UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

MARCOS DOFF SOTTA FILHO

DESENVOLVIMENTO DE UM SOFTWARE DE RASTREAMENTO DE ROEDORES EM ARENAS

CURITIBA

2022

MARCOS DOFF SOTTA FILHO

DESENVOLVIMENTO DE UM SOFTWARE DE RASTREAMENTO DE ROEDORES EM ARENAS

Development of a software to track rodents is arenas

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Eletrônica, do Departamento Acadêmico de Eletrônica, da Universidade Tecnológica Federal do Paraná (UTFPR).
Orientador: Prof. Dr. André Eugênio Lazzaretti.

CURITIBA

2022

MARCOS DOFF SOTTA FILHO

DESENVOLVIMENTO DE UM SOFTWARE DE RASTREAMENTO DE ROEDORES EM ARENAS

Trabalho de Conclusão de Curso apresentado (a) como requisito para obtenção do título(grau) de Bacharel em Engenharia Eletrônica, do Departamento Acadêmico de Eletrônica, da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 20/dezembro/2021

André Eugênio Lazzaretti
Doutorado
Universidade Tecnológica Federal do Paraná

Gustavo Benvenutti Borba
Doutorado
Universidade Tecnológica Federal do Paraná

Bruno Jacson Martynhak
Doutorado
Universidade Federal do Paraná

CURITIBA

2022

Dedico este trabalho a minha família e aos meus amigos, pelos momentos de ausência.

AGRADECIMENTOS

Primeiramente, gostaria de agradecer ao meu orientador Prof. Dr. André Eugenio Lazzaretti por toda ajuda e todo apoio durante a realização desse projeto. Agradeço também aos professores Bruno Jacson Martynhak e Marcela Verginia de Medeiros por trazer esse projeto à luz e por confiar no meu potencial de realizá-lo.

Agradeço, também, toda a minha família pelo apoio, não só durante essa etapa, mas durante toda a minha vida.

Obrigado, também, à Patrícia e ao Luiz pela ajuda, pelo apoio emocional, pelos rolês loucos e pelas *cherished memories*.

Obrigado ao Gabriel, ao Lucas, à Stéfani, ao Yuri e ao Gustavo por proverem as distrações, quando necessárias, também.

Por fim, agradeço aos demais que ajudaram direta ou indiretamente no desenvolvimento desse trabalho. Apesar de numerosos demais para serem listados, suas contribuições serão sempre lembradas.

„Wozu hab ich Flügel, wenn zu fliegen ich nicht weiß?“ (D’Artagnan)

RESUMO

A coleta de dados é uma parte importante de pesquisas acadêmicas, independente da área, e, como consequência, a automação dessa coleta é um tema recorrente no meio acadêmico. Este trabalho visa fazer a automação da aquisição de dados de movimento de roedores durante experimentos comportamentais. Muitas vezes, grupos de pesquisa que fazem experimentos desse tipo acabam fazendo a coleta de dados de forma manual, sujeitando-se, assim, a erros. Afim de fazer a aquisição desses dados de forma automática, este trabalho propõe o desenvolvimento de um software faz a análise de gravações desses experimentos. Adicionalmente, empregando uma variedade de técnicas de processamento digital de sinais e visão computacional, faz a aquisição dos dados. Após desenvolvido o software, uma série de testes de validação foram realizados. O software apresentou resultados bons, mesmo quando apresentado com condições adversas de luminosidade. Esse software é de grande relevância para a realização de experimentos comportamentais em roedores, pois possibilita a aquisição mais precisa de dados nesse tipo de experimento.

Palavras-chave: processamento de imagens; visão por computador; roedores.

ABSTRACT

Data collection is an important part of academic research, no matter the field, and, therefore, the automation of this task is a reoccurring theme in academia. This thesis aims to automate the acquisition of movement data of rodents during behavioral experiments. Often, research groups that perform this type of experiments, do that manually, subjecting themselves to errors. As to do this acquisition automatically, this thesis proposes the development of a software that analyses the recordings of behavioral experiments. And, utilizing a variety of digital image processing and computer vision techniques, acquires the data. After the development, several validation tests were done. The software presented good results, even in poor lighting conditions. This software is of grate relevance to behavioral experiments in rodents, because it allows precise data acquisition in these experiments.

Keywords: image processing; computer vision; rodents.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Na área de neurociências, existe uma série de testes comportamentais que precisam ser realizados. Estes testes podem ser usados, desde avaliar o estado fisiológico dos roedores, até para verificar os efeitos de novos medicamentos ou novos usos para medicamentos já existentes. Estes testes são realizados com ratos e camundongos devido às relações que podem ser feitas entre alterações no comportamento desses animais e patologias em humanos.

Esses testes são realizados, normalmente, da seguinte forma. Inicialmente, medicamentos são administrados aos roedores. Posteriormente, esses são colocados em arenas e a atividade dos roedores é gravada (para análise posterior) durante um tempo predeterminado. As gravações são analisadas para obter dados sobre o comportamento do roedor durante o experimento (a distância total percorrida sendo o dado de maior interesse, normalmente).

Muitas vezes, a análise da gravação é feita de forma manual. Isso se dá, pois, mesmo existindo softwares capazes de fazer essa análise, esses normalmente possuem preços muito elevados, ou possuem muitas limitações, tornando, assim, sua utilização inviável. Porém, a análise manual apresenta alguns problemas, dos quais se destaca: a imprecisão da medida (que se dá pelo fato do método utilizado para a medição possuir muitas falhas); erros de medida; desperdício de tempo de pesquisadores; e viés dos pesquisadores (mesmo que inconsciente) (ANDREWS *et al.*, 2016).

Para resolver os problemas descritos acima, esse trabalho propõe o desenvolvimento de um software que faz a análise da gravação do experimento e levanta os dados relevantes do movimento do roedor. Esse software utilizará técnicas de processamento digital de sinais, bem como técnicas clássicas de visão computacional para encontrar a posição do roedor na gravação ao longo do experimento.

1.1 OBJETIVOS

1.1.1 Objetivo geral

A proposta consiste em desenvolver um sistema de monitoramento de movimento de roedores durante testes comportamentais na área de neurociências. Esse sistema utilizará gravações realizadas por pesquisadores a fim de determinar dados sobre a movimentação dos

roedores em suas respectivas arenas, principalmente a distância percorrida durante o experimento, utilizando processamento digital de imagens.

1.1.2 Objetivos específicos

Os objetivos específicos do projeto, são:

- Desenvolver um sistema que fará o rastreio de um roedor dentro da arena durante o período do teste, as posições do animal ao longo do tempo serão, então, gravadas em um arquivo CSV.
- Utilizar as informações da posição do roedor ao longo do tempo. Serão levantadas estatísticas sobre a movimentação do animal durante o experimento, como a distância total percorrida.
- Desenvolver uma interface gráfica com o usuário (GUI) que permitirá ao usuário controlar alguns parâmetros do sistema, como a escala, e fazer o desenho das arenas (como um *overlay*).

1.2 ESTRUTURA DO TRABALHO

Este capítulo apresenta o problema e a motivação da realização do trabalho. Em seguida, o capitulo 2 apresenta a fundamentação teórica necessária para a realização e para o entendimento do trabalho. O capítulo 3 apresenta os métodos utilizados para o desenvolvimento. No capítulo 4 são apresentados os resultados obtidos dos testes realizados para a validação do trabalho. Por fim, o capítulo 5 apresenta a conclusão do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 PROCESSAMENTO DIGITAL DE IMAGENS

Uma imagem digital pode ser definida como uma função bidimencional $f(i,j)$, onde a amplitude de f em um par de coordenadas (i,j) é chamada de intensidade ou nivel de cinza da imagem naquele ponto. O campo de processamento digital de imagens se refere ao processamento de imagens digitais utilizando um computador (GONZALES; WOODS, 2008).

Apesar de não haver consenso de autores em relação à distinção entre processamento digital de imagens e campos similares, como visão computacional (GONZALES; WOODS, 2008), para os fins desse trabalho, processamento digital de imagens se referirá a processos onde a entrada e a saída são imagens digitais.

2.1.1 Equalização de Histograma

Equalização de histograma é uma técnica basica para melhorar a qualidade de uma imagem. É utilizada em diversas situações como para melhorar imagens de raio-X e para pré-processamento para detecção de objetos (HONDA *et al.*, 2020).

2.1.1.1 Histograma de uma imagem digital

Para definir um histograma é preciso, primeiramente, assumir que $X = f(i,j)$ é uma imagem composta de L niveis discretos de intensidade, denotados por $\{X_0,X_1,...,X_{L-1}\}$. Aqui, $f(i,j)$ representa a intensidade da imagem na coordenada (i,j) com a condição que $f(i,j) \in \{X_0,X_1,...,X_{L-1}\}$. Dessa forma, pode-se definir o histograma h da imagem, como:

$$h(X_k) = n_k, \, para \; k = 0,1,...,L-1, \tag{1}$$

onde X_k é o k-ésimo nivel de cinza e n_k representa o número de vezes que o nivel de cinza X_k aparece na imagem.

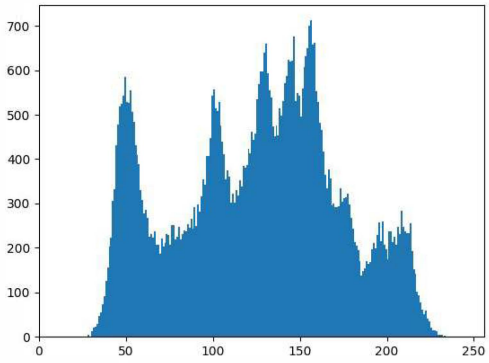
Alternativamente, o histograma também pode ser definido como a função de densidade de probabilidade (do inglês, *Probability Density Function* - PDF) para a intensidade X_k, como:

$$p(X_k) = n_k/N, \, para \; k = 0,1,...,L-1, \tag{2}$$

Figura 1 – Exemplo de imagem e respectivo histograma.

(a) Imagem Original.

Fonte: (CALIFORNIA, 2021)



(b) Histograma da imagem original.

Fonte: Autoria própria

onde N é o número total de amostras na imagem (KONG *et al.*, 2013).

Na Figura 1 é possível observar uma imagem em tons de cinza 1(a) e seu respectivo histograma 1(b).

2.1.1.2 *Global Histogram Equalization* (GHE)

Baseado na PDF descrita na equação 2, é possível definir a função de densidade cumulativa (do inglês, *Cumulative Density Function* - CDF) $c(X_k)$ para a intensidade X_k como a seguinte equação:

$$c(X_k) = \sum_{j=0}^{k} para\ k = 0,1,...,L-1. \tag{3}$$

Agora, fazendo $x = X_k$, é possível definir uma função de transformação $g(x)$, definida como:

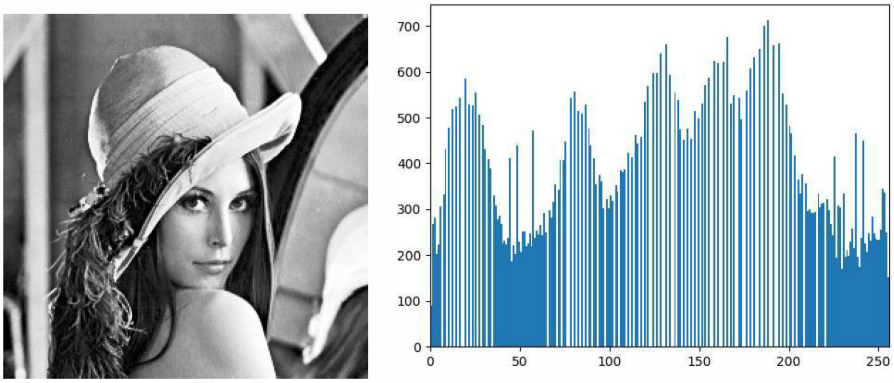
$$g(x) = X_0 + (X_{L-1} - X_0).c(x)\ para\ k = 0, 1, ..., L-1. \tag{4}$$

Dessa forma, a imagem resultante, que é a saída do GHE, $Y = \{Y_{i,j}\}$, pode ser expressada da seguinte forma:

$$Y = g(X) = \{g(f(i,j))|\forall f(i,j) \in X\}. \tag{5}$$

Na Figura 2 é possível observar a imagem presente na figura 1, após o GHE 2(a) e seu respectivo histograma 2(b).

Figura 2 – Exemplo de imagem após GHE e respectivo histograma.



(a) Imagem após GHE. **(b) Histograma da imagem após GHE.**

Fonte: Autoria própria

2.1.1.3 *Contrast Limiting Adaptive Histogram Equalization* (CLAHE)

Sabe-se que GHE não consegue adaptar-se às características de iluminação locais da imagem de entrada. E, como consequência, falha em melhorar a qualidade de imagens de entrada com problemas de iluminação.

Para superar essa limitação, uma extensão da equalização de histograma conhecida como *Adaptive Histogram Equalization* (AHE) foi criada. a principal ideia do AHE é definir uma função de transformação para cada pixel, baseada nos pixels vizinhos. Tipicamente, o AHE usa uma pequena janela para definir a região de contexto. Apenas pixels interiores à janela são levados em consideração no cálculo da CDF (KONG *et al.*, 2013).

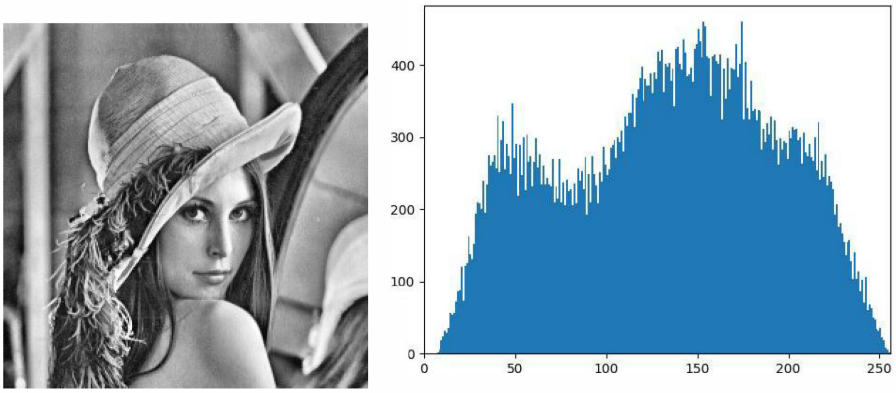
Para prevenir problemas de geração de ruído nas bordas e cantos da janela, foi proposto *Contrast Limiting Adaptive Histogram Equalization* (CLAHE). Na CLAHE, após a geração do histograma da janela, este é grampeado em um *threshold* predeterminado. Isso reduz ruído pois previne que a CDF seja muito íngreme (HONDA *et al.*, 2020).

Na figura 3 é possível observar a imagem após CLAHE 3(a) e seu respectivo histograma 3(b).

2.1.2 Desfoque Gaussiano

A presença de ruído em imagens digitais é inevitável, muitas vezes o ruído é inerente à forma de aquisição das imagens e pode fazer com que pequenas estruturas desapareçam, o que pode interferir no processo de interpretação dessas imagens, portanto é necessário filtrá-las. O

Figura 3 – Exemplo de imagem após CLAHE e respectivo histograma.



(a) Imagem após CLAHE. **(b) Histograma da imagem após CLAHE.**

Fonte: Autoria própria

Figura 4 – Exemplo de imagem com ruído antes e depois de ser aplicado o filtro gaussiano.

(a) Imagem com ruído. **(b) Imagem após aplicação do filtro Gaussiano.**

Fonte: Autoria própria

processo de filtrar envolve reduzir o máximo possível de ruído, perdendo o mínimo possível de informação (GEDRAITE; HADAD, 2011).

Um filtro que pode ser utilizado para fazer isso é o filtro gaussiano, muitas vezes conhecido como desfoque gaussiano. A aplicação desse filtro consiste da convolução de um núcleo com os pixels da imagem que se deseja filtrar. A função usada para gerar o núcleo é dada por:

$$f(x,y) = A.e^{-\left(\frac{(x-x_0)^2}{2\sigma_x^2}+\frac{(y-y_0)^2}{2\sigma_y^2}\right)}, \quad (6)$$

onde A é a amplitude, (x_0,y_0) é o centro do núcleo, σ_x é o desvio padrão na direção x e σ_y é o desvio padrão na direção y (GEDRAITE; HADAD, 2011).

Na Figura 4 é possível observar uma imagem com ruído 4(a) e o resultado da aplicação do desfoque gaussiano nela 4(b).

2.1.3 Transformações morfológicas

Morfologia matemática fornece uma abordagem baseada em formas para o processamento digital de imagens. Quando usada de forma apropriada, morfologia matemática tende a simplificar imagens, preservando as formas essenciais e removendo irrelevâncias (HARALICK *et al.*, 1987). A seguir, são detalhadas as transformações mais importantes usadas neste trabalho.

2.1.3.1 Dilatação

Dilatação é a transformação morfológica que combina dois conjuntos usando soma vetorial para somar os elementos dos conjuntos. Se A e B são conjuntos em um espaço N-dimensional (E^N) com elementos a e b respectivamente, onde $a = (a_1, a_2,...,a_N)$ e $b = (b_1, b_2,...,b_N)$ são vetores de N dimensões, a dilatação de A por B (representada por $A \oplus B$) é o conjunto de todas as possíveis somas de pares de elementos, onde um dos elementos vem de A e o outro vem de B. E pode ser expressa da seguinte forma:

$$A \oplus B \ = \ \{c \in E^N | \ c = a + b \ para \ algum \ a \in A \ e \ algum \ b \in B\}. \tag{7}$$

(HARALICK *et al.*, 1987)

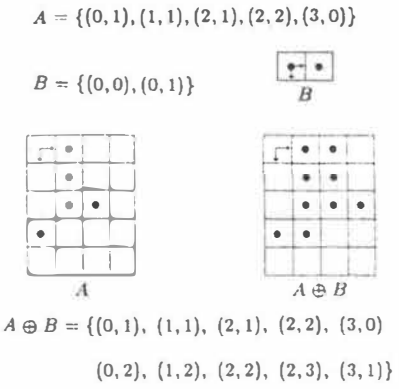
A Figura 5 representa a dilatação de A por B quando ambos possuem poucos elementos. E a Figura 6 representa a dilatação sendo aplicada a uma imagem, representando o mais próximo de como operações morfológicas serão utilizadas nesse trabalho.

2.1.3.2 Erosão

Erosão é o dual morfológico da dilatação. É a transformação morfológica que combina dois conjuntos utilizando a subtração de vetores. Se A e B são conjuntos em um espaço N-dimensional (E^N), então, a erosão de A por B é o conjunto de todos os elementos x tal que $x + b \in A$ para todo $b \in B$. Pode ser expressa da seguinte forma (HARALICK *et al.*, 1987):

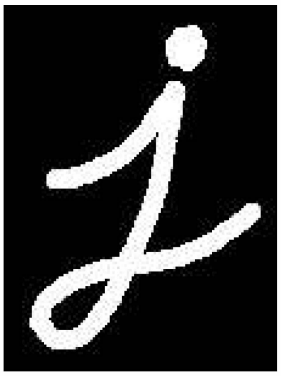
$$A \ominus B \ = \ \{x \in E^N | \ x + b \in A \ para \ todo \ b \in B\}. \tag{8}$$

Figura 5 – Exemplo de dilatação com A e B possuindo poucos elementos.



Fonte: (HARALICK *et al.*, 1987)

Figura 6 – Exemplo de dilatação com A sendo uma imagem de 150x112 pixels.



(a) Imagem Original.

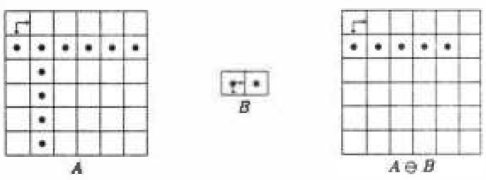
Fonte: (OPENCV, 2021)

(b) Imagem após Dilatação.

Fonte: Autoria própria

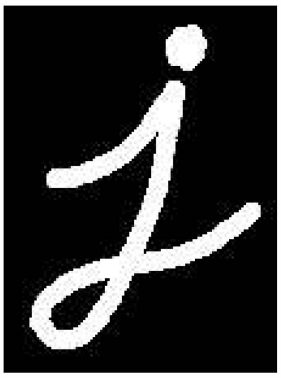
Figura 7 – Exemplo de erosão com A e B possuindo poucos elementos.

$$A = \{(1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 1), (3, 1), (4, 1), (5, 1),\}$$
$$B = \{(0, 0), (0, 1)\}$$
$$A \ominus B = \{(1, 0), (1, 1), (1, 2), (1, 3), (1, 4)\}$$



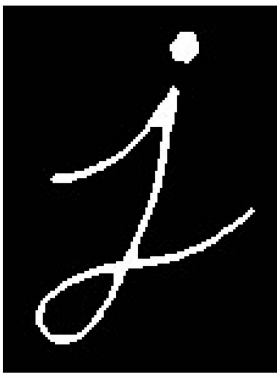
Fonte: (HARALICK *et al.*, 1987)

Figura 8 – Exemplo de erosão com A sendo uma imagem de 150x112 pixels.



(a) Imagem Original.

Fonte: (OPENCV, 2021)



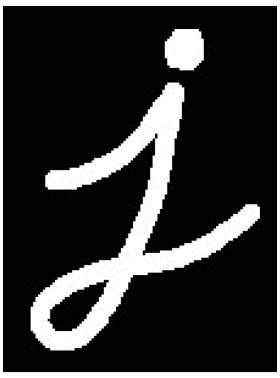
(b) Imagem após erosão.

Fonte: Autoria própria

Figura 9 – Exemplo de abertura de uma imagem.

(a) Imagem Original.

Fonte: (OPENCV, 2021)



(b) Imagem após abertura.

Fonte: Autoria própria

A Figura 7 representa a erosão de A por B quando ambos possuem poucos elementos. Já a Figura 8 representa a erosão sendo aplicada a uma imagem.

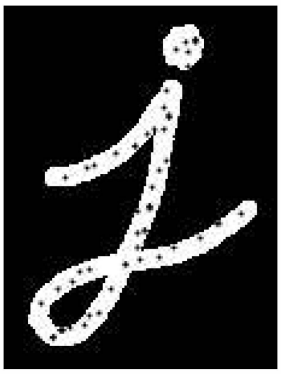
2.1.3.3 Abertura

A abertura de uma imagem B por uma elemento estrutural K (denotada $B \circ K$) pode ser definida como:

$$B \circ K \ = \ (B \oplus K) \ominus K. \tag{9}$$

Abertura pode ser utilizada para suprimir detalhes menores do que o elemento estrutural, sem causar distorção geométrica das características não suprimidas. Por exemplo, utilizando um disco como elemento estrutural é possível suavizar o contorno de uma imagem, suprimindo istmos e eliminando pequenas ilhas (HARALICK *et al.*, 1987), como é possível observar na figura 9.

Figura 10 – Exemplo de fechamento de uma imagem.



(a) Imagem Original.

Fonte: (OPENCV, 2021)

(b) Imagem após fechamento.

Fonte: Autoria própria

2.1.3.4 Fechamento

O fechamento de uma imagem B por uma elemento estrutural K (denotado $B \bullet K$) pode ser definida como:

$$B \bullet K \; = \; (B \ominus K) \oplus K. \tag{10}$$

Similarmente à abertura, o fechamento também pode ser utilizado para suprimir detalhes menores que o elemento principal. Por exemplo, fechar uma imagem utilizando um disco como elemento estrutural eliminará pequenas descontinuidades e buracos na imagem (HARALICK *et al.*, 1987), como é possível observar na Figura 10.

2.2 VISÃO COMPUTACIONAL

A visão é um dos mecanismos que permite que humanos percebam e entendam o mundo a seu redor, enquanto que visão computacional visa replicar os efeitos da visão humana fazendo essa percepção e esse entendimento de forma eletrônica (SONKA *et al.*, 2007).

Fazer com que computadores enxerguem não é uma tarefa fácil. Vivemos em um mundo tridimensional, porém a maioria dos sensores visuais disponíveis entregam imagens bidimensionais e essa projeção em um número menor de dimensões gera uma grande perda de informação (SONKA *et al.*, 2007). As subseções a seguir detalham aspectos importantes de visão computacional, usados neste trabalho.

2.2.1 Detecção de Contornos

Detecção de contornos é uma das técnicas fundamentais no processamento de imagens digitais binarizadas. Essa detecção extrai a sequência de coordenadas da borda entre um elemento de 1-pixels (branco) e 0-pixels (preto).

O contorno (ou borda) será, então, o conjunto de pixels que existe entre um 1-componente (conjunto de 1-pixels contíguos) e um 0-componente (conjunto de 0-pixels contíguos). O contorno externo de um determinado componente pode ser obtido da seguinte forma (SUZUKI; ABE, 1985):

1. Escaneia-se a imagem (como rasterização de TV);
2. Encontra-se o primeiro pixel do contorno. Ele será o primeiro pixel onde $f(i,j) = 1$ e $f(i,j+1) = 0$;
3. Percorre-se os 8 pixels adjacentes a esse pixel, procurando pixels que também são de borda, ou seja, 1-pixels com 0-pixel(s) adjacentes;
4. Verifica-se se esse pixel já pertence ao contorno (essa etapa depende da implementação):
    a) Se não for, marca-se o pixel como sendo parte do contorno, e repete-se os passos a aprtir de 3;
    b) Se for, significa que o contorno foi fechado, e o processo para esse contorno pode ser terminado.

2.2.2 Subtração de fundo

Subtração de fundo é o método computacional utilizado para separar objetos no plano frontal de objetos no plano de fundo em uma sequência de *frames* de um vídeo. Um método possível para se fazer essa subtração é o método MoG (*Mean of Gaussian*) (MOHAMED *et al.*, 2010).

2.2.2.1 MoG - *Mean of Gaussian*

Esse método utiliza múltiplas distribuições gaussianas para determinar o fundo da imagem. MoG mantém uma função de densidade de probabilidade para cada pixel, e um pixel

é dito como pertencente à frente da imagem se intensidade desse pixel está muito distante da media dessa distribuição (SEN-CHING; KAMATH, 2004).

2.3 INTERFACES GRÁFICAS DE USUÁRIO

A interface de usuário é a parte de computadores e *softwares* que as pessoas podem ver, ouvir, tocar, falar com ou direcionar. Uma interface bem desenvolvida é extremamente importante para o usuário. Isso se dá porque para muitos usuários ela é o sistema(GALITZ, 2007).

Softwares baseados em *Graphical user interfaces* (GUI) representam mais de 60% dos softwares produzidos atualmente. Uma das maiores motivações por trás do uso de GUIs é que estas promovem interação fácil e natural entre o sistema e o usuário (ISSA *et al.*, 2012).

2.3.1 Qt

Qt é um *framework cross-platform* compreensivo para criação de aplicações com GUIs usando a abordagem "escreva uma vez, compile em qualquer lugar". Apesar de Qt ter construído sua reputação como um *framework cross-platform*, muitas organizações o utilizam para desenvolver aplicações para apensa uma plataforma (BLANCHETTE; SUMMERFIELD, 2008).

Qt *Widgets* são elementos tradicionais de interface tipicamente encontrados em ambientes de *desktop*. Os *widgets* se integram bem com a plataforma usada, apresentando aparência nativa em Windows, Linux ou MacOS. *Widgets* são uma uma boa escolha para aplicações tradicionais para *desktop* (COMPANY, 2021b).

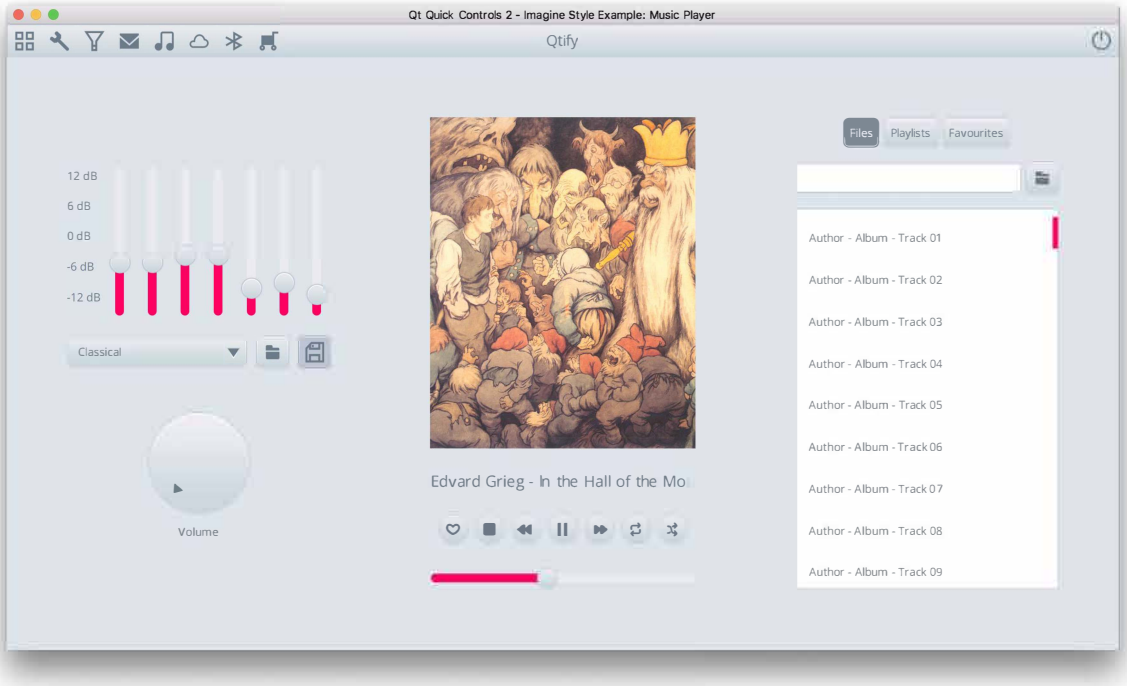
A Figura 11 apresenta um exemplo de aplicação feita utilizando o *framework* Qt.

2.4 GEOMETRIA ANALÍTICA

2.4.1 Ponto interior a um polígono qualquer

Um problema muito natural na área de geometria computacional é o teste de ponto em polígono: dado um ponto R e um um polígono fechado P representado por um $array$ de n pontos $P_0, P_1, ..., P_{n-1}, P_n = P_0$, determinar se R está dentro ou fora do polígono P (HORMANN;

Figura 11 – Exemplo de aplicação *desktop* que pode ser criada com Qt.



Fonte: (COMPANY, 2021a)

AGATHOS, 2001), como detalhado no algoritmo a seguir.

2.4.1.1 Algoritmo *winding number*

O algoritmo *winding number* é baseado no *winding number* de R em relação a P, que é o numero de voltas feitas ao redor de R quando se percorre P uma vez. O ponto R estará dentro do polígono P se o *winding number* for não zero (HORMANN; AGATHOS, 2001).

Formalmente, é possível definir o *winding number* de um ponto R em relação a um polígono P como (HUGHES *et al.*, 2013):

$$WindingNumber(R,P) = \frac{1}{2\pi}\sum_{i=0}^{n-1} cos^{-1}\left(\frac{(P_{i+1}-R).(P_i-R)}{||P_{i+1}-R||.||P_i-R||}\right). \quad (11)$$

3 MATERIAL E MÉTODOS

Este capítulo trata da identificação e análise do problema relatado pelo Prof. Dr. Bruno Jacson Martynhak do departamento de fisiologia da Univesidade Federal do Paraná (UFPR), bem como a proposta de solução, incluindo seus métodos e o planejamento para sua elaboração.

3.1 ANÁLISE DO PROBLEMA

O Prof. Bruno Jacson Martynhak realiza testes comportamentais na área de neurociências utilizando roedores. Estes testes consistem da administração de medicamentos (sejam esses novas drogas ou drogas antigas para as quais está se analisando um novo uso) a fim de avaliar o estado fisiológico (ou patofisiológico) do animal. Após a administração do medicamento, o roedor é colocado em uma arena.

São feitas gravações dos experimentos para documentação e análise destes. A análise dos experimentos pode ser feita de duas maneiras:

1. Análise manual: um humano (normalmente um dos pesquisadores) assiste ao vídeo;
2. Análise com *software*: Utilizar um software que faça a extração ou análise dos dados.

A abordagem 1 normalmente é feita dividindo a arena em quadrantes e contabilizando o número de vezes em que o roedor cruza de um quadrante para o outro. Esta abordagem, no entanto, possui alguns problemas:

- Falta de precisão: devido à divisão em grandes áreas e ao fato de todo movimento feito dentro de um quadrante ser ignorado, uma vez que o roedor não efetuou nenhum cruzamento;
- O viés de observação do pesquisador: mesmo que de forma não intencional, o pesquisador pode desviar os resultados em favor de uma hipótese.

A combinação desses problemas torna a abordagem 2 a mais desejável. Porém, muitos dos *softwares* que realizam esse tipo de análise possuem preços elevados, ou dependem de outros softwares que possuem preços elevado (por exemplo, Matlab).

A solução para o problema analisado, portanto, deverá ser um *software* aberto. A Figura 12 mostra um exemplo de um *frame* retirado da gravação de um dos experimentos.

Figura 12 – Exemplo de *frame* de uma gravação de um dos experimentos.

Fonte: Autoria própria

3.2 PROPOSTA DE SOLUÇÃO

O primeiro passo para a formulação de uma solução para esse problema deve ser o entendimento das necessidades que os pesquisadores têm para com um *software* que faça a análise dos movimentos dos roedores.

O Prof. Bruno apresentou algumas das necessidades dele em relação ao sistema. Essas são:

- O *software* deve possuir uma GUI (*graphical user interface*);
- O usuário deve poder desenhar as arenas;
- O software deve medir a distância percorrida pelo roedor;
- Saída deverá ser em formato de planilha.

Além das necessidades do usuário, há outras considerações a serem feitas sobre como o sistema suprirá essas necessidades, como quais tecnologias serão utilizadas e quais métodos e técnicas serão empregadas. Com base nas necessidades, e nas considerações feitas sobre elas, é possível fazer o levantamento dos requisitos o sistema.

3.2.1 Requisitos de Sistema

Os requisitos de sistema estão descritos abaixo. Os requisitos funcionais são identificadas com **RTFx** (*rodent tracking functional*) e os não-funcionais são identificados com **RTNFx** (*rodent tracking non-functional*).

Os requisitos funcionais são:

RTF1 O sistema deverá medir a distância percorrida pelos roedores durante o experimento.

RTF1.1 O sistema deverá fazer a medição da distância com base em gravações feitas durante os experimentos.

RTF2 O sistema deverá utilizar uma GUI como interface com o usuário.

RTF2.1 O sistema deverá permitir que o usuário selecione a gravação a ser utilizada através da GUI.

RTF2.2 O sistema deverá solicitar ao usuário a escala de referência para análise da gravação.

RTF2.3 O sistema deverá possibilitar que o usuário desenhe as arenas para análise da gravação.

RTF3 O sistema apresentará uma planilha como saída para cada roedor presente na gravação.

RTF3.1 A planilha de saída deverá conter os dados crus de posição do roedor em cada *frame* da gravação.

RTF3.2 A planilha de saída deverá conter a distância percorrida em pixels pelo roedor entre cada *frame* da gravação.

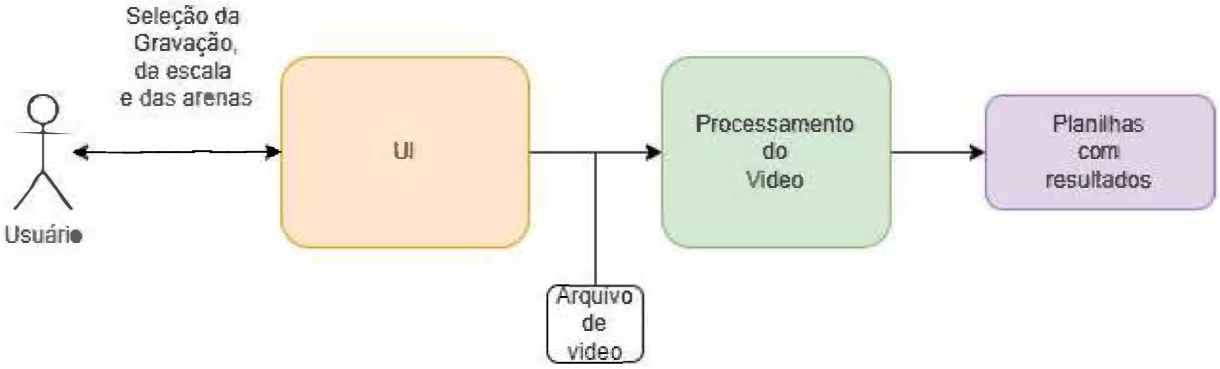
RTF3.3 A planilha de saída deverá conter a distância percorrida em metros pelo roedor entre cada *frame* da gravação.

RTF3.4 A planilha de saída deverá conter a distância total percorrida pelo roedor durante toda a gravação.

Os requisitos não-funcionais são:

RTNF1 O sistema uilizará técnicas clássicas de visão computacional para fazer o rastreamento dos roedores.

Figura 13 – Diagrama de funcionamento (alto nível) do software



Fonte: Autoria própria

RTNF1.1 O sistema utilizará a API OpenCV como API de visão computacional.

RTNF1.2 O sistema utilizará uma combinação de subtração de fundo e detecção de contorno para fazer o rastreamento dos roedores.

RTNF2 O sistema utilizará Python como linguagem de programação para implementação do *software*.

RTNF3 O sistema utilizará Qt como *framework* para implementação da GUI.

RTNF3.1 O sistema utilizará o módulo PySide2 como interface entre Python e Qt.

RTNF4 O sistema deverá ser disponibilizado de forma gratuita.

3.3 DESENVOLVIMENTO DO SISTEMA

Com base nos requisitos, é possível, então, começar a projetar e desenvolver o sistema. O sistema consistirá de um *software* onde as entradas são os *inputs* do usuário (escolha da gravação, escolha das escalas e desenho das arenas) e o vídeo da gravação do experimento, e as saídas serão planlhas (em formato .csv) que contém os resultados.

É possível observar o diagrama básico de funcionamento na Figura 13. Torna-se aparente, observando o diagrama, que o funcionamento do *software* é linear. Primeiro, todas as informações de *input* de usuário são obtidas através da UI, posteriormente, o vídeo é processado, e as planilhas de saída (que contêm os resultados do experimento) são geradas.

Nas próximas seções será detalhado o funcionamento exato do sistema (e de cada uma de suas partes) utilizando conceitos do Capítulo 2.

3.4 FUNCIONAMENTO DO SISTEMA

3.4.1 UI

Além de ser uma necessidade do usuário (formalizada como **RTF2**), GUIs estão presentes na maioria dos *softwares* atuais, conforme mostra a seção 2.3. A GUI foi toda desenvolvida em Qt.

Para os fins desse trabalho, a GUI é a forma de o usuário informar ao *software* alguns parâmetros, sendo eles:

- Qual é o arquivo de vídeo que contém a gravação do experimento.
- Quantas arenas serão analisadas e o formato destas.
- A escala que servirá para fazer a correlação entre pixel percorridos e a distância percorrida em metros.
- O desenho de cada arena.

O funcionamento da parte referente a GUI pode ser resumido utilizando o fluxograma na figura 14.

3.4.1.1 Seleção do aquivo de vídeo

A primeira tela presente após a inicialização do *software* é a tela de seleção do arquivo de vídeo, ilustrada na Figura 15. O usuário, então, clica no botão (...), o que causa a abertura de uma janela de seleção de arquivos, como representado na Figura 16.

Esta janela de diálogo é invocada utilizando uma chamada de API (*QFileDialog.getOpenFileName*), e é uma janela nativa do sistema operacional. Nela, o usuário faz a seleção do arquivo (como em qualquer outro programa de computador).

Após selecionado o arquivo, o usuário é apresentado novamente com a janela de seleção, porém, agora, com o caminho completo do arquivo no local onde antes havia: "*No file Selected*", como representado na Figura 17. Clicando no botão (*Next*) o usuário é levado para a próxima tela.

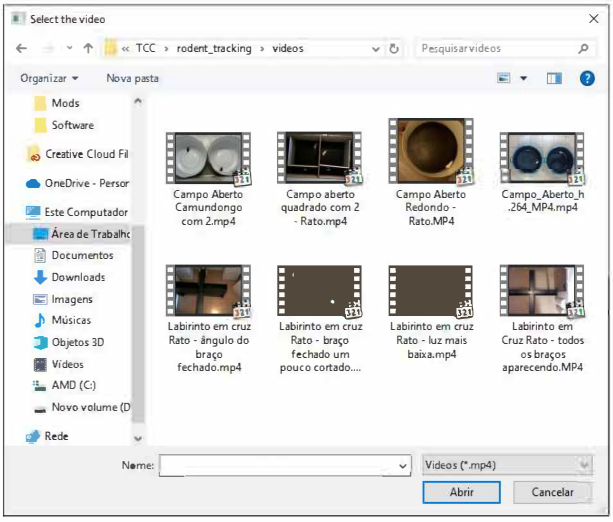
Figura 14 – Diagrama de fluxo da GUI.

Fonte: Autoria própria

Figura 15 – Tela de seleção do arquivo de vídeo.

Fonte: Autoria própria

Figura 16 – Diálogo de seleção de arquivos.



Fonte: Autoria própria

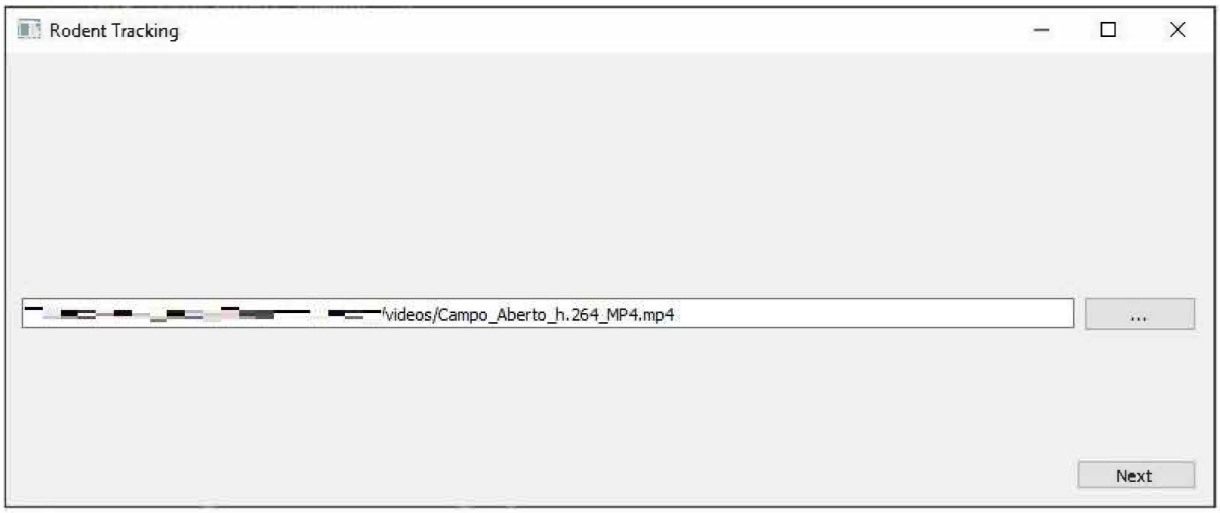
3.4.1.2 Configuração das arenas

Na próxima tela é solicitado ao usuário o número de arenas que serão analisadas e o tipo de desenho da arena (são três tipo possíveis: *CIRCLE*, *RECTANGLE* e *FREE FORM*) conforme a Figura 18.

A seleção do número de arenas é feita através de uma *spinbox*. Enquanto a seleção de uma *combo box* com as opções já predeterminadas, conforme Figura 19.

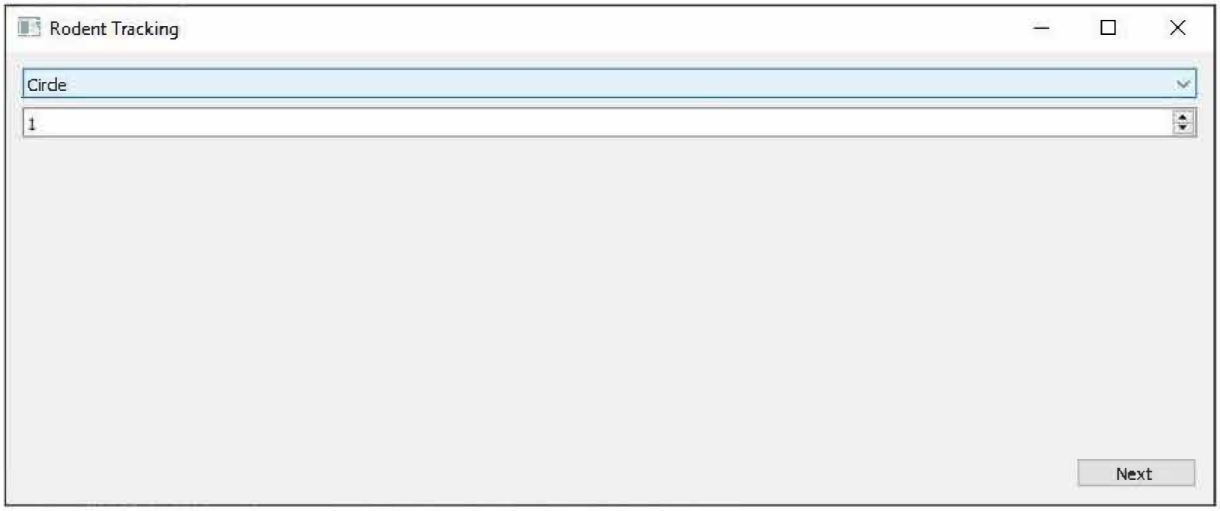
Após a seleção do tipo e da quantidade de arenas o usuário deve clicar no botão (*Next*), e será levado à próxima tela.

Figura 17 – Tela de seleção do arquivo de vídeo (após seleção).



Fonte: Autoria própria

Figura 18 – Tela de configuração das arenas.



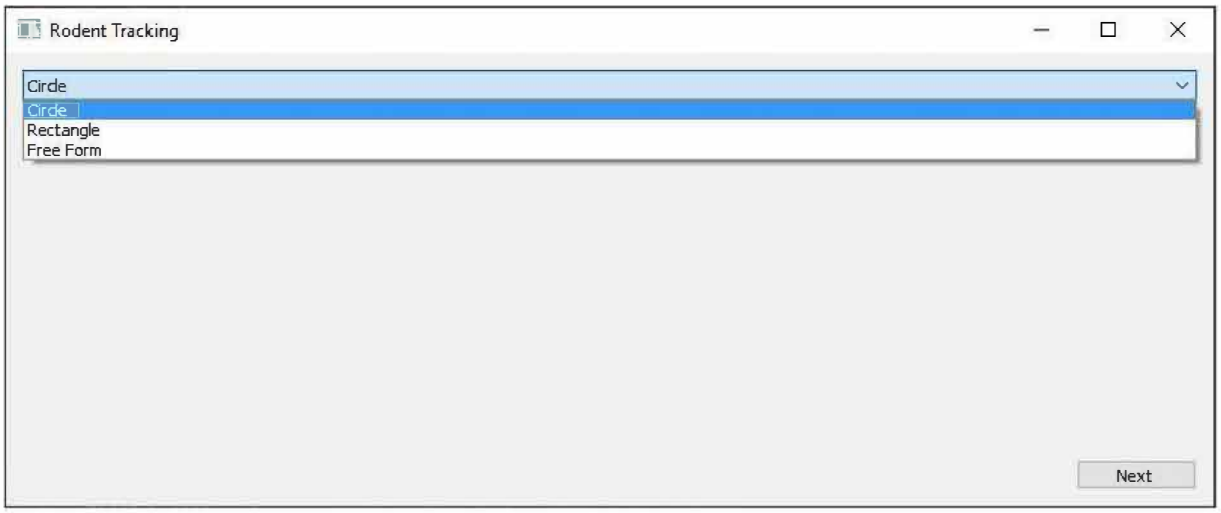
Fonte: Autoria própria

3.4.1.3 Definição da escala

A próxima tela é a tela de definição de escala (Figura 20), onde o usuário é apresentado com um *frame* do vídeo. Nesta tela o usuário seleciona dois pontos (que estão a uma distância conhecida um do outro) e informa a distância em centímetros entre eles. Esses dois dados são combinados para estabelecer uma relação entre as distâncias em pixels do vídeo e as distâncias reais.

Após selecionados os pontos, uma linha azul aparece na imagem entre os pontos selecionados, conforme mostra a Figura 21. Depois de selecionar os pontos e informar a distância

Figura 19 – Tela de configuração das arenas (com *combo box* aberta).



Fonte: Autoria própria

entre eles, o usuário deve clicar no botão (*Next*), e será levado à próxima tela.

3.4.1.4 Desenho das arenas

A próxima tela é a tela onde serão feitos os desenhos das arenas. Essa é a última tela com na qual o usuário fornecerá informações para o *software*. Essa tela, novamente, apresenta um *frame* do vídeo, conforme Figura 22. Nesta tela, o usuário pode fazer o desenho da arena da seguinte forma:

- Para a arena circular: clicar com o botão esquerdo no centro da circunferência, e depois clicar com o botão esquerdo em qualquer ponto da circunferência.
- Para a arena retangular: clicar com o botão esquerdo no vertice superior esquerdo do retângulo, e depois clicar no vertice inferior direito do retângulo.
- Para a arena de forma livre: clicar com o botão esquerdo em todos os vertices da arena, e em seguida clicar com o botão direito em qualquer ponto.

Finalizado o desenho da arena a tela mudará para incluir uma representação da arena desenhada, conforme a Figura 23. Após isso o usuário deve clicar no botão (*Next*), e o processo deve ser repetido até que não hajam mais arenas a serem desenhadas, como ilustrado na Figura 24.

Figura 20 – Tela de definição de escala.

Fonte: Autoria própria

3.4.1.5 Vídeo da detecção

Após serem definidas todas as arenas, o *software* começará o processo de detecção dos roedores, conforme descrito na seção 3.4.2, e uma tela será mostrada para o usuário, onde os *frames* do vídeo são mostrados em sequência (à medida que são processados), com a adição do desenho das arenas e de pequenos círculos, onde se encontram os centroides de cada roedor, naquele *frame*. Um exemplo disso está presente na Figura 25.

Figura 21 – Tela de definição de escala (preenchida).

Fonte: Autoria própria

Figura 22 – Tela de desenho das arenas (sem nenhuma arena completa)

Fonte: Autoria própria

Figura 23 – Tela de desenho das arenas (com uma arena completa).

Fonte: Autoria própria

Figura 24 – Tela de desenho das arenas (com todas as arenas completas).

Fonte: Autoria própria

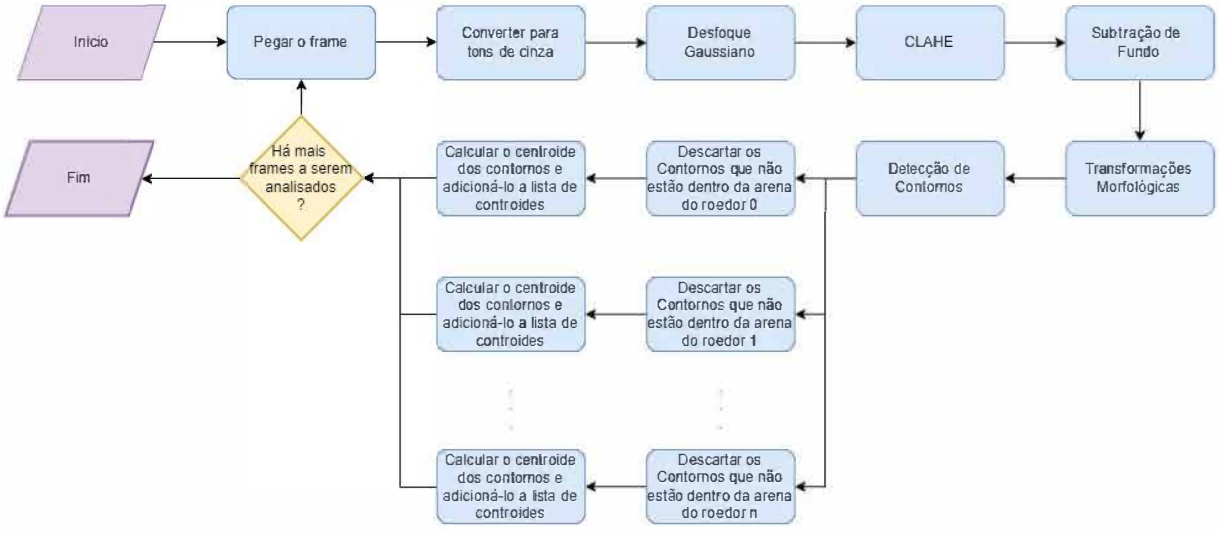
Figura 25 – Exemplo de *frame* com roedores detectados.

Fonte: Autoria própria

3.4.2 Detecção dos roedores

A detecção dos roedores se baseia em detecção de contornos e em restringir esses contornos aos que estão dentro das arenas definidas pelo usuário, porém antes de fazer a detecção de contornos de fato, é necessário fazer uma série de operações de processamento digital de imagens e visão computacional. Esse processo pode ser resumido com o fluxograma presente na Figura 26 e é detalhado nas seções seguintes.

Figura 26 – Fluxograma do funcionamento da detecção dos roedores.



Fonte: Autoria própria

3.4.2.1 Pré-processamento do *frame*

O primeiro passo da detecção consistem em carregar o próximo *frame* do vídeo, um exemplo de *frame* está presente na figura 27 (esse *frame* será utilizado como exemplo para as operações realizadas).

Figura 27 – Exemplo de *frame* extraído de um dos vídeos

Fonte: Autoria própria

Como nenhuma parte da análise do problema envolve análise das cores da imagem, converte-se a imagem para tons de cinza, conforme mostrado na Figura 28.

Após feita essa conversão, pode-se fazer a aplicação de um filtro gaussiano (desfoque gaussiano) a imagem (conforme descrito na seção 2.1.2), o resultado pode ser observado na figura 29. A aplicação desse filtro é uma boa prática antes de ser feita a equalização do histograma.

Figura 28 – *Frame* exemplo após ser convertido para tons de cinza.

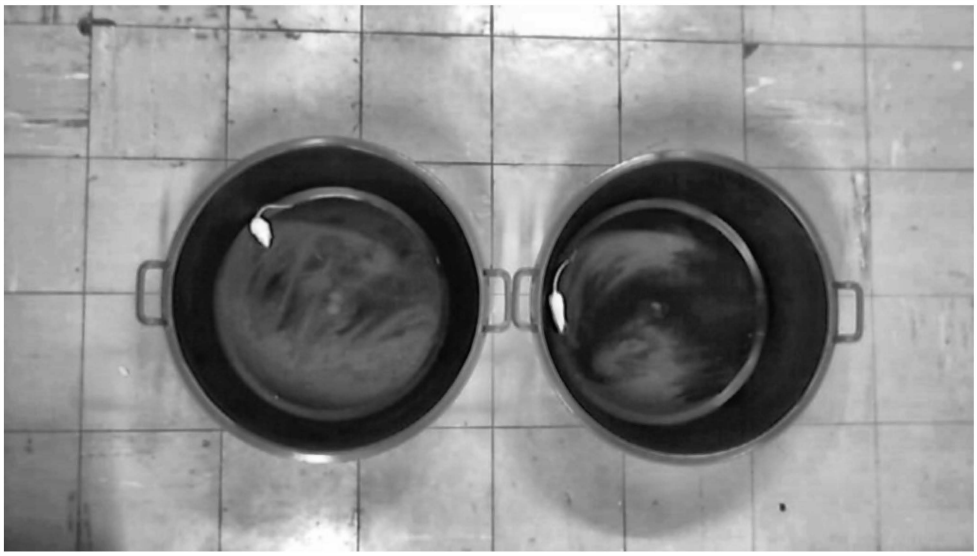
Fonte: Autoria própria

Figura 29 – *Frame* exemplo após aplicação do filtro gaussiano.

Fonte: Autoria própria

O próximo passo é fazer a equalização do histograma, utilizando CLAHE (conforme descrito na seção 2.1.1.3). O resultado dessa operação pode ser visto na figura 30, e serve para realçar o contraste da imagem (principalmente o contraste dos roedores para o fundo da arena).

Figura 30 – *Frame* exemplo após CLAHE.



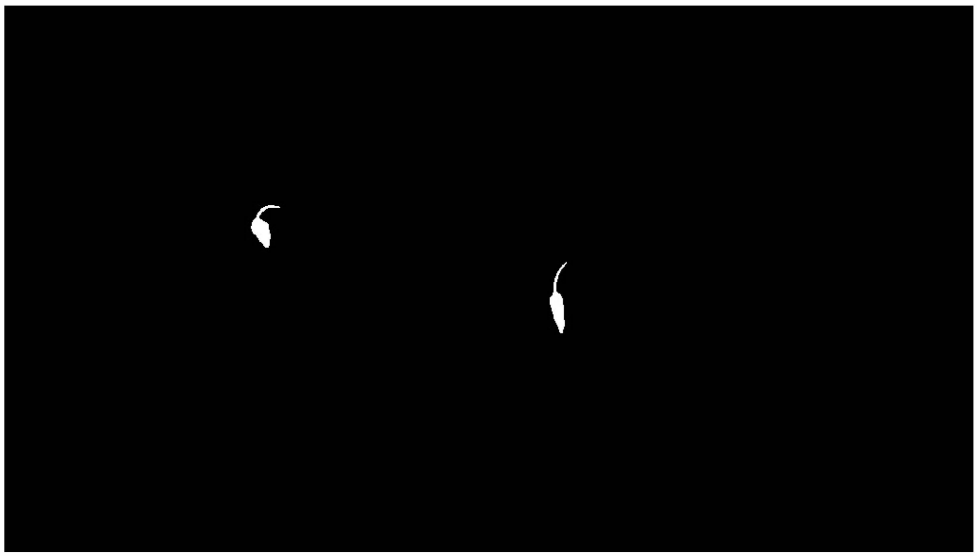
Fonte: Autoria própria

3.4.2.2 Subtração de fundo

Após feito o pré-processamento, pode-se fazer a subtração do fundo do *frame* (conforme descrito na seção 2.2.2), essa etapa é onde a maior parte da imagem será descartada, sobrando apenas os roedores e um eventual ruído.

Normalmente utiliza-se a saída do algoritmo de subtração de fundo como uma máscara, que então é aplicada à imagem original para obter a imagem, porém sem o fundo. No entanto, para detecção de contornos é mais simples utilizar apenas a máscara, pois esta já é uma imagem binária. A máscara para o *frame* de exemplo pode ser observada na Figura 31.

Figura 31 – Máscara da subtração de fundo para o *frame* exemplo.



Fonte: Autoria própria

3.4.2.3 Transformações morfológicas

Algumas vezes, devido à iluminação do local onde a gravação foi feita, pode haver ruído na máscara gerada na seção 3.4.2.2. Para eliminar esse ruído, são feitas algumas operações morfológicas:

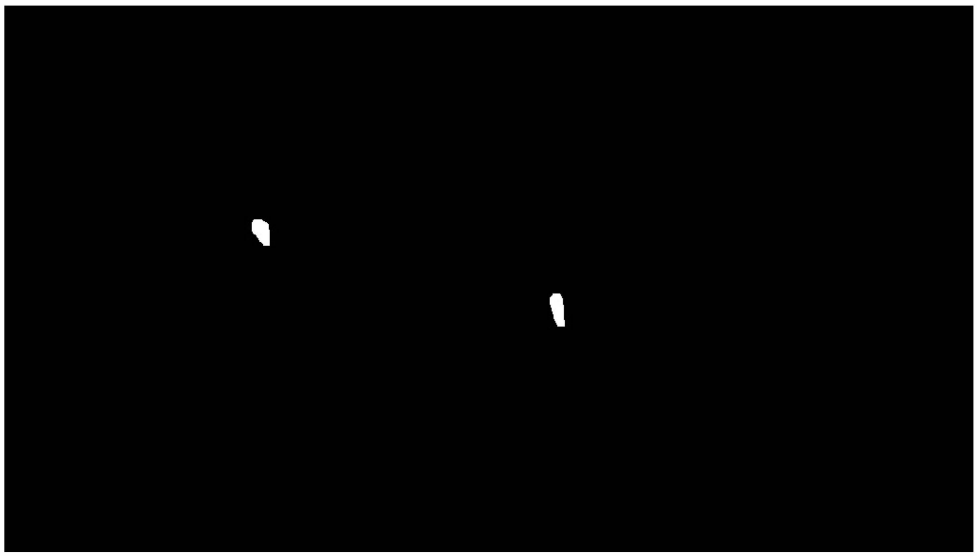
- Uma abertura;
- Um gradiente (diferença entre a dilatação e a erosão);
- Um fechamento.

Essas operações são feitas para remover o ruído e possíveis descontinuidades na máscara, aumentando, assim a robustez do sistema. Esses defeitos podem ocasionar erros nas etapas subsequentes se não forem eliminados (ou, pelo menos, reduzidos). Na Figura 32 é possível observar o efeito dessas operações na máscara do *frame* exemplo.

3.4.2.4 Detecção de contornos

A próxima etapa é a detecção de contornos, feita conforme descrita na seção 2.2.1. Essa é a última etapa de visão computacional e retorna todos os contornos presentes na imagem após as etapas de processamento anteriores. Estes contornos serão usados para determinar a posição dos roedores no *frame* que está sendo analisado.

Figura 32 – Máscara da subtração de fundo para o *frame* exemplo, após operações morfológicas.



Fonte: Autoria própria

Figura 33 – *Frame* de exemplo, com os contornos detectados sobrepostos.

Fonte: Autoria própria

3.4.2.5 Etapas finais

Feita a detecção dos contornos, torna-se, agora, necessário separar esses contornos em duas categorias: (i) Interior à arena que está sendo analisada; (ii) Exterior à arena que está sendo analisada.

Para cada arena que foi definida pelo usuário é, então, feita essa separação. Para determinar se um contorno é interno ou externo à arena, determina-se se os pontos do contorno são internos ou externos à arena. Isto é feito da seguinte forma:

- Para a arena circular: determina-se se a distância entre o ponto e o centro do círculo é menor que o raio do círculo:
  - Se for: o ponto é interno à arena;
  - Se não for: o ponto é externo à arena.
- Para a arena retangular: determina-se se o ponto está abaixo e à direita do ponto superior esquerdo, e se está acima e à esquerda do ponto inferior direito:
  - Se estiver: o ponto é interno à arena;
  - Se não estiver: o ponto é externo à arena.
- Para a arena de forma livre: utiliza-se o método descrito na seção 2.4.1.1.

Uma vez selecionados os contornos internos à arena de interesse, calcula-se o centroide desses contornos. Em seguida, faz-se a média ponderada desses centróides (tendo a área co contorno como peso) para calcular o centróide do roedor. Esse passo é necessário, pois mesmo com as medidas tomadas, algumas descontinuidades podem ainda estar presentes nos contornos quie compõem o roedor.

Após todos esses passos, a informação do centróide do roedor naquele *frame* é salva em um *array*, para ser analisada posteriormente.

3.4.3 Processamento dos dados

Antes de salvar os dados na planilha que será disponibilizada para o usuário é necessário fazer algumas operações. Estas são necessárias para fornecer ao usuário dados legíveis e interpretáveis por humanos, além de eliminar alguns erros de medição.

O primeiro passo nesse processamento de dados é eliminar as primeiras medidas. Isso é necessário, pois as primeiros *frames* inevitavelmente conterão perturbações. Isso se dá ao fato da situação inicial da gravação se encaixar em uma das situações abaixo:

1. O roedor não está na arena no começo da gravação ;
2. O roedor está na arena no começo da gravação .

No caso 1 a perturbação ocorrerá no momento em que o roedor for colocado na arena. Neste momento haverá diversos “objetos” dentro da arena, que poderão ser considerados como parte do roedor, mas não são.

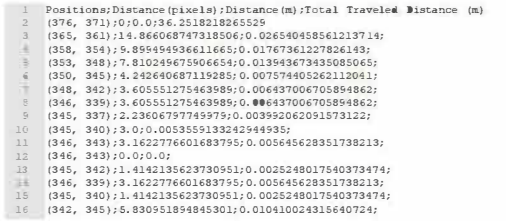
No caso 1, a perturbação está na saída do algoritmo de remoção de fundo. Como o roedor está presente nos primeiros *frames*, o algoritmo pode considerar o roedor como parte no fundo inicialmente.

Após isso, basta fazer o cálculo da distância percorrida entre cada *frame* em pixels, calculando a distância entre o ponto do *frame* atual e o ponto do *frame* anterior.

A distância em metros entre os *frames*, multiplicando a distância em pixels pela escala fornecida pelo usuário. Tendo todas as distâncias entre *frames*, basta somá-las para obter a distância total percorrida.

Tendo todos os dados, a planilha é criada e os dados são colocados nela. Na Figura 34 é possível observar as primeiras linhas da saída.

Figura 34 – Exemplo de planilha de saída (apenas as primeiras linhas são mostradas).



```
Positions;Distance(pixels);Distance(m);Total Traveled Distance (m)
(376, 371);0;0.0;36.25182182655529
(365, 361);14.866068747318506;0.026540458561213714;
(358, 354);9.899494936611665;0.017673612278261433;
(353, 348);7.810249675906654;0.013943673435085065;
(350, 345);4.242640687119285;0.007574405262112041;
(348, 342);3.605551275463989;0.006437006705894862;
(346, 339);3.605551275463989;0.006437006705894862;
(345, 337);2.23606797749979;0.003992062091573122;
(345, 340);3.0;0.005355913324294935;
(346, 343);3.1622776601683795;0.005645628351738213;
(346, 343);0.0;0.0;
(345, 342);1.4142135623730951;0.0025248017540373474;
(346, 339);3.1622776601683795;0.005645628351738213;
(345, 340);1.4142135623730951;0.0025248017540373474;
(342, 345);5.830951894845301;0.010410024315640724;
```

(a) Arquivo csv, visto em um editor de texto.

	A	B	C	D	E	F
1	Positions	Distance(p	Distance(r	Total Traveled Distance (m)		
2	(376, 371)	0	0	36.25182		
3	(365, 361)	14.86607	0.02654			
4	(358, 354)	9.899495	0.017674			
5	(353, 348)	7.81025	0.013944			
6	(350, 345)	4.242641	0.007574			
7	(348, 342)	3.605551	0.006437			
8	(346, 339)	3.605551	0.006437			
9	(345, 337)	2.236068	0.003992			
10	(345, 340)	3	0.005356			
11	(346, 343)	3.162278	0.005646			
12	(346, 343)	0	0			
13	(345, 342)	1.414214	0.002525			
14	(346, 339)	3.162278	0.005646			
15	(345, 340)	1.414214	0.002525			
16	(342, 345)	5.830952	0.01041			

(b) Arquivo csv, visto em um editor de planilhas (Microsoft Excel).

Fonte: Autoria própria

Além da informação de maior interesse para os pesquisadores (a distância total percorrida), os dados crus e as distâncias entre cada pixel foram entregues na planilha para que, caso haja interesse, os dados possam ser processados novamente, sem necessidade de modificar o programa, ou executá-lo novamente.

4 RESULTADOS E DISCUSSÃO

Durante o desenvolvimento do sistema, diversos testes foram realizados afim de validar o sistema, ou adequá-lo, caso o sistema não estivesse apresentando os resultados desejados. Os testes podem ser divididos em três categorias principais: (i) testes de unidade; (ii) testes de integração; e (iii) testes de sistema.

4.1 TESTES DE UNIDADE

Durante o desenvolvimento, diversas sub-rotinas foram criadas. Os testes de unidade consistem em executar estas sub-rotinas de forma isolada, com entradas conhecidas e resultados esperados também conhecidos.

O intuito desses testes é verificar se todas as partes do programa funcionam de forma isolada antes de serem executadas em conjunto. Se as partes não apresentam erros de funcionamento, problemas que aparecerem após a união serão, provavelmente, resultado da união em si e não das partes.

Um exemplo disso é a função que verifica se um ponto é externo ou interno a uma arena. Foram passados pontos conhecidamente fora e dentro de arenas predeterminadas e olhada a saída da função.

Os testes de unidade foram realizados sem auxílio de uma API de testes, afim de economizar tempo de desenvolvimento.

4.2 TESTES DE INTEGRAÇÃO

À medida que as unidades são validadas, elas são unidas e colocadas para funcionar em conjunto. Nesse passo, similarmente aos testes de unidade, esses conjuntos de unidades são executados com entradas conhecidas e resultados esperados.

Um exemplo disso é a integração entre a função que faz o cálculo das distâncias e a função que gera a planilha. Foram colocados pontos aleatórios como entrada e calculadas as distâncias manualmente, depois checou-se se as saídas batiam.

Os testes de integração foram realizados sem auxílio de uma API de testes, afim de economizar tempo de desenvolvimento.

4.3 TESTES DE SISTEMA

Para o sistema proposto, os testes de sistema se tornam um pouco mais complexos, uma vez que a falta de resultados precisos é justamente o problema que se quer resolver.

Para fazer estes teste, portanto, foram seguidos os seguintes passos:

1. Alguns vídeos contendo gravações de experimentos foram adquiridos com os pesquisadores. Estes vídeos apresentam diversas iluminações diferentes e diversos formatos de arenas;
2. O programa é executado para cada um dos vídeos;
3. Durante a execução, observa-se se o ponto marcado como o centro do roedor é coerente;
4. Observa-se se a planilha de saída apresenta resultados coerentes;
5. Além disso foi utilizado um *script* de Matlab com função similar ao software desse trabalho, porém já estabelecido e usado em estudos comportamentais, a fim de comparar os resultados quantitativamente.

4.3.1 Campo aberto com dois camundongos

Esse vídeo apresenta dois camundongos pretos em uma arena de fundo branco, apesar do ambiente estar bem iluminado, há duas áreas de sombreamento na parte superior das arenas, como é possível observar na Figura 35.

Figura 35 – *Frame* da gravação de campo aberto com dois camundongos.

Fonte: Autoria própria

Durante todo o experimento, o indicador do centro do roedor permaneceu sobre o roedor, mesmo quando o roedor estava na zona sombreada.

Tabela 1 – Resultados Quantitativos para Campo aberto com dois camundongos

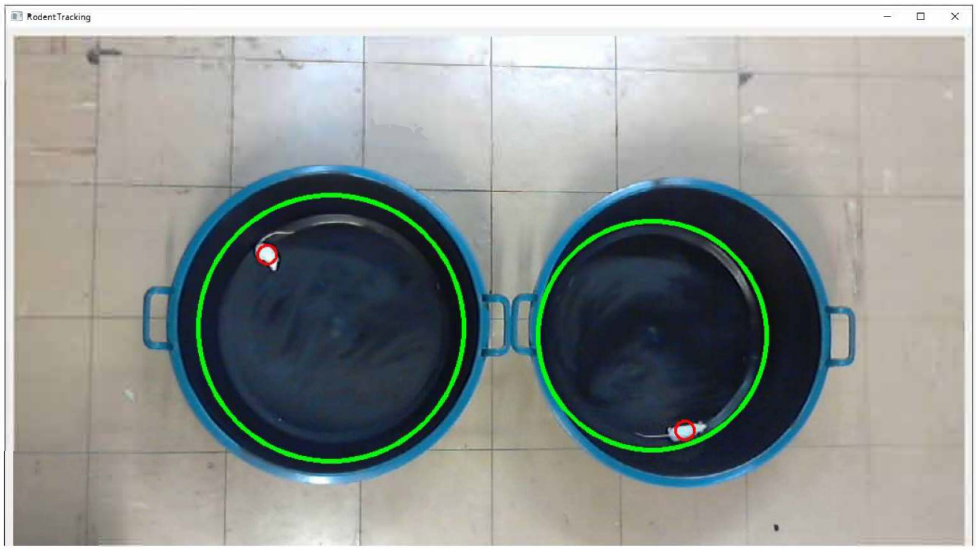
Roedor	Resultado do Software	Resultado do Script de Matlab	Erro (%)
1	16,23 m	15,54 m	4,44%
2	18,24 m	19,03 m	4,15%

Fonte: Autoria própria

4.3.2 Campo aberto com dois ratos (arena circular)

Esse vídeo apresenta dois ratos brancos em arenas circulares de fundo preto. Dos vídeos utilizados para a análise esse é o que possuía melhor iluminação, como é possível observar na Figura 36.

Figura 36 – *Frame* da gravação de campo aberto com dois ratos (arena circular).



Fonte: Autoria própria

Durante todo o experimento o indicador do centro do roedor permaneceu sobre o roedor.

Tabela 2 – Resultados Quantitativos para Campo aberto com dois ratos (arena circular)

Roedor	Resultado do Software	Resultado do Script de Matlab	Erro (%)
1	35,72 m	36,16 m	1,22%
2	31,95 m	32,98 m	3,12%

Fonte: Autoria própria

4.3.3 Campo aberto com dois ratos (arena quadrada)

Similar a apresentada na seção 4.3.2, porém com menos iluminação (a ponto de gerar ruído na imagem) e as arenas são quadradas e possuem paredes brancas, como é possível observar na Figura 37.

Figura 37 – *Frame* da gravação de campo aberto com dois ratos (arena quadrada).

Fonte: Autoria própria

Mesmo com a presença de ruído e com a área de sombra na parte superior das arenas, o ponto que marca o centro do roedor permaneceu sobre o roedor durante todo o experimento.

Tabela 3 – Resultados Quantitativos para Campo aberto com dois ratos (arena quadrada)

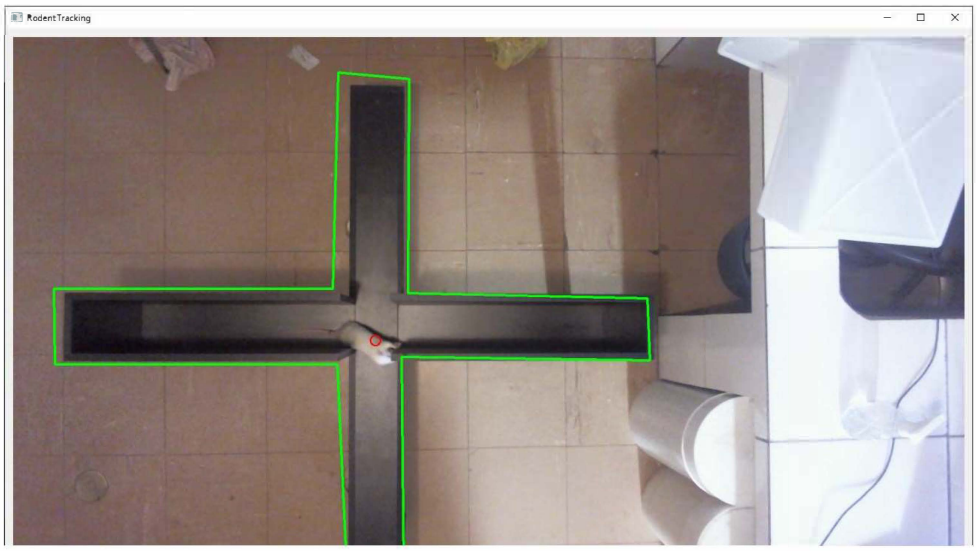
Roedor	Resultado do Software	Resultado do Script de Matlab	Erro (%)
1	7,89 m	7,68 m	2,73%
2	7,36 m	7,04 m	4,55%

Fonte: Autoria própria

4.3.4 Rato em labirinto em cruz

Nesse vídeo o rato branco se encontra em um labirinto em cruz, com fundo escuro. A iluminação é similar a do vídeo da seção 4.3.3, como é possível observar na Figura 38.

Figura 38 – *Frame* da gravação de rato em labirinto em cruz.



Fonte: Autoria própria

No início do vídeo, o rato já está presente na arena, e conforme discutido na seção 3.4.3, no começo do experimento o algoritmo de subtração de fundo considera a posição onde o rato estava como não sendo parte do fundo, o que ocasiona erros no começo da gravação (que serão eliminados no processamento de dados). Após alguns segundos o centro medido do roedor fica sempre sobre o roedor.

Tabela 4 – Resultados Quantitativos para Rato em labirinto em cruz

Roedor	Resultado do Software	Resultado do Script de Matlab	Erro (%)
1	27,75 m	26,43 m	4,99%

Fonte: Autoria própria

4.4 DISCUSSÃO DOS RESULTADOS

Através dos testes realizados é possível perceber que os resultados estão de acordo com o esperado, ou seja, durante os experimentos o sistema mede a posição do roedor de forma correta, independente de luminosidade, cor do roedor e da arena, sombreamento e formato da arena.

Comparando com o *script* de Matlab, é possível perceber que o erro é pequeno em relação a uma ferramenta já utilizada em estudos de comportamento de roedores, sendo o maior erro ainda inferior a 5%.

5 CONCLUSÕES E PERSPECTIVAS

Este trabalho teve como objetivo o desenvolvimento de um *software* para a análise dos movimentos de um roedor durante experimentos comportamentais. Apesar das dificuldades durante o desenvolvimento e das adequações que precisaram ser feitas, é possível dizer que esse objetivo foi alcançado.

O *software* torna possível que pesquisadores façam a análise dos experimentos de forma rápida, precisa e gratuita. E mesmo com condições adversas de iluminação o sistema ainda apresenta bons resultados.

A principal dificuldade no desenvolvimento do trabalho foram os vídeos com iluminação mais precária, algumas abordagens foram usadas para tentar contornar os problemas ocasionados de fazer *thresholding* em imagens com muito ruído gerado por ISOs altas. Devido a isso, foi tomada a decisão de usar subtração de fundo.

5.1 TRABALHOS FUTUROS

Existem melhorias possíveis a serem feitas no software. A mais importante seria a implementação de um filtro Kalman a fim de tornar a saída mais robusta. Além disso, melhoras a GUI, como adicionar instruções no próprio programa. Outros trabalhos com objetivos similares podem explorar o uso de aprendizado de máquina ao invés de utilizar apenas técnicas clássicas de visão computacional.

REFERÊNCIAS

ANDREWS, Nick A; LATRÉMOLIÈRE, Alban; BASBAUM, Allan I; MOGIL, Jeffrey S; PORRECA, Frank; RICE, Andrew SC; WOOLF, Clifford J; CURRIE, Gillian L; DWORKIN, Robert H; EISENACH, James C *et al.* Ensuring transparency and minimization of methodologic bias in preclinical pain research: Pprecise considerations. **Pain**, Wolters Kluwer Health, v. 157, n. 4, p. 901, 2016.

BLANCHETTE, Jasmin; SUMMERFIELD, Mark. **C++ GUI Programming with Qt 4**. *[S.l.]*: Prentice Hall, 2008.

CALIFORNIA, University of Southern. **The USC-SIPI Image Database**. 2021. Disponível em: https://sipi.usc.edu/database/.

COMPANY, Qt. **Qt Quick Controls - Imagine Style Example: Music Player**. 2021. Disponível em: https://doc.qt.io/qt-6/qtquickcontrols-imagine-musicplayer-example.html.

COMPANY, Qt. **User Interfaces**. 2021. Disponível em: https://doc.qt.io/qt-5/topics-ui.html.

GALITZ, Wilbert O. **The essential guide to user interface design: an introduction to GUI design principles and techniques**. *[S.l.]*: John Wiley & Sons, 2007.

GEDRAITE, Estevão S; HADAD, Murielle. Investigation on the effect of a gaussian blur in image filtering and segmentation. *In*: IEEE. **Proceedings ELMAR-2011**. *[S.l.]*, 2011. p. 393–396.

GONZALES, Rafael C.; WOODS, Richard E. **Digital Image Processing**. *[S.l.]*: Pearson Prentice Hall, 2008.

HARALICK, Robert M; STERNBERG, Stanley R; ZHUANG, Xinhua. Image analysis using mathematical morphology. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, n. 4, p. 532–550, 1987.

HONDA, Koki; WEI, Kaijie; ARAI, Masatoshi; AMANO, Hideharu. Clahe implementation on a low-end fpga board by high-level synthesis. p. 282–285, 2020.

HORMANN, Kai; AGATHOS, Alexander. The point in polygon problem for arbitrary polygons. **Computational Geometry**, v. 20, n. 3, p. 131–144, 2001. ISSN 0925-7721. Disponível em: https://www.sciencedirect.com/science/article/pii/S0925772101000128.

HUGHES, John F.; DAM, Andries Van; MCGUIRE, Morgan; SKLAR, David F.; FOLEY, James D.; FEINER, Steven K.; AKELEY, Kurt. **Computer Graphics**. *[S.l.]*: Addison-Wesley, 2013.

ISSA, Ayman; SILLITO, Jonathan; GAROUSI, Vahid. Visual testing of graphical user interfaces: An exploratory study towards systematic definitions and approaches. *In*: **2012 14th IEEE International Symposium on Web Systems Evolution (WSE)**. *[S.l.: s.n.]*, 2012. p. 11–15.

KONG, Nicholas Sia Pik; IBRAHIM, Haidi; HOO, Seng Chun. A literature review on histogram equalization and its variations for digital image enhancement. **International Journal of Innovation, Management and Technology**, IACSIT Press, v. 4, n. 4, p. 386, 2013.

MOHAMED, Shahrizat Shaik; TAHIR, Nooritawati Md; ADNAN, Ramli. Background modelling and background subtraction performance for object detection. *In*: **2010 6th International Colloquium on Signal Processing its Applications**. *[S.l.: s.n.]*, 2010. p. 1–6.

OPENCV. **Morphological Transformations**. 2021. Disponível em: https://docs.opencv.org/3.4/d9/d61/tutorial_py_morphological_ops.html.

SEN-CHING, S Cheung; KAMATH, Chandrika. Robust techniques for background subtraction in urban traffic video. *In*: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. **Visual Communications and Image Processing 2004**. *[S.l.]*, 2004. v. 5308, p. 881–892.

SONKA, M.; HLAVAC, V.; BOYLE, R. **Image Processing, Analysis, and Machine Vision**. *[S.l.]*: Thomson-Engineering, 2007.

SUZUKI, Satoshi; ABE, Keichi. Topological structural analysis of digitized binary images by border following. **Computer Vision, Graphics, and Image Processing**, v. 30, n. 1, p. 32–46, 1985. ISSN 0734-189X. Disponível em: https://www.sciencedirect.com/science/article/pii/0734189X85900167.

GLOSSÁRIO

abnTEX2 uma suíte para LATEX que atende os requisitos das normas da Associação Brasileira de Normas Técnicas (ABNT) para elaboração de documentos técnicos e científicos brasileiros, como artigos científicos, relatórios técnicos, trabalhos acadêmicos como teses, dissertações, projetos de pesquisa e outros documentos do gênero. 53, veja LATEX.

UTFPR-CT-TEX uma suíte para LATEX, baseada na suíte abnTEX2, que atende os requisitos das normas definidas pela Universidade Tecnológica Federal do Paraná (UTFPR), câmpus Curitiba, para elaboração de trabalhos acadêmicos. veja LATEX.

UTFPR-PG-TEX uma suíte para LATEX, baseada na suíte abnTEX2, que atende os requisitos das normas definidas pela Universidade Tecnológica Federal do Paraná (UTFPR), câmpus Ponta Grossa, para elaboração de trabalhos acadêmicos. veja LATEX.

equilíbrio da configuração uma consistência entre os componentes. veja também componente.

LATEX um conjunto de macros para o processador de textos TEX, utilizado amplamente para a produção de textos matemáticos e científicos devido à sua alta qualidade tipográfica. 53

pai um exemplo de entrada pai que possui subentradas (entradas filhas). 53

componente um exemplo de uma entrada componente, subentrada da entrada chamada pai. 53

TEX é um sistema de tipografia criado por Donald E. Knuth. 53

APÊNDICES

APÊNDICE A – LINKS RELEVANTES

Link para o repositório com o código: https://github.com/MarcosDoff/rodent$_t racking$

Link para pasta com exemplos de vídeos: https://drive.google.com/drive/folders/1WkD6M1fSJalloY-r-yqkadlemJ6JIHj1?usp=sharing

Link para script de Matlab: https://github.com/HanLab-OSU/MouseActivity