Valor obtido para a escala	Diferença do valor real para o obtido
Amostra 1	395	-	-
Amostra 2	390	354	36
Amostra 3	395	376	19
Amostra 4	315	301	14
Amostra 5	165	-	-
Amostra 6	300	-	-
Amostra 7	390	-	-
Amostra 8	305	304	1
Amostra 9	350	342	8
Amostra 10	340	-	-

3.4.5 Experimento 04: Cálculo da altura e largura das patas

Com as patas do rato isoladas e o valor da escala da imagem descoberto, este experimento tem o objetivo de averiguar a eficiência do algoritmo utilizado para calcular o tamanho das patas (largura e altura). A Tabela 5 apresenta os resultados das dimensões extraídas das patas. Ela apresenta os valores em pixels, caso não tenha sido encontrada a escala para a imagem ou, em centímetros, caso a escala tenha sido descoberta com sucesso.

Na Tabela 5, para todas as imagens de amostra, foram descobertos os valores para altura e largura de cada uma das patas traseiras, assim como os valores reais dessas dimensões. Ela também apresenta a quantidade de patas que tiveram o valor de alguma dimensão próxima da real.

A margem de erro para uma dimensão ser considerada próxima do valor real é de 5%, para mais ou para menos. Para cada uma das 10 amostras utilizadas, foram calculadas 4 dimensões, com isso, existem ao todo 40 dimensões calculadas. Dessas dimensões, 15 ficaram dentro da margem de erro tolerável. Desse modo, a taxa de acerto das dimensões das patas ficou em 37,5%.

Tabela 5 - Resultados do cálculo das dimensões

Amostra	Escala	Altura real da pata 1	Altura obtida da para 1	Largura real da pata 1	Largura obtida da pata 1	Altura real da pata 2	Altura obtida da para 2	Largura real da pata 2	Largura obtida da pata 2	Patas com dimensão próxima a real
Amostra 1	px	819	791	390	397	801	793	276	230	3
Amostra 2	cm	3,82	3,93	1,74	1,66	4,16	1,41	1,57	0,89	2
Amostra 3	cm	4,02	3,20	1,97	1,84	3,80	3,5	1,66	1,59	1
Amostra 4	cm	4,00	3,77	1,91	1,99	3,86	3,95	1,56	1,56	3
Amostra 5	px	339	162	183	59	336	128	187	76	0
Amostra 6	px	592	579	248	353	563	595	268	285	1
Amostra 7	px	644	943	320	655	778	785	328	346	1
Amostra 8	cm	4,37	4,37	2,16	2,01	3,87	4,02	1,36	1,41	3
Amostra 9	cm	4,36	3,83	1,76	1,64	4,05	3,84	1,86	1,69	1
Amostra 10	px	714	1080	296	518	605	647	305	267	0
Total	-	-	-	-	-	-	-	-	-	15

Pixel (px) e Centímetro (cm)

As falhas no cálculo das dimensões aconteceram nos casos em que as patas dianteiras estavam juntas, sendo reconhecidas como sendo apenas uma pata com uma dimensão maior do que o esperado. Houve casos em que a pata apresentava muitas variações de tons, causando separações ou perdas de algumas partes da mesma. Na Figura 46, pode-se perceber que a grande variação de cores dificulta o reconhecimento completo das patas, fazendo com que uma parte do contorno delas seja perdida.

Figura 46 - Patas com pouco padrão de cor

4 CONCLUSÕES

A proposta deste trabalho foi desenvolver um protótipo para auxiliar o processo de acompanhamento de recuperação de LNP em ratos, calculando as dimensões de suas patas traseiras. Foi utilizada a linguagem Java para construção do protótipo, utilizando a biblioteca JavaCV para facilitar o trabalho de processamento de imagens.

Inicialmente foram recebidas imagens das patas de ratos retiradas em experimentos prévios, entretanto essas imagens se mostraram de baixa qualidade. Para contornar essa situação, foram feitos experimentos de coletas para levantamento da própria base de imagens.

O objetivo de identificar as patas traseiras do rato foi atendido parcialmente. Foram encontrados casos em que as patas dianteiras ficavam agrupadas. Como a técnica de descoberta das patas utilizava os dois maiores contornos encontrados a partir do corpo dos ratos, isso causou um falso positivo para uma das patas traseiras.

O objetivo de calcular a largura da pata foi atendido de forma parcial. O problema encontrado foi a não obtenção da escala da imagem para conseguir retornar a informação em centímetros. Nesses casos, a informação da largura foi retornada em pixels, o que não serve para realizar o cálculo do IFC.

O objetivo de estabelecer a altura das patas foi atendido parcialmente. A razão do problema foi semelhante ao encontrado na definição da largura da pata, a falta da obtenção do valor da escala da imagem. Além disso, também se observou uma maior variação na definição da altura em relação à largura. Talvez isso tenha ocorrido devido a pouca definição do calcanhar e das pontas dos dedos, que nem sempre eram bem definidos ao utilizar a camada matiz do modelo de cores HSV.

Pode-se concluir que o protótipo alcançou valores de largura e altura próximos aos reais. Porém, ainda é possível obter resultados mais precisos e consistentes em relação à taxa de acerto, pois neste trabalho, conseguiu-se apenas 35% de exatidão. Para ser utilizada por fisioterapeutas em tratamento de LNP, ainda é necessário melhorar a taxa de acerto. Mas, este protótipo indica que é possível criar uma ferramenta para automatização desse processo.

4.1 EXTENSÕES

Como sugestões de extensões para o protótipo propõem-se:

a) aumentar a quantidade de amostras da base de imagens, utilizando uma grade um pouco mais destacada;
b) substituir o esquema de separação dos canais RBG para o HSV durante o reconhecimento do corpo do rato;

c) melhorar a forma de descoberta das patas traseiras para evitar os falsos positivos de patas dianteiras agrupadas, aplicando validações da forma da pata;
d) avaliar a utilização da Transformada de Hough para a detecção das linhas da grade que estabelecem a escala.

REFERÊNCIAS

CAVALCANTE, E. V. V. **Os efeitos da eletroestimulação na regeneração do nervo isquiático de ratos submetidos à lesão por esmagamento**. 2011. 41 f. Dissertação (Pós-Graduação em Fisioterapia) - Pós-Graduação em Fisioterapia do Centro de Ciências da Saúde da Universidade Federal de Pernambuco, Recife.

CONSELHO NACIONAL DE FISIOTERAPIA, 2015. Disponivel em: <http://www.coffito.org.br/site/index.php/fisioterapia/definicao.html>. Acesso em: 26 jun. 2016.

COSTA, J.; CAMARGO, V. M.; ANDRÉ, E. S. Desenvolvimento de um método de baixo custo para avaliação da marcha em ratos. **Fisioterapia em Movimento**, Curitiba, v. 21, n. 2, p. 115-123, 2008.

FACON, J. **Morfologia matemática:** teoria e exemplos. Curitiba: Champagnat, 1996.

FRANCO, R. E. A. Avaliação de método digital para análise do índice funcional do ciático em ratos. **Bioscience Journal**, Uberlândia, v. 27, n. 2, p. 305-311, 2011.

GONZALES, R. C.; WOODS, R. E. **Processamento de imagens digitais**. 2ª. ed. São Paulo: Edgard Blucher, 2000.

HOUGH, P. V. C. **Method and means for recognizing complex patterns**. 3.069.654, 18 Dezembro 1962.

KLAVA, B. Ferramenta interativa para segmentação de imagens digitais, 2006. Disponivel em: <http://www.ime.usp.br/~klava/tfs/tfs_klava.pdf>. Acesso em: 25 jun. 2016.

MACHADO, A. **Neuroanatomia funcional**. 2ª. ed. São Paulo: Atheneu, 2000. 102-03 p.

MEDINACELLI, L.; FREED, W. J.; WYATT, R. J. An Index of the functional condition of rat sciatic nerve. **Experimental Neurology**, Amsterdam, v. 77, p. 634-643, 1982.

MONTE-RASO, V.; BARBIERI, C.; MAZZER, N. Sciatic functional index in smashing injuries of rats' sciatic nerves. Evalution of method reproducibility among examiners. **Acta Ortop Bras**, v. 14, n. 3, p. 133-36, 2006.

POSSAMAI, F. E. A. Repercussões morfológicas e funcionais do exercício sobre a regeneração nervosa periférica. **Fisioterapia em movimento**, Curitiba, v. 25, n. 3, p. 617-627, 2012.

SURAL, S.; QIAN, G.; PRAMANIK, S. **Segmentation and histogram generation using the HSV color space for image retrieval**. International Conference on Image Processing. Rochester, NY: [s.n.]. 2002. p. 589-592.

YILMAZTÜRK, F. Full-automatic self-calibration of color digital cameras using color targets. **Optics Express**, v. 19, n. 19, 2011.

YUAN, G. Understanding tracks of diferente species of rats. **Communication and Information Technology Research Technical Report**, Auckland, v. 187, 2006.

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.

O caso de uso `UC01 Carregar imagem` é detalhado no Quadro 35.

Quadro 35 – `UC01 - Carregar imagem`

Número	01
Caso de Uso	Carregar imagem
Descrição	Este caso de uso realiza o carregamento da imagem a ser processada pelo protótipo.
Ator	`Usuário`
Pré-Condições	1. Carregar Imagem
Cenário Principal	1. O Usuário abre o menu de ações. 2. O usuário clica no item carregar imagem. 3. O protótipo abre uma janela para navegação dos arquivos. 4. O usuário seleciona o arquivo de imagem desejado para abrir. 5. O protótipo realiza o carregamento da imagem.

O caso de uso `UC02 Processar dimensões das patas` é detalhado no Quadro 36.

Quadro 36 - `UC02 - Processar dimensões das patas`

Número	02
Caso de Uso	Processar dimensões das patas
Descrição	Este caso de uso realiza a busca das patas traseiras do rato dentro da região do corpo do mesmo, para então calcular as dimensões de altura de larguras dessas patas.
Ator	`Usuário`
Pré-Condições	1. Carregar Imagem
Cenário Principal	1. O protótipo utiliza a imagem carrega. 2. O Usuário clica no botão para processar. 3. O protótipo busca o rato e as suas patas traseiras. 4. O protótipo calcula a altura e largura de cada pata encontrada.