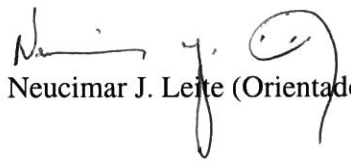
# Rastreamento de Animais por Imagens de Vídeo em Experimentos de Laboratório

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Rafael Henrique Castanheira de Souza e aprovada pela Banca Examinadora.

Campinas, 25 de abril de 2008.



Neucimar J. Leite (Orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

**FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP**
**Bibliotecária: Maria Júlia Milani Rodrigues**

Souza, Rafael Henrique Castanheira de
So89r Rastreamento de animais por imagens de vídeo em experimentos de laboratórios / Rafael Henrique Castanheira de Souza -- Campinas, [S.P. :s.n.], 2008.

Orientador : Neucimar Jerônimo Leite
Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Processamento de imagens. 2. Rastreamento automático. 3. Laboratórios experimentais. 4. Morfologia matemática. 5. Ant algorithms. I. Leite, Neucimar Jerônimo. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Título em inglês: Animal tracking by video images in laboratory experiments

Palavras-chave em inglês (Keywords): 1. Image processing. 2. Automatic tracking. 3. Testing laboratories. 4. Mathematical morphology. 5. Ant algorithms.

Área de concentração: Processamento de Imagens

Titulação: Mestre em Ciência da Computação

Banca examinadora:

Prof. Dr. Neucimar Jerônimo Leite (IC-UNICAMP)
Prof. Dr. Ricardo M.L. de Barros (FEF-UNICAMP)
Prof. Dr. Ricardo da Silva Torres (IC-UNICAMP)

Data da defesa: 28-03-2008

Programa de pós-graduação: Mestrado em Ciência da Computação

# TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 28 de março de 2008, pela Banca examinadora composta pelos Professores Doutores:

**Prof. Dr. Ricardo Machado Leite de Barros**
**FEF - UNICAMP.**

**Prof. Dr. Ricardo da Silva Torres**
**IC - UNICAMP.**

**Prof. Dr. Neucimar Jerônimo Leite**
**IC - UNICAMP.**

Instituto de Computação
Universidade Estadual de Campinas

# Rastreamento de Animais por Imagens de Vídeo em Experimentos de Laboratório

**Rafael Henrique Castanheira de Souza**[1]

Março de 2007

**Banca Examinadora:**

- Neucimar J. Leite (Orientador)
- Ricardo M. L. de Barros
  FEF - UNICAMP
- Ricardo da Silva Torres
- Alexandre Falcão (suplente)

[1]Suporte financeiro de: Bolsa Fapesp (processo 05-03534-0) 2006–2008

# Resumo

O rastreamento automático de animais permite um estudo comportamental mais consistente e rápido do que o feito normalmente utilizando-se registro manual dos parâmetros de experimentos em biologia. O registro automático é realizado por um sistema analisador de imagens que, a partir de uma sequência contínua de quadros de um vídeo, calcula uma série de descritores associados ao movimento das cobaias. O objetivo deste trabalho é criar um sistema de rastreamento para experimentos de laboratório, levando em conta múltiplas cobaias que podem vir a sofrer oclusão. Além disso, pretende-se que o modelo de rastreamento proposto seja robusto a baixa qualidade do vídeo, além de ser geral o suficiente para ser adaptado a outros experimentos com poucas modificações.

# Abstract

The automatic tracking of animals allows a quicker and more consistent analysis of behaviour than the usual manual method for registering experimental parameters in biology. The automatic register of parameters is performed by a system that analyses a sequence of images and computes a number of descriptors that characterizes the behaviour of each target. Our objective is to create a framework for tracking in biology experiments, with multiple targets that may suffer occlusion. Besides, we intend to create a framework that can deal with low-quality videos and capable of being adapted to other classes of tracking.

# Agradecimentos

Primeiramente, agradeço ao professor Neucimar por sua orientação sem a qual este projeto não seria possível. Agradeço também a todos meus professores que tive ao longo destes 6 anos de Unicamp pela formação acadêmica que me proporcionaram. Agradeço ao grupo de pesquisa do *LIS*, pela oportunidade de utilizar seu laboratório e todos os conselhos que me ofereceram sobre meu trabalho. Agradeço ao professor Francesco Langone, do IB-Unicamp, pela idéia inicial para esta dissertação. Agradeço aos funcionários do IC, em especial a Daniel Capeleto, que sempre foram prestativos quando precisei de ajuda. Agradeço à FAPESP pelo apoio financeiro. Agradeço a minha família pelo apoio, tanto material quanto afetivo, a mim dados não só nestes anos de universidade mas em toda minha vida. Sem voces não haveria chegado até aqui. Agradeço a meus amigos por todos estes anos de convivência e camaradagem, além de todo o apoio que me prestaram. Por último mas não menos importante, agradeço a minha noiva Katherine. Seu carinho, paciência, atenção e principalmente seu amor fizeram destes 2 anos os mais felizes de minha vida, apesar de toda a tristeza trazida pela distância. Te amo.

A todos muito obrigado.

# Sumário

# Lista de Tabelas

# Lista de Figuras

# Capítulo 1

# Introdução

Animais são utilizados em uma grande variedade de experimentos. Tais experimentos envolvem pesquisas médicas, modelagem de processos de aprendizagem ou mesmo procedimentos que eticamente não podem ser realizados com humanos [25]. Labirintos são comuns nestes experimentos, sendo utilizados para avaliar a memória espacial e aprendizagem [26]. Nestes experimentos os dados de interesse (parâmetros experimentais) são, em geral, extraídos de forma manual ou semi-automática. Nestes casos, um sistema automático de rastreamento teria as vantagens de ser mais consistente e veloz na análise dos dados experimentais. Tal consistência se deve ao fato de que os algoritmos implementados seguem um mesmo padrão de análise dos dados experimentais, ao passo que dois pesquisadores observando o mesmo experimento podem colher registros diferentes. Além disso, um sistema de rastreamento se mostra muito mais eficiente na extração de certos parâmetros experimentais, como a velocidade e distância percorrida pela cobaia, do que um operador humano.

Um sistema de rastreamento é, de modo geral, utilizado para obter informações compreensíveis sobre objetos móveis ou estáticos presentes em uma cena a partir de uma sequência de quadros de um filme [23]. No caso de rastreamento em labirintos, o sistema deve ser capaz de extrair uma série de informações e calcular parâmetros experimentais particulares a cada tipo de experimento. Os sistemas automáticos de rastreamento são especialmente indicados em três casos: registro de comportamentos breves, que podem ocorrer com espaçamento de várias horas; registros de comportamentos que perduram por várias horas (como o comportamento diurno de uma cobaia); e medidas espaciais, como velocidade e distância percorrida pela cobaia [21].

Existe um significativo aumento de complexidade quando se passa do problema de rastrear um único alvo para o problema de rastrear múltiplos. Em processamento de imagens, segmentação é o processo de dividir uma imagem em regiões. A oclusão considerada neste projeto ocorre quando dois ou mais elementos sendo rastreados se aproximam muito ou se sobrepõem, sendo por isto representados pela mesma região segmentada, o que torna difícil a distinção entre os alvos. Define-se como rastreador uma instância de um algoritmo de rastreamento. Uma

abordagem inicial para realizar o rastreamento de múltiplos alvos seria utilizar um rastreador para cada objeto sendo rastreado. Todavia, o problema da oclusão torna esta alternativa inviável. Quando isto acontece, as instâncias de rastreadores de um único alvo podem deixar de rastrear seus respectivos alvos ou trocar de alvos entre si.

Neste trabalho, foi desenvolvido um modelo de rastreamento para múltiplos alvos com resolução de oclusões. Modelou-se esta solução como um problema de encontrar um conjunto de caminhos em um grafo. Este problema foi resolvido usando-se metodos de otimização combinatória. Os métodos para rastreamento e resolução de oclusões formam um *framework* para rastreamento em experimentos em biologia, com possibilidades de adaptação para outros tipos de rastreamento.

## 1.1 Objetivos

O objetivo deste projeto foi gerar um *framework* para rastreamento, com aplicação inicial em experimentos de biologia. Os métodos utilizados deviam ser robusto o suficiente para lidar com ruídos e degenerações dos vídeos. Este framework define métodos para o rastreamento de múltiplos alvos que podem sofrer oclusão entre si. Tais métodos são gerais o suficiente para permitir fácil adaptação do sistema para diferentes tipos de rastreamento. A utilização inicial do sistema desenvolvido neste projeto é o rastreamento de cobaias em experimentos de laboratório mas, como apresentado na seção 1.2 , o sistema pode ser adaptado a outros tipos de rastreamento.

## 1.2 Exemplos de experimentos biológicos

Existem vários tipos de experimentos utilizados em biologia para, por exemplo, mensurar a orientação espacial, os processos de aprendizado e os efeitos de drogas e doenças sobre cobaias. Neste trabalho, são descritos dois experimentos comumente realizados em geral com ratos ou camundongos: o labirinto aquático de Morris, apresentado na seção 1.2.1, e o teste do campo aberto, na seção 1.2.2. A seção 1.2.3 introduz o problema do rastreamento de leucócitos em tecido nervoso que será considerado aqui como exemplo de extensão do sistema proposto.

### 1.2.1 Labirinto aquático de Morris (*LAM*)

Criado por Morris [26, 22], este labirinto é composto por uma piscina preenchida com água e leite (ou qualquer outra substância que a turve), com uma plataforma submersa poucos centímetros. Como não há pistas da localização da plataforma, o animal deve buscar a plataforma valendo-se de estratégias de mapeamento espacial.

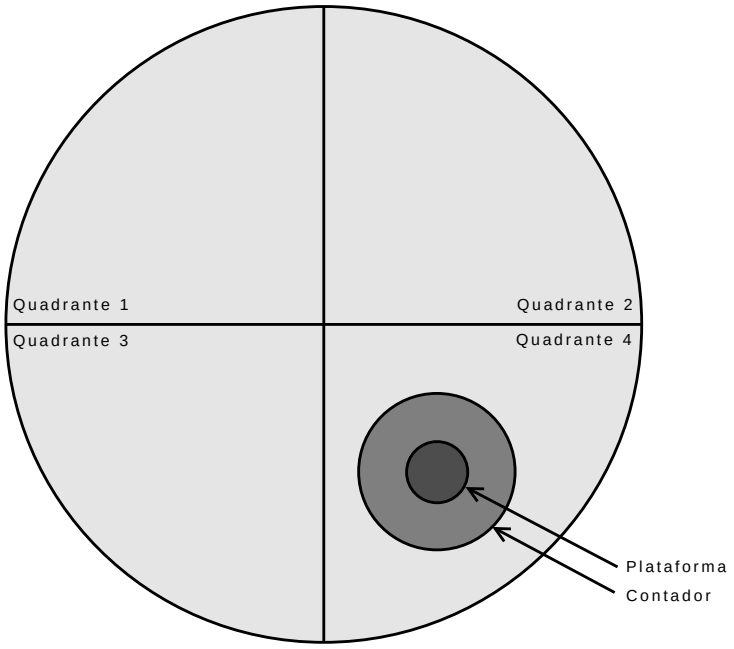



Figura 1.1: O labirinto aquático de Morris.

A figura 1.1 apresenta o modelo da piscina, para fins de coleta de parâmetros. Quadrantes são regiões formadas por duas linhas imaginárias que partem das bordas e se cruzam no centro da piscina, perpendicularmente. O contador é uma região circular, concêntrica à região da plataforma, apresentando um diâmetro três vezes maior que esta. Há vários parâmetros de interesse neste experimento. Os mais importantes são: tempo que o rato demora para encontrar a plataforma, distância nadada, velocidade, tempo em cada quadrante e número de passagens pelo contador. Este foi o tipo de labirinto inicialmente abordado em [8].

### 1.2.2 Teste de campo aberto

Este aparelho experimental é composto por uma arena para testar o efeito de ambientes não familiares à emocionalidade das cobaias (fig 1.2). Foi inicialmente descrito por Hall, em 1941, para testar os efeitos de ambientes não familiares na emotividade de ratos [11, 27]. Os parâmetros de interesse neste experimento são: distância percorrida, tempo em movimento, tempo parado, frequência em que a cobaia apresenta o comportamento de se levantar nas patas traseiras, tempo gasto até começar a se mover, etc.

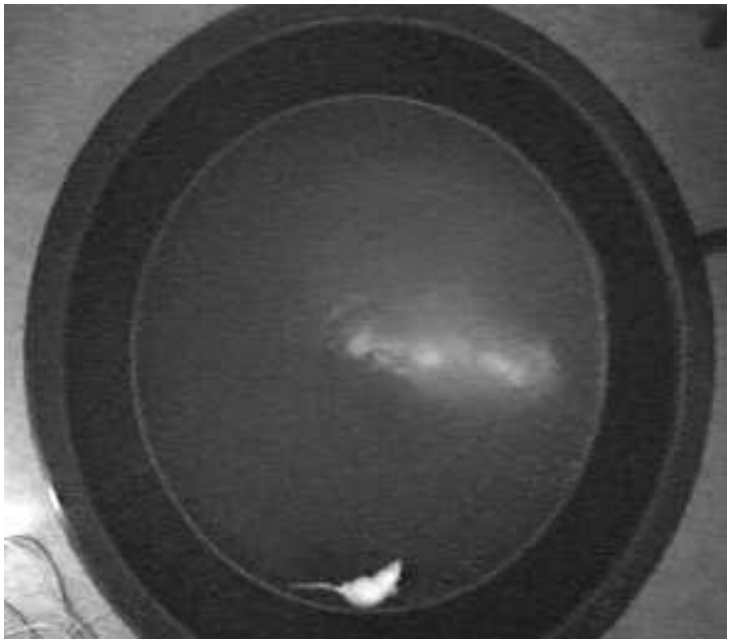


Figura 1.2: Teste do campo aberto.

### 1.2.3 Rastreamento de células do tecido nervoso

Neste tipo de experimento, são filmadas imagens microscópicas de células em um pedaço de tecido nervoso. Estas células se deslocam com a passagem do tempo. Como complicadores, elas mudam de forma e sofrem oclusão. O objetivo do rastreamento neste tipo de experimento é conseguir as coordenadas de cada célula ao longo do tempo, com o objetivo de estudar o comportamento migratório dentro do tecido. A figura 1.3 mostra um quadro de um vídeo deste experimento. As células são as regiões escuras na imagem. Como se pode observar, elas diferem em tamanho e em intensidade de brilho, o que dificulta a segmentação e o rastreamento.

## 1.3 Trabalhos correlatos

Poucos são os artigos que abordam o rastreamento de animais em experimentos. O trabalho apresentado em [3] descreve um método não robusto para coleta de parâmetros do *LAM*, pois realiza apenas uma busca exaustiva pela cabeça do roedor. Em [1], é proposto um sistema para rastrear múltiplas formigas em experimentos de laboratóio. O algoritmo contudo é pouco robusto, pois se utiliza da remoção de fundo (seção 3.1.5) para a segmentação das formigas, apresentando problemas quando a cor das formigas é próxima à cor de objetos do experimento. Não é tratado o problema da oclusão, e as formigas não são tratadas individualmente, isto é, sabe-se em cada momento quando há uma formiga em uma dada posição, mas não se sabe

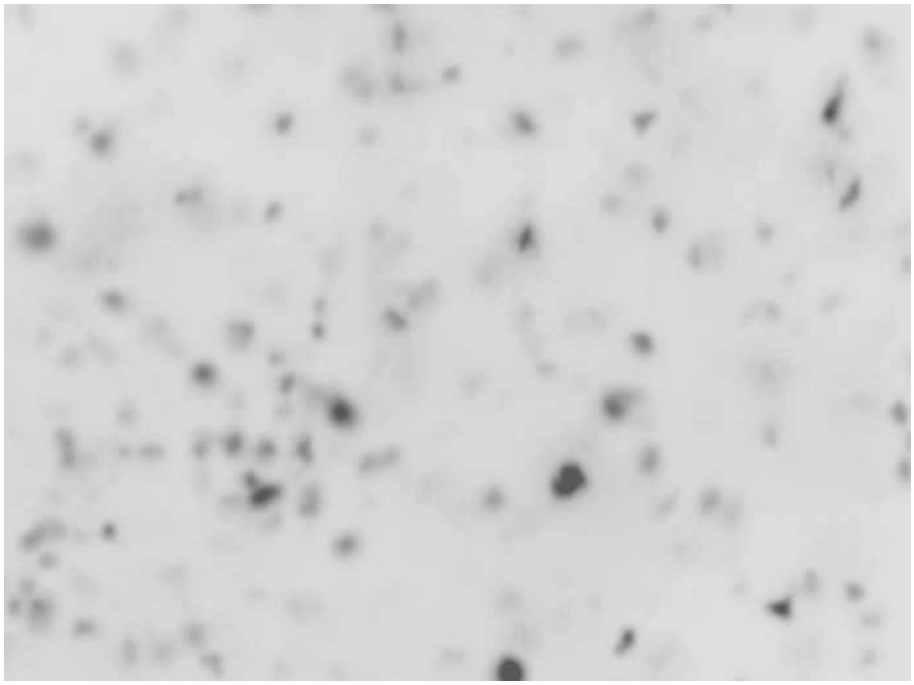


Figura 1.3: Conjunto de células em um experimento.

onde esta formiga se encontrava no quadro anterior. Nos trabalhos em [20, 19] é realizado o rastreamento de múltiplos ratos no experimento de campo aberto, com possibilidade de oclusão. O método proposto, contudo, supõe que o vídeo tem bom contraste e considera oclusão por contato, ou seja, quando os ratos estão muito próximos, não aparentando lidar com o caso em que os mesmos se sobrepõem na imagem. Além disso, não são definidos métodos para coleta de parâmetros deste experimento.

Como exemplo de um sistema comercial, pode-se mencionar o Ethovision que, de acordo com [29], é um sistema genérico que trabalha com tipos diferentes de experimentos, coletando poucos parâmetros mais específicos a um dado tipo destes. Além disto, o método de rastreamento implementado baseia-se apenas em técnicas bastante simples de extração de fundo, o que pode gerar facilmente problemas na detecção do alvo se, por exemplo, a água de uma piscina refletir a luminosidade ambiente. O modelo do projeto em questão não possui este inconveniente pois, além do contraste, leva em conta informações geométricas na identificação do alvo e pode ser empregado em ambientes menos controlados. Outro sistema comercial, Water 2020 [18] é um sistema especializado no LAM. Entretanto, além do seu alto custo, como é um sistema proprietário, não é possível adaptá-lo a outras necessidades nem incrementar suas funcionalidades básicas (introdução de novas operações específicas a uma dada experimentação).

O modelo de resolução de oclusões aqui cosiderado é baseado no modelo de representação em grafos apresentado em [12]. Este modelo é estendido e generalizado para que seja adequado

a diferentes tipos de rastreamento realizados pelo sistema desenvolvido.

O próximo capítulo apresenta o modelo proposto neste trabalho.

# Capítulo 2

# Modelo de rastreamento proposto

Neste capítulo será apresentado o modelo do sistema proposto neste projeto. Inicialmente, na seção 2.1 é dada uma visão geral deste modelo. A seção 2.2 aborda os aspectos relativos à leitura de vídeos e a seção 2.3 discute o tipo de rastreamento considerado pelo sistema e alguns exemplos de rastreadores propostos. Na seção 2.4 é apresentada a forma de armazenamento de dados, e na seção 2.5 é discutido o modelo de tratamento das oclusões. Finalmente, na seção 2.6 são apresentados os quantificadores, módulos do sistema proposto que têm papel fundamental na resolução destas oclusões. Detalhes sobre a implementação dos algoritmos usados pelo modelo em questão são apresentados no capítulo 3.

## 2.1 Visão geral do modelo

A figura 2.1 dá uma visão geral do modelo em questão. Cada caixa representa um módulo. As setas indicam a comunicação entre os módulos e a direção do fluxo de dados.

O sistema é subdividido nos seguintes módulos:

- *Gerenciador de vídeo*: responsável por ler um vídeo, separá-lo em quadros e carregá-los nas estruturas de dados do sistema.
- *Gerenciador de rastreadores*: realiza o rastreamento dos alvos e detecta as oclusões, sem todavia resolvê-las.
- *Gerenciador de dados*: armazena os dados gerados pelo módulo rastreador. Disponibiliza métodos para leitura e inserção de dados.
- *Resolvedor de oclusões*: resolve os problemas de oclusão, realizando as correções necessárias nos dados armazenados no gerenciador de dados.

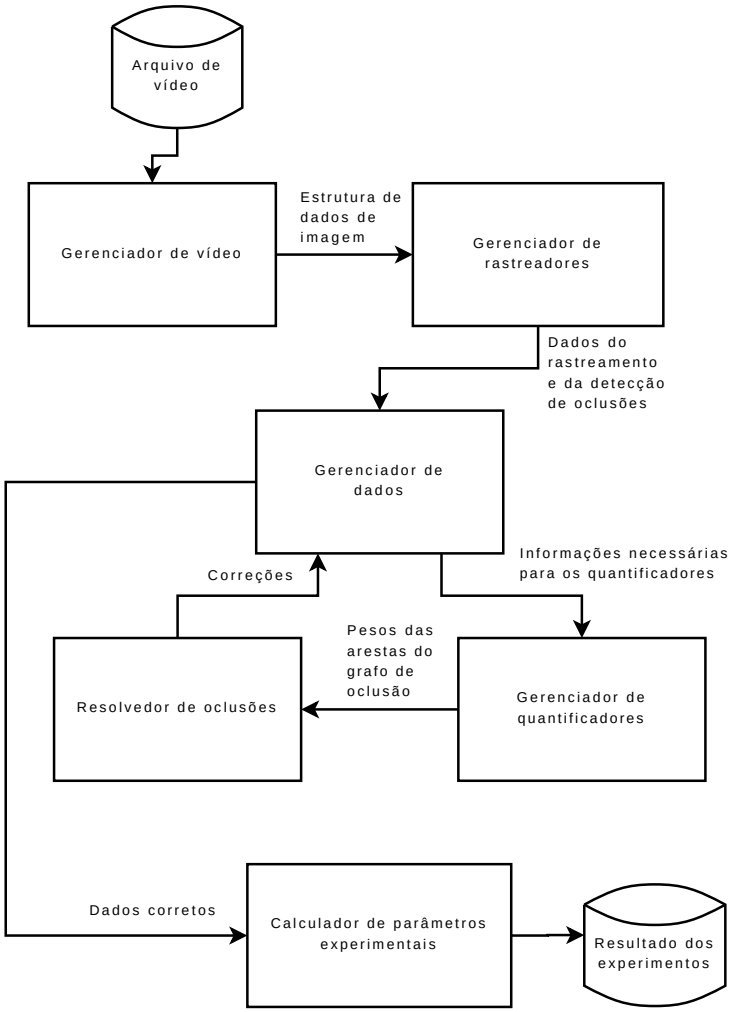



Figura 2.1: Modelo do sistema.

- *Gerenciador de quantificadores*: gerencia algoritmos que são utilizados pelo módulo resolvedor de oclusões.
- *Calculador de parâmetros*: gera um lista de parâmetros de interesse para o experimento realizado

Cada módulo é abordado mais detalhadamente nas próximas seções.

## 2.2 Gerenciador de vídeo

O módulo *Gerenciador de vídeo* é responsável por decodificar um vídeo e carregar seus quadros de acordo com as estruturas de dados do sistema. A principal função deste módulo é desacoplar

o sistema do formato do arquivo de vídeo. Deve haver uma implementação deste gerenciador para cada formato de vídeo utilizado.

## 2.3 Rastreadores

Aqui será discutido o módulo *gerenciador de rastreadores*. Será apresentada sua função, o tipo de rastreamento realizado, seus sub-módulos, etc. Na seção 2.3.1 é definido o tipo de rastreamento inicialmente realizado e na seção 2.3.2, a forma como os rastreadores são gerenciados. A seção 2.3.3 apresenta um rastreador que, utilizando-se de morfologia matemática, é robusto a ruídos e degenerações do vídeo. Na seção 2.3.4 é introduzido um rastreador que se utiliza de remoção de fundo para segmentar o alvo. A seção 2.3.5 apresenta os requisitos necessários a um rastreador de um único alvo para que o mesmo possa ser utilizado no rastreamento de múltiplos alvos. Na seção 2.3.6 é apresentado um módulo gerenciador de rastreamento capaz de rastrear múltiplos alvos, utilizando segmentação por remoção de fundo. Finalmente, a seção 2.3.7 apresenta um método de rastreamento baseado no casamento de regiões.

### 2.3.1 Tipo de rastreamento considerado

Este projeto inicialmente surgiu como uma solução para o rastreamento de cobaias em experimentos em biologia, mais especificamente o labirinto aquático de Morris e o campo aberto, mencionados na seção 1.2. Assim, os rastreadores apresentados nas próximas seções consideram que os alvos são gravados por uma câmera posicionada sobre os mesmos e que estes não somem do campo de visão. Além do mais, os rastreadores registram a posição do centróide de cada alvo. Em outras palavras, para todos os $n$ pontos $p_i = (x, y), (x, y) \in \Re^2, i \in \{1, ..., n\}$ pertencentes ao alvo, é registrada a posição $c = \frac{1}{n} \sum_{i=1}^{n} p_i$. Este rastreamento, apesar de restrito, é adequado para uma grande variedade de experimentos, incluídos os apresentados na seção 1.2.

Os alvos do rastreamento devem ser regiões. Para que seja possível a resolução de oclusões, estes alvos devem diferenciar-se entre si por algum conjunto de características. Estas características podem ser permanentes (como cor, tamanho e forma) podem ser variáveis (direção de deslocamento e velocidade) ou características de interação entre alvos (ausência de sobreposição, por exemplo).

Por último, a taxa de amostragem dos vídeos deve ser alta o suficiente para registrar os momentos de oclusão.

### 2.3.2 Gerenciador de rastreadores

Seja *rastreador de um alvo* um rastreador que segue apenas um alvo. O gerenciador de rastreadores é um sub-módulo que gerencia um conjunto de rastreadores de um alvo ou, em outra abordagem de implementação, realiza o rastreamento de todos os alvos. Este módulo tem como tarefas realizar um pré-processamento inicial do quadro, realizar a detecção de oclusões e organizar um conjunto de dados necessários ao rastreamento (seção 2.4).

O critério para determinar se dois alvos estão em oclusão é discutido na seção 2.5.1. Contudo, este módulo não tem a responsabilidade de solucionar as trocas de alvos entre os rastreadores geradas por problemas de oclusão, que são resolvidas em conjunto pelos módulos *Resolvedor de oclusões* e *Gerenciador de quantificadores*. Isto implica que o conjunto de dados, mencionados anteriormente, podem estar inconsistente, pois dados atribuídos a um alvo podem conter dados de outros alvos. Mais detalhes sobre as condições de rastreamento são apresentados na seção 2.3.5.

### 2.3.3 Rastreador de um alvo morfológico

Este exemplo de rastreador para um único alvo foi desenvolvido visando à obtenção de robustez em vídeos de baixíssima qualidade, que podem apresentar ruído gaussiano, variações de contraste, iluminação e foco. Para tanto, são considerados basicamente métodos de morfologia matemática. A figura 2.2 apresenta o modelo deste rastreador.

A cada iteração do rastreador, um quadro do vídeo é carregado. Para evitar ter que processar todos os pontos do quadro, utiliza-se uma janela de processamento. Apenas os pontos dentro desta janela são processados. Para posicionar a janela de processamento dentro do quadro de vídeo, utiliza-se um método de predição baseado no filtro de *Kalman* [13], usando a dinâmica do movimento linear. A seguir é realizado uma filtragem com aberturas e fechamentos morfológicos para redução de ruído e aumento de contraste, visando a um melhor desempenho das etapas posteriores. A seguir, é feita a detecção de contornos, utilizando-se o gradiente morfológico. Paralelamente, é realizada a criação de marcadores. Nesta etapa é importante que haja no mínimo um marcador dentro do alvo sendo rastreado. Utilizando a imagem de gradiente e os marcadores gerados nas etapas anteriores, o algoritmo de *watershed* (seção 3.1.3) segmenta a imagem em regiões, das quais uma delas é o alvo. Na última etapa, é realizada a seleção pela região com maior probabilidade de ser a região do alvo. Os algoritmos utilizados são descritos no capítulo 3.

Mais detalhes sobre este rastreador podem ser obtidos ainda em [8].

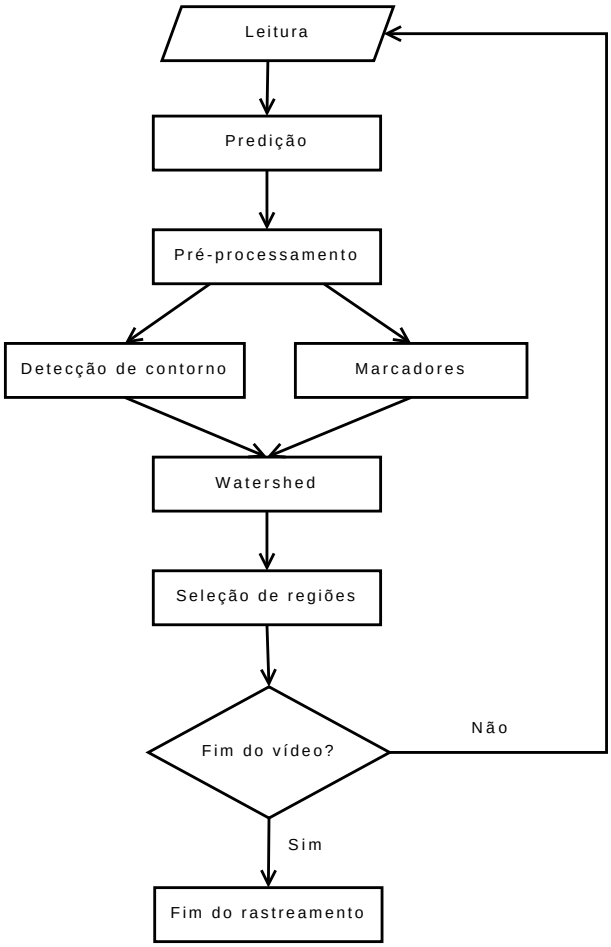



Figura 2.2: Modelo do rastreador morfológico.

### 2.3.4 Rastreador de um alvo por remoção de fundo

Este rastreador para um único alvo utiliza segmentação por remoção de fundo e algoritmos de busca em grafos para realizar o rastreamento do alvo. Em um dado quadro é realizada a segmentação por remoção de fundo (seção 3.1.5). Esta segmentação gera uma imagem binarizada, em que os pontos definidos como fundo ou *background* são representados pela cor preta e aqueles considerados como alvo, pela cor branca, por exemplo. A seguir, é realizada uma filtragem desta imagem para remoção de ruído. Na próxima etapa, é definido um grafo da seguinte forma: cada ponto da imagem que não é fundo é representado por um vértice. Um vértice $v$ representando um ponto $x$ tem arestas com todos os vértices $u$ representando um ponto $y$ tal que $y$ está na vizinhança de $x$. Neste grafo, então, são calculadas as componetes conexas (conjunto $C$ de vértices tal que, dados $u \in C$ e $v \in C$ quaisquer, existe ao menos um caminho ligando $u$ a $v$). A componente conexa que estiver mais próxima da posição do alvo no quadro anterior é

considerada o alvo neste quadro.

Este rastreador é adequado para o uso em vídeos que apresentem ruído branco, algo muito comum em gravações de experimentos. Isto se deve ao fato de que, na subtração de fundo, o alvo pode ser facilmente distinguido do ruído, devido à robustez do modelo estatístico.

### 2.3.5 Requisitos para a utilização de rastreadores de um alvo no rastreamento de múltiplos alvos

O rastreador de um alvo deve cumprir três requisitos para poder ser utilizado por um gerenciador de rastreadores. Seja $n$ o número de alvos a serem rastreados em um dado vídeo. Neste caso, deve haver $n$ instâncias de rastreadores de um alvo para rastreá-los. Como primeiro requisito, a cada passo do rastreamento, não deve haver alvo que não seja rastreado por ao menos um rastreador (um rastreador pode seguir mais de um alvo, em caso de oclusão). O segundo requisito é que um rastreador não pode se perder, isto é, rastrear algo que não seja um dos $n$ alvos definidos. O terceiro requisito é que um rastreador só pode trocar de alvo com outro na ocasião do término de uma oclusão.

Estes tres requisitos são a chave para o método de tratamento de oclusões da seção 2.5. O primeiro e o segundo requisito garantem que os todos dados de rastreamento sejam coletados. O terceiro requisito é fundamental para o tratamento de oclusão, já que não permite que os rastreadores troquem de alvo aleatoriamente. Isto implica menor inconsistência nos dados, simplificando assim o modelo de solução de oclusões.

### 2.3.6 Rastreador de múltiplos alvos por remoção de fundo

O gerenciador de rastreadores realiza a segmentação por remoção de fundo. Na imagem binarizada resultante, cada rastreador calcula os pontos que lhe pertencem. Intiutivamente, cada ponto associado a uma determinada região pertence ao rastreador mais próximo. Em outras palavras, um ponto $x$ pertence a um rastreador $tr$ em um quadro $i$ se e somente se $x$ está mais próximo da posição do alvo rastreado por $tr$ no quadro $i-1$ do que dos outros rastreadores. Na etapa seguinte, são excluídos os pontos *outliers* de cada alvo, de acordo com o método definido na seção 3.1.7. Este rastreador tem a vantagem de ser computacionalmente eficiente e robusto à oclusão por contato. Contudo, o cálculo do centro de cada alvo pode sofre interferência de ruídos de grande porte (como o reflexo do alvo nas paredes do equipamento usado no experimento ou a presençã de sombras).

### 2.3.7 Rastreador de múltiplos alvos por casamento de regiões

No rastreamento de células, um conjunto inicial de células de interesse é considerado. As demais células não rastreadas são considerados ruído, pois comprometem o desempenho dos

demais rastreadores. O algoritmo de rastreamento por casamento de regiões foi desenvolvido para lidar com este problema.

Para cada quadro, a imagem é segmentada as regiões obtidas são representadas por gaussianas bidimensionais. Para um destes quadros, o seguinte procedimento é realizado.

Seja $R = \{r_1, r_2, \ldots, r_n\}$ o conjunto das regiões segmentadas. Cada região $r_i$ é um conjunto de pontos: $r_i = \left\{p_1^i, \ldots, p_{n_i}^i\right\}$. Seja $T = \{t_1, t_2, \ldots, t_k\}$ o conjunto de rastreadores, cada um com seu centróide e matriz de covariância que representam seu alvo no quadro anterior. O centróide e a matriz de covariância de cada região são calculados. Seja $C^R = \{c_1^r, \ldots, c_n^r\}$ estes centros e $Cv^R = \{cv_1^r, \ldots, cv_n^r\}$ a covariância das coordenadas de cada região. Definimos uma função distância entre a região e o rastreador da seguinte maneira. Seja $cov_{i,j} = \{var_{xx}^{i,j}, var_{xy}^{i,j}, var_{yy}^{i,j}\}$ a matriz de covariância dos pontos da região $r_i$, calculados como se o centro dos pontos fosse o centro do rastreador $t_j$. A distância $d_{i,j}$ entre a região $r_i$ e o rastreador $t_j$ é então definida baseada nos valores $var_{xx}^{i,j}$ (variância da coordenada $x$ dos pontos de $r_i$), $var_{xy}^{i,j}$ (variância de $xy$), e $var_{yy}^{i,j}$ (variância de $y$).

$$d_{i,j} = var_{xx}^{i,j} + 2 \times var_{xy}^{i,j} + var_{yy}^{i,j}$$

Logo, cria-se um grafo bipartido completo. Uma das partições é o conjunto $R$ e a outra $T$. A distância entre regiões e rastreadores é empregada como a função de peso das arestas. Então, cada rastreador é casado com uma região usando-se algum dos algoritmos da seção 3.1.8. No fim de cada iteração, os centros e matrizes de covariância dos rastreadores são atualizados.

## 2.4 Gerenciador de dados

O Gerenciador de dados é o módulo do *framework* responsável pelo armazenamento dos dados gerados pelo Gerenciador de rastreadores, pelo acesso e gerenciamento de tais dados utilizados pelos módulos Resolvedor de oclusões e o Gerenciador de quantificadores. O gerenciador de dados mantém um grafo de oclusões (grafo que representa as oclusões ao longo do rastreamento) e, associados a cada vértice deste grafo, um conjunto de dados utilizados pelo Resolvedor de oclusões e o Gerenciador de quantificadores. Seja *grupo de oclusão* um conjunto de alvos em oclusão. Na implementação realizada para o projeto em questão, consideramos os três conjuntos de dados a seguir:

- Dados do tamanho dos grupos de oclusão:

  Uma lista com inteiros que indica a quantidade de alvos representada por cada vértice. Este dado é utilizado como restrição para a solução das oclusões (seção 2.5).

- Dados das coordenas dos grupos de oclusão:

Uma lista com coordenadas que indicam a posição dos alvos representados por cada vértice. Este dado é utilizado pelos quantificadores baseados em velocidade e deslocamento (seções 3.4.1 e 3.4.2).

- Dados dos recortes normais e recortes segmentados dos grupos de oclusão:

  Duas listas com imagens dos alvos representados por cada vértice. Uma das listas possui as imagens originais do alvo e a outra, imagem binária indicando quais pontos pertencem a alvos e quais pontos pertencem ao fundo da cena. Estes dados são utilizados para cálculos de descritores de regiões da imagem, como área e histograma. Tais descritores são utilizados pelos quantificadores baseados em área e histograma (seções 3.4.3 e 3.4.4, respectivamente).

## 2.5 Modelo de Resolução de oclusões

No modelo de rastreamento de múltiplos alvos proposto, é instanciado um único rastreador para cada alvo sendo rastreado. Contudo, quando dois alvos estão muito próximos ou se sobrepõem no vídeo, os rastreadores podem se confundir na discriminação dos alvos correspondentes. A figura 2.3 mostra um exemplo de oclusão. Neste caso, os rastreadores podem errar e, assim, trocar os alvos sendo rastreados (ver figura 2.4).

O Resolvedor de oclusões é o módulo responsável pelo tratamento das oclusões no sistema. Utilizando as informações calculadas pelo Gerenciador de quantificadores (seção 2.6), ele realiza as correções necessárias nos dados armazenados.

Esta seção introduz o modelo empregado no Resolvedor de oclusões. A seção 2.5.1 apresenta uma definição geral de evento de oclusão. A seção 2.5.2 define a forma de representação, pelo Resolvedor de oclusões, dos eventos de oclusão. A seção 2.5.3 aborda melhorias no grafo representando estes eventos e na seção 2.5.4 é dada uma definição relativa à solução das oclusões. Finalmente, a seção 2.5.5 trata do cálculo da qualidade de uma solução de oclusão.

### 2.5.1 Definição de oclusão

Para o tipo de rastreamento aqui considerado, a oclusão é definida a partir de um processo de segmentação de imagem. Suponha que um quadro do vídeo é segmentado utilizando a remoção de fundo (seção 3.1.5). Para este quadro, cada alvo é representado por uma região, como ilustra a figura 2.5. A oclusão ocorre quando os rastreadores, de alguma forma, não conseguem separar as regiões dos alvos. Outro critério que pode ser opcionalmente empregado pelo sistema baseia-se na proximidade entre alvos. Se a distância entre dois alvos for menor que um valor $d$, então considera-se a ocorrência de uma oclusão. Assim, ambos os critérios

Figura 2.3: Momento de oclusão entre dois ratos.

consideram a aproximação, contato ou sobreposição de alvos. Neste caso, não é tido como oclusão, por exemplo, o fato do alvo desaparecer do campo de visão.

A oclusão entre dois alvos ou, de maneira equivalente entre dois rastreadores, é uma relação reflexiva: se o alvo $a_i$ está em oclusão com $a_j$, então $a_j$ está em oclusão com $a_i$. Pode-se também definir a relação de transitividade para a oclusão, ou seja, se o alvo $a_i$ está em oclusão com $a_j$ e $a_j$ está em oclusão com $a_k$, então, por transitividade, $a_i$ está em oclusão com $a_k$. Contudo, como veremos na seção 2.5.3, essa pode não ser uma definição muito adequada para alguns tipos de rastreamento.

### 2.5.2 Representação dos eventos de oclusão

Nesta seção será apresentada a forma de armazenamento no sistema das informações referentes aos eventos de oclusão. O módulo Gerenciador de rastreadores (seção 2.3.2) é responsável por detectar os eventos oclusão. Um *grupo de oclusão* $og_i = \{r_1, r_2, ..., r_k\}$ contém os $k$ rastreadores tal que os alvos rastreados por $r_i$ e $r_j$ estão em oclusão, para $i, j \in [1, k], i \neq j$.

O conjunto $OG_t = \{og_1, og_2, ..., og_n\}$ representa todos os grupos de oclusão ocorridos no instante $t$ do vídeo. Todas os rastreadores devem pertencer ao menos a um grupo de oclusão. Um rastreador que não está em oclusão está em um grupo de oclusão de tamanho 1. A partir

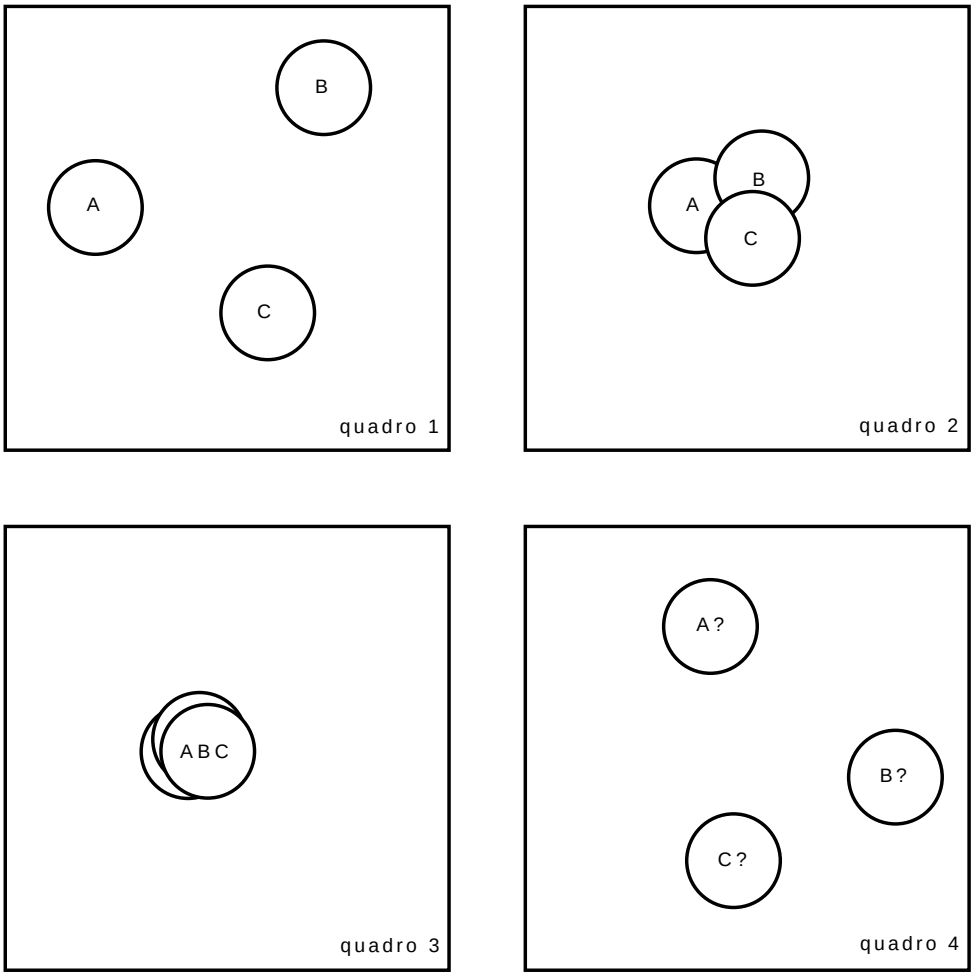



Figura 2.4: Ilustração de possível troca de alvos entre rastreadores.

destas informações é criado um grafo que representa a ocorrência de oclusões durante todo o processo de rastreamento. Tal grafo é um grafo direcionado acíclico e formado por camadas. A primeira camada representa o estado inicial dos alvos, e cada uma das camada seguintes representa os grupos de oclusão existentes em um quadro do vídeo. Existirá uma camada para cada conjunto não vazio $OG_t$ , $t = 1, \ldots, n_q$, onde $n_q$ é o número de quadros do vídeo. Há na *i-ésima* camada, um vértice para cada grupo de oclusão que pode ser distinguido pelo Gerenciador de rastreadores durante a etapa de segmentação. As arestas existem apenas entre as camadas $i$ e $i + 1$, indicando a movimentação dos alvos entre os grupos de oclusão. Uma aresta $e_{uv}$ entre os vértices $u$ e $v$ indica que ao menos um dos rastreadores presentes no grupo de oclusão representado por $u$ está presente no grupo de oclusão representado por $v$.

A figura 2.6 ilustra a geracão deste grafo. Os primeiros $6$ quadros (figura 2.6.a) mostram um vídeo de um experimento com $4$ alvos. Os seguintes $6$ quadros (figura 2.6.b) mostram os

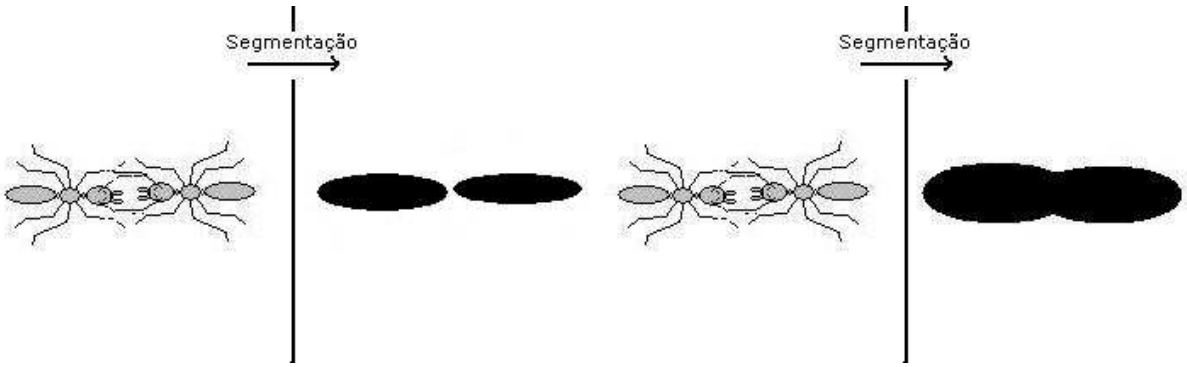



Figura 2.5: Segmentação de dois alvos em duas regiões e em uma única região.

grafos auxiliares de oclusão, onde cada vértice representa um rastreador e uma aresta entre dois vértices representa a oclusão entre os dois rastreadores. O quadro final (figura 2.6.c) representa o grafo de oclusões, com os números representando o tamanho dos grupos de oclusões representados pelos vértices. Note-se que cada camada do grafo de oclusões representa um quadro do vídeo. No primeiro quadro, os $4$ alvos não estão em oclusão. No segundo quadro, os rastreadores $a$ e $c$ entram em oclusão, o que é representado no grafo auxiliar. Neste instante, há $3$ grupos de oclusão, $\{a, c\}$, $\{b\}$ e $\{d\}$. As duas arestas que incidem no segundo vértice da segunda camada indicam a oclusão entre duas componentes. Na terceira camada, os rastreadores $a$ e $c$ saem da oclusão. Isso é representado pelas duas arestas saindo do segundo vértice da segunda camada. Em outras palavras, as arestas representam os possíveis grupos de oclusão pelos quais os alvos passaram. As outras camadas são geradas de maneira análoga.

### 2.5.3 Otimização do grafo auxiliar de oclusões

Como visto na seção anterior, o grafo auxiliar de oclusão é um grafo que indica quais rastreadores se encontram em oclusão em um dado quadro. Nele, existe um vértice para cada rastreador e uma aresta entre dois vértices indica a ocorrência de oclusão entre os rastreadores. Este grafo é usado para a detecção dos grupos de oclusão. Há várias formas possíveis de se definir o que é um grupo de oclusão. Uma delas é a utilização da propriedade de transitividade para a relação de oclusão (ver seção 2.5.1) e considerar as componentes conexas do grafo como grupos de oclusão. Contudo, tomemos como exemplo um experimento de rastreamento de formigas, como o estudado em [1] e exemplificado na figura 2.7. Neste caso, supomos que uma formiga está em oclusão com formigas que se encontram muito próximas. Se a relação de transitividade é utilizada, quase todas as formigas estarão em oclusão entre si. Isto contudo não é verdade pois dificilmente rastreadores que se encontram distantes trocariam de alvo, o que originaria mais complicações para a etapa de resolução de oclusões, dado que o grafo de oclusões (seção 2.5.2) ficaria mais complexo (entendendo-se por complexidade vértices com grau desnecessariamente

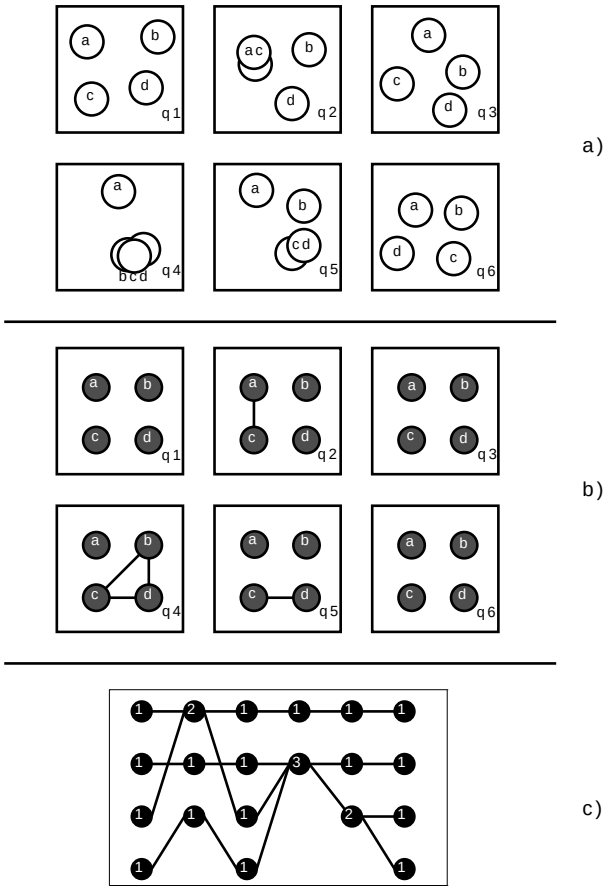



Figura 2.6: Vídeo do rastreamento (a), grafos auxiliares de oclusão (b) e o grafo de oclusões resultante.

alto).

Logo, sem a utilização de transitividade, todos os eventos de oclusão teriam cardinalidade 2, pois os critérios utlilizados para detectar oclusões pelo rastreador (ver seção 2.5.1) levam em conta a oclusão entre pares de alvos. Contudo, com esta abordagem, cada camada do grafo de oclusões pode ter até $\frac{k.(k-1)}{2}$ (combinação de $k$, 2 a 2) vértices, onde $k$ é o número de alvos sendo rastreados. Pode-se, contudo, diminuir o tamanho deste grafo utilizando-se uma etapa de otimização cujo objetivo é aumentar o tamanho dos grupos de oclusão. Como definido na seção 2.5.2, um grupo de oclusão é composto por um conjunto de rastreadores que estão em oclusão entre si. Logo, no grafo auxiliar de oclusão, cada clique representa um grupo de oclusão.

Portanto, para maximizar o tamanho dos grupos de oclusão e assim minimizar-se o número de vértices em uma dada camada do grafo de oclusões, deve ser feita, no grafo auxiliar, uma cobertura de arestas por cliques de cardinalidade mínima. Esta cobertura é definida por um conjunto $C$ de cliques tal que cada aresta pertence ao menos uma clique de $C$. Cada clique representa um vértice no grafo de oclusões. A idéia por trás desta otimização é diminuir a re-

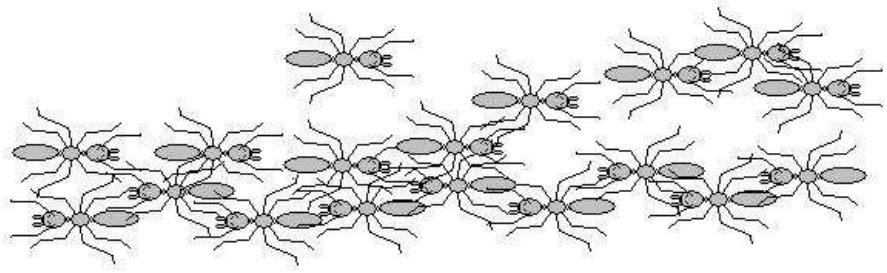


Figura 2.7: Formigas em um experimento.

dundância no grafo de oclusões, diminuindo assim seu tamanho, otimizando o uso de memória, algo importante considerando que os grafos de oclusões são grandes.

Na seção 3.2 são apresentadas heurísticas para a otimização do grafo em questão.

### 2.5.4 Resolução das oclusões

Após a conclusão do processo de rastreamento, tem-se um grafo que representa todos os eventos de oclusão, como discutido na seção 2.5.2. Nesta seção será definida uma solução para o problema da resolução de oclusões neste grafo.

Os grupos de oclusão pelo qual um alvo passou ao longo do processo de rastreamento são representados por um caminho da primeira até a última camada do grafo. Isso ocorre porque cada vértice representa um grupo de oclusão (portanto, representa ao menos um alvo) e as arestas entre as camadas representam a união e separação destes grupos.

De posse desta informação, seja $Cap(v)$ a capacidade de um vértice $v$, isto é, o número de alvos representados por este vértice. Define-se, como uma solução para o rastreamento, um conjunto de caminhos que vão de todos os vértices da primeira camada a última camada, dado que o número de caminhos passando por um vértice $v$ é menor ou igual ao valor $Cap(v)$. Esta é uma restrição de fluxo, em que as capacidades estão associadas aos vértices do grafo.

A figura 2.8 mostra um exemplo de solução de oclusões. Em 2.8(a), temos um grafo de oclusões, com cada vértice etiquetado com com o valor de sua capacidade. Logo, nas figuras

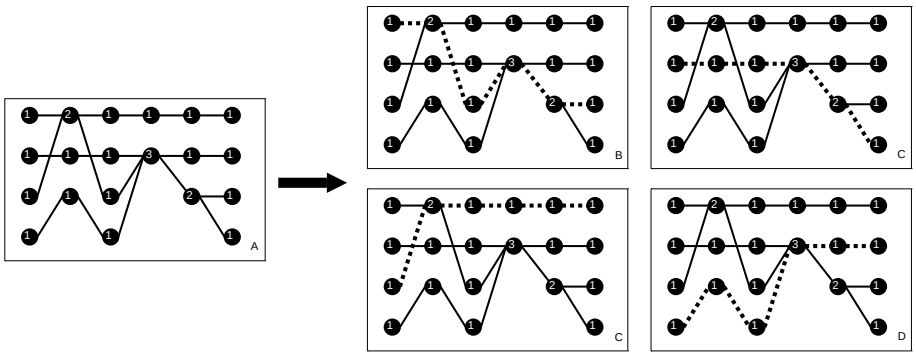



Figura 2.8: Um grafo de oclusões (a) seguido de uma solução composta por quatro caminhos (b,c,d,e).

2.8(b), 2.8(c), 2.8(d) e 2.8(e) há uma solução composta por $4$ caminhos que respeitam a restrição de capacidade dos vértices (um vértice não é utilizado por um número de caminhos maior que sua capacidade).

### 2.5.5 Qualidade de uma solução

Dadas as definições da seção anterior, pode haver um grande número de soluções de diferentes qualidades para o problema da resolução de oclusões abordado anteriormente. Cada aresta no grafo de oclusões possui um conjunto associado de valores correspondente ao seu peso. Cada caminho entre a primeira e a última camada possui um peso total dado pela soma dos pesos das arestas dos caminhos. O peso de cada aresta é calculado em função do caminho iniciado na primeira camada e que chega até ela. Seja $path_i = \{v_0, \ldots, v_l\}$ o caminho no grafo de oclusões referente ao $i$-ésimo alvo sendo rastreado. O peso de $p_i$ é definido então como:

$$wgt_{pi} = \sum_{j=2}^{l} pr\left(e_{v_{j-1},v_j} | v_0, v_1, \ldots, v_{j-1}\right) \tag{2.1}$$

Onde $l$ é o número de quadros do vídeo (ou, de forma equivalente, o número de camadas do grafo de oclusões), $e_{v_{j-1},v_j}$ é a aresta que liga os vértices $v_{j-1}$ e $v_i$ e $pr\left(e_{v_{j-1},v_j} | v_0, v_1, \ldots, v_{j-1}\right)$ é a probabilidade da aresta $e_{v_{j-1},v_j}$. Esta probabilidade é calculada pelos algoritmos quantificadores (seção 2.6).

O peso total da solução é dado então como a somatória dos pesos dos caminhos referentes a todos os $k$ alvos.

## 2.6 Gerenciador de quantificadores

O Gerenciador de quantificadores é o módulo do sistema que gerencia e combina os quantificadores, auxiliando no processo de resolução de oclusões. Quantificadores são um conjunto de algoritmos utilizados no cálculo do peso das arestas do grafo de oclusões. Seja $e_{u,v}$ uma das arestas deste grafo unindo os vértices $u$ e $v$. Um quantificador deve indicar a probabilidade $pr_i\,(a, r_i, r_j)$ de um alvo $a$ que se encontrava na região $r_i$, num instante $t$, encontrar-se na região $r_j$, no instante $t + 1$. Esta probabilidade é utilizada como peso das arestas do grafo de oclusões (seção 2.5), que representam as transições dos alvos entre regiões durante o rastreamento. Alguns exemplos de quantificadores são apresentados na seção 3.4. Vários destes quantificadores podem ser empregados ao mesmo tempo, em função do tipo de rastreamento. Um método para a combinação dos quantificadores será discutido na seção 3.5.

O próximo capítulo apresenta o conjunto de algoritmos utilizados pelos modúlos do sistema.

# Capítulo 3

# Algoritmos e procedimentos utilizados

Esta seção discute os principais aspectos dos algoritmos relacionados ao sistema proposto. Estes algoritmos envolvem métodos de processamento de imagens, grafos, heurísticas e meta-heurísticas para otimização, entre outros. A seção 3.1 apresenta os algoritmos referentes aos rastreadores implementados e a seção 3.2 as heurísticas para a resolução do problema da cobertura mínima de arestas por cliques cuja solução serve para otimizar o grafo representando oclusões, como discutido na seção 2.5.3. Na seção 3.3 é apresentado um método heurístico empregado na solução das oclusões. A seção 3.4 ilustra algumas implementações de quantificadores. A seção 3.5 apresenta um método para a combinação dos resultados de quantificadores distintos e a seção 3.6 apresenta a prova de que o problema da oclusão, da forma apresentada neste trabalho, é *NP-Difícil*.

## 3.1 Algoritmos dos rastreadores

Nesta seção, são descritas as transformações de imagem, os operadores morfológicos e outros métodos empregados no trabalho aqui proposto.

Sejam $f, g : \mathbb{Z}^2 \rightarrow \mathbb{Z}$ imagens em níveis de cinza, $B$ um elemento estruturante, $\xi_B(f)$ e $\delta_B(f)$ as operações de erosão e dilatação morfológicas, respectivamente. Seja, ainda, $\gamma_B(f)$ a abertura morfológica da imagem $f$. O seguinte conjunto de transformações pode ser definido a partir destes operadores. Uma descrição destes operadores pode ser encontrada em [28, 17, 2].

### 3.1.1 Aumento de contraste

Este operador realiza um realce de contraste, empregando para isto a dilatação e erosão morfológicas. Gera-se uma imagem realçada $g$ a partir de $f$, de acordo com as seguintes relações [28]:

$$g(x,y) = \begin{cases} \xi_B(f)(x,y), \text{se} f(x,y) - \xi_B(f)(x,y) < \delta_B(f)(x,y) - f(x,y); \\ \delta_B(f)(x,y), \text{se} f(x,y) - \xi_B(f)(x,y) > \delta_B(f)(x,y) - f(x,y); \\ f(x,y), caso\ contrário. \end{cases}$$

Este operador possui uma atuação local menos sensível a variações de brilho no ambiente. Ele é empregado na etapa de pré-processamento do rastreador morfológico abordado na seção 2.3.3 para facilitar o processamento de vídeos de baixa qualidade. Na figura 3.1 pode-se ver o efeito do mesmo, considerando-se um elemento estruturante planar de dimensão $7 \times 7$.

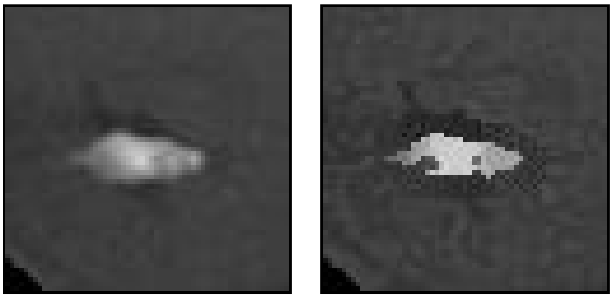



Figura 3.1: Uma imagem normal (a) e após um realce de contraste (b).

### 3.1.2 Gradiente Morfológico

A idéia básica da detecção de contornos consiste em localizar descontinuidades de intensidade na imagem [28]. Na literatura, pode-se encontrar uma grande variedade de métodos para a detecção de contornos. Um método bastante simples é o gradiente morfológico, definido por:

$$g(x,y) = (\delta_B(f)(x,y)) - (\xi_B(f)(x,y)) \tag{3.1}$$

Este operador é empregado no rastreador morfológico abordado na seção 2.3.3. A figura 3.2 mostra o gradiente morfológico de uma imagem, utilizando-se um elemento estruturante planar de dimensão $3 \times 3$.

### 3.1.3 Linhas Divisoras de Águas (*Watersheds*)

Uma das ferramentas morfológicas mais importantes em segmentação de imagens é a tranformação de linhas divisoras de águas (*watersheds*) [4, 7]. Uma definição intuitiva deste operador é dada considerando-se a imagem $f$ como uma superfície topográfica. Esta superfície apresenta máximos regionais. Supondo-se que estes máximos sejam perfurados, a superfície é submersa

(a) (b)

Figura 3.2: Uma imagem (a) e seu gradiente morfológico (b).

progressivamente em água, de tal modo que o seu nível continue o mesmo em toda a extensão da superfície. Durante o processo de inundação, são construídos diques para evitar a união de águas provenientes de dois máximos diferentes. Ao final, o conjunto dos diques representando fronteiras entre estes máximos constitui a linha divisora de águas da imagem. No projeto em questão, visando evitar uma supersegmentação, foi implementada também a extensão do algoritmo de *watershed* que considera apenas aqueles máximos locais escolhidos como marcadores [4]. Este algoritmo foi empregado como forma de segmentação no rastreador morfológico (seção 2.3.3) e, além disso, na limiarização de bacias de watershed, apresentada na seção 3.1.6 e utilizada pelo rastreador por casamento de regiões (seção 2.3.7). A figura 3.3 mostra a linha divisoras de águas de uma imagem de um conjunto de células, empregando estas como marcador.

### 3.1.4 Cálculo de Dinâmica

Geralmente, os extremos de uma imagem, mínimos ou máximos locais, possuem informações importantes para sua análise. Contudo, a filtragem dos extremos de maior importância não é trivial. A dinâmica é uma característica que permite quantificar um extremo de acordo com seu contraste [15]. A dinâmica de um máximo local $M$ de $f$ é definida como o desnivelamento máximo a ser atingido quando, partindo-se de $M$, se deseja alcançar um ponto de maior altitude.

$$\text{Seja } C = (p_0, \ldots, p_n)$$

$$din(M) = f(M) - sup_{p_0 \in M, f(p_n) > f(p_0)}\{inf\{f(x), x \in C\}\} \tag{3.2}$$

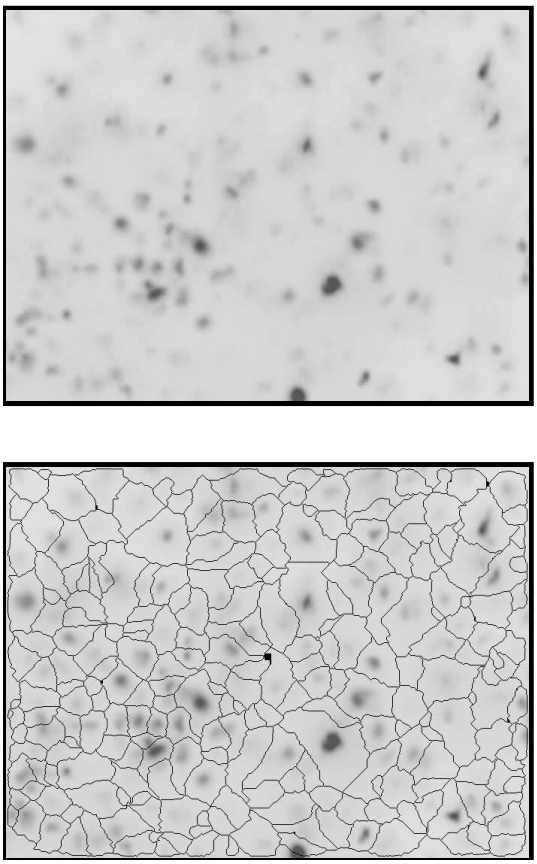



Figura 3.3: Uma imagem (a) e suas linhas de watershed (b).

Em [15] é apresentado um algoritmo para o cálculo da dinâmica que segue o mesmo princípio de inundação utilizado no algoritmo de *watershed*. Nesse algoritmo, inverte-se a imagem e calcula-se a dinâmica do máximo local $m$ da seguinte forma:

$$din(m) = h - f(m),$$

em que $h$ é o desnivelamento máximo entre $m$ e um máximo regional mais profundo. Inicialmente, cada máximo recebe uma etiqueta diferente. Durante a inundação, quando a água de dois máximos se encontram, ao invés de se construir diques como no algoritmo de watershed, é calculada a dinâmica do menor máximo, segundo a equação acima, e toda a região é etiquetada com o valor do máximo mais profundo (como se um lago absorvesse o outro). Ao final, apenas uma etiqueta associada ao máximo mais profundo é obtida. Esse máximo recebe o valor máximo possível de dinâmica (neste caso, $255$ para imagens com $2^8$ níveis de cinza). Este é o método empregado para a geração de marcadores para a segmentação por *watershed* (seção anterior) utilizado pelo rastreador morfológico (seção 2.3.3).

### 3.1.5 Remoção de fundo

A segmentação por remoção de fundo é um método eficiente para se separar o fundo do alvo sendo rastreado. Existem várias formas de se realizar a remoção de fundo. A que será empregada neste trabalho é apresentada a seguir.

Inicialmente, necessitamos de um modelo do fundo que pode ser estático ou pode se adaptar dinamicamente. Pelas características dos vídeos de experimentos de biologia, gravados em ambientes controlados, escolheu-se o modelo estático de fundo. Para a segmentação, realiza-se uma subtração matricial entre uma imagem e o respectivo fundo. Teoricamente, valores nulos em pontos indicam que estes pertencem ao fundo da imagem considerada, e os valores diferentes de zero, pontos que não pertencem. Contudo, na presença de ruído branco, por exemplo, este critério torna-se impreciso. Para lidar com este problema, é feita uma modelagem estatística do ruído, considerando que ele apresenta uma distribuição gaussiana. Neste caso, são calculados a média e o desvio padrão da imagem subtraída. Como a imagem é muito maior que o alvo, os valores do alvo constituem *outliers*, i.e., anomalias, na distribuição do ruído. A seguir, os valores que estão a menos de $k$ desvios padrões da média da gaussiana são identificados como fundo e os demais são considerados alvo. Feito isso, é realizada uma abertura morfológica para eliminar o ruído irrelevante. O valor de $k$ utilizado no sistema, após uma séria de testes, é $3$.

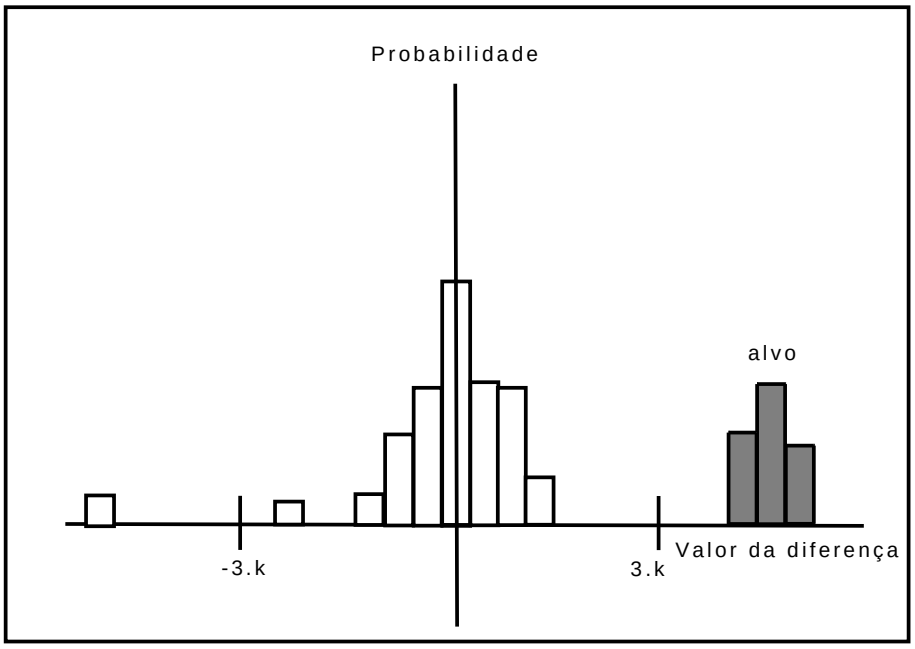



Figura 3.4: Exemplo de histograma da remoção de fundo.

A figura 3.4 mostra um exemplo de histograma resultante da remoção de fundo. Tudo que está dentro do intervalo $[-3k, 3k]$ é considerado fundo.

Esta abordagem apresenta problemas referentes às sombras e aos reflexos dos alvos na imagem que, quando presentes, são detectados igualmente como alvos. Estes problemas, que devem ser considerados pelos rastreadores que utilizam este tipo de segmentação (seções 2.3.4, 2.3.6 e 2.3.7 ), podem resultar em um rastreamento impreciso. Uma forma simples de lidar com o problema das sombras é recorrer a uma boa iluminação (os vídeos de testes foram filmados por sobre o experimento, o que elimina este tipo de problema). O problema de reflexo nas imagens será abordado posteriormente.

### 3.1.6 Segmentação por limiarização de bacias de Watershed

A definição de um limiar global que segmente um conjunto de alvos pode ser inviável em diversas aplicações de processamento de imagens. Por exemplo, no caso do experimento com células (seção 1.2.3), devido à forma do histograma das imagens, é impossível determinar um limiar global que consiga segmentar corretamente todas as células. Como ilustrado na figura 3.5(c), apenas as células são segmentadas por uma limiarização global. Para este tipo de rastreamento, aplicou-se um método de limiarização local. Inicialmente, a imagem é filtrada por aberturas e fechamentos morfológicos com o objetivo de remover mínimos locais irrelevantes [28]. Em seguida, a zona de influência [28] de todos os mínimos locais é calculada, a partir da definição das linhas de watershed da imagem original, o que resulta no conjunto de regiões disjuntas, tal como ilustrado na figura figura 3.5(a). Finalmente, um limiar estatístico é calculado para cada uma das subregiões obtidas. A figura 3.5(c) apresenta o resultado de uma segmentação considerando-se este método. O limiar estatístico é o mesmo empregado na remoção de fundo (seção 3.1.5).

### 3.1.7 Remoção de outliers

Esta seção apresenta um método simples para a remoção de *outliers* em um conjunto de pontos no espaço, dado que este conjunto apresenta distribuição gaussiana. Este método utiliza a mesma idéia da remoção de *outliers* para valores escalares. Inicialmetne, é calculada a distância média entre os pontos e a média destes. Todos os pontos que estiverem a uma distância $k$ vezes maior que a distância média são considerados *outliers* (anomalias). A figura 3.6 demonstra este procedimento em que os pontos negros representam *outliers*. Este método é utilizado pelo rastreador de múltiplos alvos por remoção de fundo (seção 2.3.6).

### 3.1.8 Algoritmos para o problema do casamento de regiões

Para o rastreador por casamento de regiões (seção 2.3.7), deve-se utilizar um algoritmo de casamento de grafos bipartidos. Quando o grafo possiu arestas com pesos constantes, há algoritmos rápidos polinomiais para o casamento [6]. Contudo, para o *framework* em questão, o peso pode

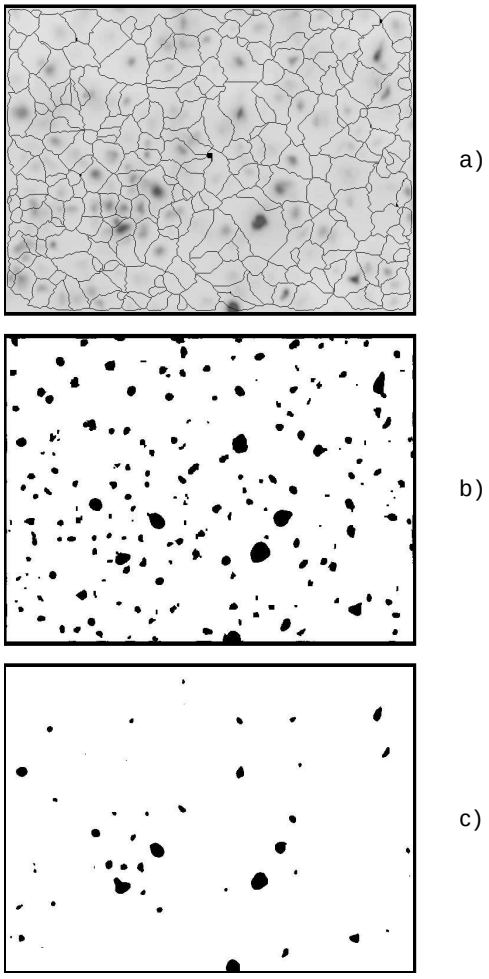



Figura 3.5: Linhas de watershed da imagem original (a), limiar por bacias de watershed (b) e limiar global que segmenta o maior número de células (c).

variar e o casamento de arestas permite poligamia, *i.e.*, os vértices de uma das partições podem ser casados com mais de um vértice da outra. Metaforicamente, um *marido* (uma região) pode ter várias *esposas* (rastreadores) mas uma esposa deve ter apenas um marido. Na figura 3.7, temos um exemplo deste tipo de casamento. Os rastreadores são representados pelo conjunto $\{r_1, r_2, r_3, r_4\}$ e as regiões pelo conjunto $\{a_1, a_2, a_3, a_4\}$. Um possível casamento é representado pelas arestas grossas. Nota-se que uma região, quando representa um grupo de alvos em oclusão, pode estar casada com mais de um rastreador (alvo $a_1$) e, no caso de representar um alvo não rastreado ou ruído, casada com nenhum rastreador (alvo $a_4$).

Formalmente, temos dois conjuntos de vértices: $R = \{r_1, \ldots, r_m\}$, representando os rastreadores e $A = \{a_1, \ldots, a_n\}$, representando as regiões dos alvos na imagem. Seja $G_{bp}$ o grafo bipartido completo cujas partições são $R$ e $T$. Seja $MA = (ma_1, \ldots, ma_m)$, $ma_i \in A$,

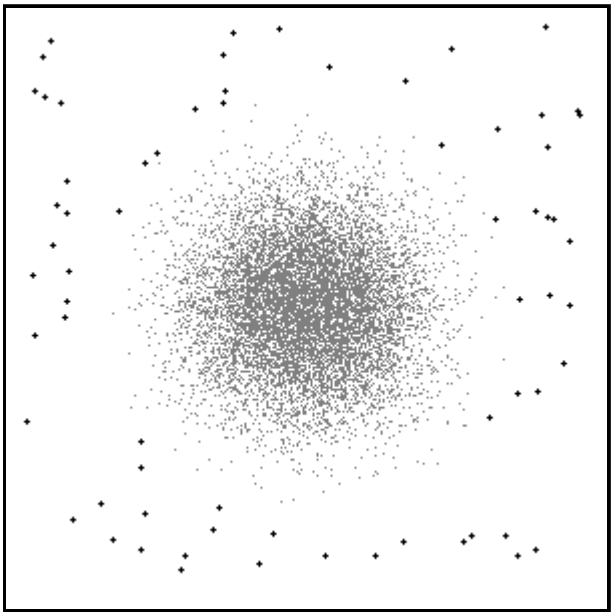


Figura 3.6: Detecção de *outliers* de um conjunto de pontos no plano.

$i \in [1, m]$ um casamento de elementos de $R$ e $A$. Por fim, seja $f_w(R, A, MA)$ uma função que retorna um valor real denotando quão bom é o casamento entre as regiões e os rastreadores. Esta função é definida como:

$$f_w(R, A, MA) = \sum_{i=1}^{m} \left(d_{r_i, ma_i}\right)^{-1} + f\left(MA\right), \tag{3.3}$$

em que $d$ é a medida de distância entre um rastreador e uma região, definida na seção 2.3.7 e $f\left(MA\right)$ uma função que calcula penalidades com base no casamento realizado. Estas penalidades podem variar entre implementações diferentes. No caso do rastreador por casamento de regiões já mencionado, as penalidades são dadas pelo número excessivo de rastreadores casados com uma mesma região. Esta função é definida a seguir.

$$oclij = \begin{cases} \frac{d_{r_i, ma_i} + d_{r_j, ma_j}}{d_{r_i, ma_i} . d_{r_j, ma_j}} \text{se os alvos rastreados por i e j estão em oclusão;} \\ 0, \text{caso contrário.} \end{cases} \tag{3.4}$$

$$f\left(MA\right) = \sum_{i=1}^{m} \sum_{j=1}^{m} ocl(i, j) \tag{3.5}$$

Esta função atribui penalidades que crescem de acordo com o tamanho dos grupos de oclusão, evitando situações de erro quando alguns alvos são seguidos por vários rastreadores enquanto outros não são rastreados.

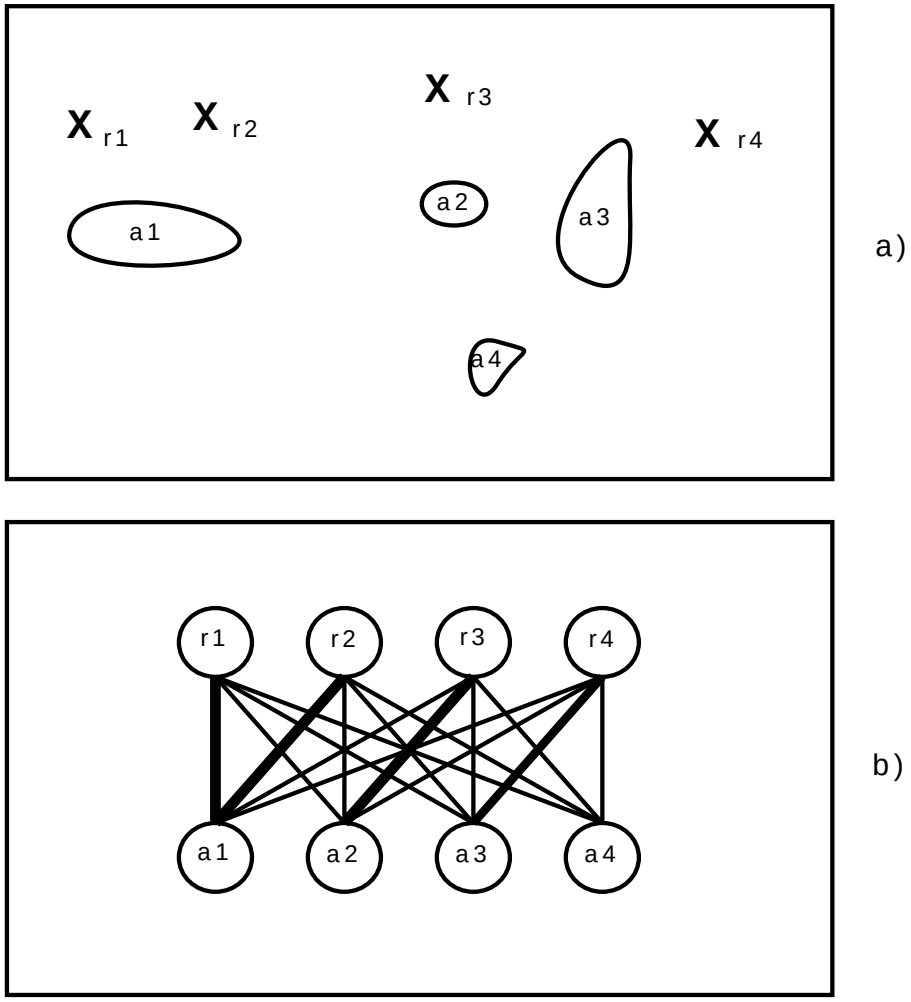



Figura 3.7: Rastreadores (marcados com *x*) e regiões da imagem (a); um exemplo de casamento (b).

Uma heurística foi implementada para solucionar o problema em questão. Seu algoritmo é composto por duas etapas: uma gulosa e outra de busca local. Na etapa gulosa, cada rastreador é casado com a região com maior similaridade, i. e., o rastreador $r_j$ se casa com a região $a_j$ que maximize o valor $d_{r_i,a_j}$. Na busca local, para cada rastreador, troca-se sua região casada por outra que incremente o valor de $f_w(R, A, MA)$. A troca é repetida até que nenhum incremento possa ser alcançado.

A etapa gulosa não garante que a equação 3.3 seja maximizada, devido às penalidades. Se aplicada isoladamente, ela tende a gerar maus casamentos, casando um número grande de rastreadores com uma única região. A busca local realiza melhoras no casamento visando eliminar este problema.

## 3.2 Heurísticas para a otimização do grafo de oclusões

A solução ótima para a otimização do grafo de oclusões considerada na seção 2.5.3 consiste da geração de um grafo com o menor número de vértices possível. Fazer isso, nas condições dadas, é solucionar o problema da cobertura de arestas por clique de cardinalidade máxima. Isso é um problema *NP-Difícil* [14]. Uma possível heurística para esse problema é a seguinte: encontra-se a maior clique do grafo auxiliar. A seguir, retiram-se suas arestas do grafo. Repete-se isso até que o grafo esteja com o conjunto de arestas vazio. O problema de encontrar a maior clique também é conhecidamente um problema *NP-Difícil*. Logo, utiliza-se um conjunto de heurísticas para a escolha desta maior clique. Seja $G = (V, E)$ grafo simples. Neste trabalho, foram consideradas as seguintes heurísticas:

- **Heurística gulosa crescente:** Seja $W$ um conjunto de vértices. No início do algoritmo, $W \leftarrow \{\}$. A cada passo da heurística, insere-se neste conjunto o vértice $i$ que deve satisfazer a condição de que $W \leftarrow W \cup \{i\}$ é uma clique. De todos os vértices possíveis, escolhe-se o de maior grau. Termina-se quando não houver nenhum vértice que possa ser inserido em $W$.

- **Heurística gulosa decrescente:** De maneira análoga, seja $W$ um conjunto de vértices. No início do algoritmo, temos $W \leftarrow V$. Enquanto $W$ não for uma clique, retira-se de $W$ o vértice com menor grau.

- **Algoritmo genético:**

  O algoritmo genético implementado segue o modelo introduzido em [24]. O método é um algoritmo genético simples, com uma etapa gulosa probabilística no *crossover* (combinação de duas soluções para a geração de uma terceira). Os genes de indivíduos são valores booleanos, sendo que o $i - ésimo$ gene indica se o vértice $i \in V$ está ou não na solução.

Cada uma destas três heurísticas possui características diferentes quanto ao tempo de execução e à qualidade das soluçõs encontradas, isto é, o tamanho das cliques. A seção 4.1 apresenta uma comparação entre os 3 métodos e o mais adequado para ser utilizado no sistema aqui proposto.

## 3.3 Heurística para a solução ótima da oclusão

Como visto na seções 2.5.4 e 2.5.5, a solução de peso máximo para as oclusões corresponde a um conjunto de caminhos que vão de todos os vértices da primeira até a última camada, respeitando as restrições de fluxo destes vértices.

Como discutido na seção 3.6, encontrar tal conjunto de caminhos é um problema *NP-Difícil*. Sendo assim, o mais indicado é, novamente, o uso de alguma heurística para a solução deste problema. De acordo com a seção 2.5.4 , para tal problema, deve-se encontrar um conjunto de caminhos de peso máximo em um grafo. Além disso, o peso das arestas varia em função do caminho considerado até as mesmas. Este tipo de problema tem uma modelagem muito simples e direta com a meta-heurística *Ant Colony*. O trabalho em [9] mostra que as heurísticas baseadas na *Ant Colony* são o estado da arte para problemas em que se precisa encontrar caminhos em grafos com arestas de peso variável. Esta foi, assim, a abordagem escolhida para o problema.

A idéia original da heurística *Ant Colony* vem da observação do comportamento da exploração de fontes de alimentos por colônias de formigas. Com base em experiências, biólogos constataram que a colônia era capaz de encontrar o caminho mais curto entre o formigueiro e a fonte de alimentos. Isso pode ser explicado pelo uso de feromônios guias. Estes feromônios são depositados pelas formigas enquanto caminham. Quando têm que escolher entre diferentes caminhos, as formigas tendem a seguir aquele com maior concentração de feromônio. Um caminho perde seu feromônio quando não é utilizado, pois este se evapora. Os caminhos mais curtos tendem a ter um maior fluxo de formigas e, consequentemente, um maior acúmulo de feromônio, sendo assim preferidos pela colônia. A figura 3.8 ilustra este processo. As formigas inicialmente tomam cada um dos dois caminhos possíveis com igual probabilidade (figura 3.8.a). Sendo assim, o caminho mais curto tem um fluxo maior e uma maior quantidade de feromônio devido à evaporação. Logo, este constitui o caminho preferido pelas formigas (figura 3.8.b). O modelo considerado pela heurística segue este princípio em que as *formigas* movimentam-se por grafos depositando feromônio nas arestas de seus camimnhos. Maiores detalhes sobre o funcionamento desta meta-heurística podem ser consultados em [9].

De modo geral, para se implementar a heurística *Ant Colony*, devemos considerar três definições: a ordem pela qual as formigas se movimentam no grafo, a forma pela qual escolhem as arestas de seus caminhos e a forma pela qual o feromônio das arestas é depositado e atualizado.

Para este problema, cada formiga conta com as seguintes informações:

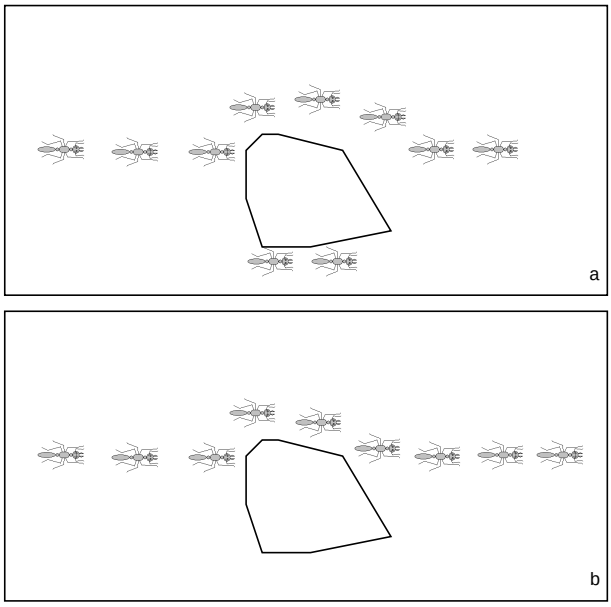



Figura 3.8: Comportamento de uma colônia de formigas na escolha entre dois caminhos.

- seu vértice inicial, usado como chave de identificação.
- um inteiro $n_{passos}$ representando o número de vértices percorridos pela formiga.
- o caminho de $n_{passos}$ contruído pela formiga até um determinado instante.
- o melhor caminho já encontrado pela formiga.

O grafo utilizado na heurística é idêntico ao grafo de oclusões descrito na seção 2.5.2, acrescido da informação sobre a quantidade de feromônio depositado em cada aresta, e do conjunto de caminhos com melhor peso definido pelas formigas.

Seja $F = (f_1, f_2, \ldots f_{n_t})$ o conjunto de formigas, onde $n_t$ é o número de alvos sendo rastreados (ou, alternativamente, o número de rastreadores). A ordem de movimentação das formigas pode ser definida da seguinte forma. Cada formiga do conjunto $F$ monta seu caminho por vez. A formiga cujo melhor caminho já encontrado tem o maior peso (seção 2.5.5) monta seu caminho primeiro. Ela é seguida por aquela que tem o melhor caminho com o segundo maior peso e assim por diante, em ordem decrescente.

Agora definamos a forma pela qual escolhe as arestas a tomar. Uma formiga $f_i \in F$ monta seu caminho aresta por aresta. As arestas são escolhidas aleatoriamente, com probabilidades diferentes para cada uma delas. Supondo que a formiga está no vértice $u$, e sendo $path_i$ seu caminho construído até o momento atual, temos que a probabilidade de que ela percorra a aresta $e_{uv}$ é dada por:

$$prob(e_{uv}|path_i) = \frac{\tau_{uvi}^{\alpha} \cdot \eta\left(e_{uv}\right)^{\beta}}{\sum_{w \in Ngb(u)} \tau_{uwi}^{\alpha} \cdot \eta\left(e_{uw}\right)^{\beta}}, \tag{3.6}$$

sendo $\tau_{uvi}$ o feromônio associado à aresta $e_{uv}$ pela formiga $i$ (o feromônio de uma formiga não afeta a outra) e $\eta\left(e_{uv}\right)$, uma função que atribui um valor heurístico à escolha da aresta $e_{uv}$. Os valores $\alpha$ e $\beta$ são parâmetros da heurística, ambos definidos empiricamente como $2.0$. $Ngb\left(u\right)$ é o conjunto dos vértices vizinhos de $u$. Para o problema em questão, $\eta\left(e_{uv}\right)$ é zero caso não haja um caminho, respeitando as restrições de fluxo do grafo de oclusões (seção 2.5), entre $v$ e algum vértice da última camada e se o vértice $v$ já estiver com seu fluxo máximo. Caso contrário, $\eta\left(e_{uv}\right) = wgt\left(e_{uv}|path_i\right)$ (peso do caminho).

Por fim, devemos descrever a forma de atualização do feromônio em cada aresta. Esta implementação da heurística foi baseada na variação da *Ant Colony* chamada *Ant Colony System* [10], seguindo o método de atualização de feromônio proposto neste modelo. Neste caso, há dois tipos de atualização: *local* e *offline*. A atualização *local* é realizada por cada formiga a cada aresta que ela percorre no grafo. A atualização *offline* é realizada apenas nas arestas pertencentes ao conjunto de melhores caminhos, a cada iteração do algoritmo.

A atualização *local* é definida por:

$$\tau_{uvi} = (1 - \varphi) \cdot \tau_{uvi} + \varphi \cdot \tau_0 \tag{3.7}$$

Em que $\varphi$ é o coeficiente de evaporação local (definido empiricamente como $0.1$) e $\tau_0$, o valor inicial do feromônio das arestas. Ambos são parâmetros fornecidos empiricamente. A atualização *offline* é dada por:

$$\tau_{uvi} = \begin{cases} (1 - \rho) \cdot \Delta\tau_{uvi} + \rho \cdot \tau_0, \text{se } e_{uv} \in path_i; \\ \tau_{uvi}, \text{caso contrário.} \end{cases} \tag{3.8}$$

Em que $\rho$ é o coeficiente de evaporação da atualização *offline* (definido empiricamente como $0.1$) e $\Delta\tau_{uvi}$ um valor heurístico que denota quão bom é o caminho gerado pela formiga $f_i$. Na implementação desta heurística, $\Delta\tau_(uvi)$ é o peso do caminho $path_i$.

Uma abordagem considerada normalmente na implementação de heurísticas em processamento de imagens e visão computacional é o uso de heurísticas gulosas, ou seja, que objetivam maximizar o ganho a cada iteração. Neste caso essa pode não ser uma boa abordagem, dada a natureza inexata dos modelos utilizados neste projeto (sujeitos a incerteza dos modelos estatísticos e a ruídos). A vantagem da criação de um método baseado em uma meta-heurística como a *Ant Colony* é a melhor exploração do espaço de busca. Tal heurística tende a encontrar boas soluções, mesmo diante de escolhas que, à primeira vista, não pareçam boas (devido a ruído, por exemplo) mas que levam a soluções corretas do problema.

## 3.4 Algoritmos dos quantificadores

Os quantificadore são os algoritmos principais empregados na resolução de oclusões pelo Gerenciador de quantificadores (seção 2.6). Estes algoritmos devem, com base em algum método, calcular $pr_t(a, r_i, r_j)$, a probabilidade de um alvo $a$ que se encontra na região $r_i$ no quadro $t-1$ encontrar-se na região $r_j$ no quadro $t$. Em termos do grafo de oclusões (seção 2.5), esta probabilidade é definida como $pr_t(v, r_i, r_j) = wgt(e_{uv}|p)$, o peso da aresta $e_{uv}$ (ligando os vértices $u$ e $v$), dado o caminho $p$ utilizado para atingir-se $u$. Para tanto, os quantificadores podem basear-se em quaisquer dados gerados durante o processo de rastreamento.

A idéia principal da função de um quantificador é avaliar alguma característica de um alvo que permita distinguí-lo dos demais. Por exemplo, o quantificador adequado, para o caso em que o alvo mantém uma trajetória aproximadamente constante pode basear-se em alguma medida do trajeto. Para o caso em que os alvos têm cores diferentes, um quantificador adequado deve considerar o histograma do alvo.

As próximas seções apresentam os algoritmos quantificadores implementados neste projeto.

### 3.4.1 Quantificador por velocidade

O quantificador por velocidade consegue diferenciar alvos que se movem em diferentes velocidades. Ele necessita de um grafo de oclusões com informação sobre as posições dos alvos durante o rastreamento. Como cada vértice do grafo de oclusões representa uma região sendo rastreada, as posições estão associadas a estes vértices.

Dado o caminho $p = \{v_0, v_1, \ldots, v_k\}$, a média e o desvio padrão das velocidades das posições dos vértices de $p$ são calculados sobre este caminho. A probabilidade do vértice $v_{k+1}$ é inversamente proporcional ao número de desvios padrões da diferença entre a velocidade média e a velocidade calculada entre os vértices $v_k$ e $v_{k+1}$. Esta probabilidade é normalizada entre todos os vértices vizinhos de $v_k$.

### 3.4.2 Quantificador por direção

O quantificador por direção diferencia alvos que se deslocam linearmente em direções distintas. Ele necessita da mesma informação do quantificador por velocidade.

Dado o caminho $p = \{v_0, v_1, \ldots, v_k\}$, um vetor de módulo unitário que represente a direção média no trajeto $p$ e a covariância desta medida são calculados. A probabilidade do vértice $v_{k+1}$ é inversamente proporcional à distância de Mahalanobis [30] (a distância ponderada pela covariância) entre a direção média e a direção das posições representadas pelos vértices $v_k$ e $v_{k+1}$. Esta probabilidade é normalizada entre todos os vértices vizinhos de $v_k$.

### 3.4.3 Quantificador por área

O quantificador por área diferencia alvos que possuam regiões de área distintas. Para o cálculo destas áreas, é necessário um grafo de oclusões que possua recortes das imagens das regiões dos alvos. Como cada vértice do grafo de oclusões representa um ou mais alvos sendo rastreados, os recortes estão associados aos respectivos vértices.

Dado o caminho $p = \{v_0, v_1, \ldots, v_k\}$, média e o desvio padrão das áreas das regiões representadas nos vértices de $p$ são calculadas. A probabilidade do vértice $v_{k+1}$ é inversamente proporcional ao número de desvios padrões da diferença entre a área média e a área calculada da região representada pelo vértice $v_{k+1}$. Esta probabilidade é normalizada entre todos os vértices vizinhos de $v_k$.

### 3.4.4 Quantificador por histograma

O quantificador por histograma diferencia alvos com distribuições de cores distintas. Ele necessita da mesma informação do quantificador por área.

Dado o caminho $p = \{v_0, v_1, \ldots, v_k\}$, o histograma médio das regiões representadas nos vértices de $p$ é calculado. A probabilidade do vértice $v_{k+1}$ é inversamente proporcional à distância de Bhattacharya (distância que pode ser interpretada como um produto escalar entre os vetores do histograma) entre o histograma médio e o histograma da região representada pelo vértice $v_{k+1}$. Esta probabilidade é normalizada entre todos os vértices vizinhos de $v_k$.

## 3.5 Combinação de quantificadores

Cada tipo de quantificador, como os apresentados na seção anterior, baseia-se numa informação específica dos alvos sendo rastreados. Muitas vezes, contudo, uma única informação pode não ser suficiente para diferenciar os alvos. Por exemplo, se em um dado momento dois alvos de cores diferentes tornam-se da mesma cor, um quantificador baseado no histograma de cor do alvo pode não conseguir discriminá-los. Assim, dois ou mais quantificadores combinados tendem a comportar-se de maneira mais robusta, fazendo com que um quantificador corrija as falhas do outro.

Uma combinação de quantificadores também é um quantificador. Seja $Q = \{q_1, \ldots, q_m\}$ um conjunto de quantificadores. Define-se a probabilidade combinada como:

$$wgt(e_{uv}|p) = \sum_{i=1}^{m} \alpha_i \cdot wgt_{q_i}(e_{uv}|p) \tag{3.9}$$

Neste caso, a probabilidade computada é uma combinação linear dos resultados dos quantificadores de $Q$. O conjunto $A = \{\alpha_1, \ldots, \alpha_m\}$ é o conjunto dos pesos, referentes à *taxa de*

*consenso* dos quantificadores. Esta *taxa de consenso* é baseada na idéia de que se um quantificador atribui probabilidades semelhantes para todas as arestas de um vértice, ele não adiciona tanta informação útil à solução da oclusão, ao contrário do caso em que o mesmo atribuísse probalidades diferentes. Estes pesos são calculados de acordo com a equação abaixo:

$$avg_i = \frac{1}{|Ngb\,(u_n)\,|} \sum_{w \in Ngb(u_n)} wgt_{q_i}(e_{uw}|p)$$

$$\alpha_i = \sqrt{\frac{1}{|Ngb\,(u_n)\,|} \sum_{w \in Ngb(u_n)} \left[wgt_{q_i}(e_{uw}|p) - avg_i\right]^2} \tag{3.10}$$

Ou seja, a *taxa de consenso* de um quantificador $q_i$ é dada pelo desvio padrão da probabilidade das arestas candidatas. O conjunto $A$ é normalizado tal que a soma dos pesos é igual a $1.0$.

## 3.6 Prova de complexidade

Nesta seção, provaremos que o problema da oclusão (*PO*), definido na seção 2.5, é um problema *NP-difícil*. Para demonstrá-lo, usaremos o conhecido problema do caixeiro viajante (*CV*), que o circuito Hamiltoniano (circuito que passa exatamente uma vez por cada vértice de um grafo) de maior ou menor peso deve ser encontrado [16]. Realizaremos uma redução do *PO* para o *CV*, em outras palavras, transformaremos os parâmetros de entrada do *CV* em parâmetros de entrada do *PO*. Como o *CV* é um problema *NP-difícil*, mostraremos que o *PO* também o é. Assim, seja $G_{cv}$ o grafo completo dado como entrada do *CV*. Seja $V_{cv} = \{v_1, \ldots, v_n\}$ o conjunto de vértices de $G_{cv}$ e $wgt_{cv}\,(v_i, v_j)$ uma função que mapeia uma aresta de $G_{cv}$ em um valor real. Transformamos $G_{cv}$ no grafo $G_{po}$, parâmetro de entrada do *PO*, da seguinte maneira. Seja $U_{po}$ o conjunto de vértices de $G_{po}$. $U_{po}$ é dividido em camadas. A primeira camada de $U_{co}$ possui apenas um vértice. $U_{po} = \{\{u_0\}, U_1, \ldots, U_l\}$. Como $G_{po}$ é um grafo de oclusões, há arestas apenas entre a aresta $i$ e $i+1$. Também, para a redução, cada subgrafo $U_i \cup U_{i+1}$ é bipartido completo. Cada aresta de $G_{po}$ possui capacidade 1. Seja $path = \{v_0, p_1, \ldots, p_k\}$ um caminho em $G_{po}$ e $wgt_{po}\,(p)$ uma função que mapeia o caminho $path$ em um número real. Seja $g\,(u_i)$ uma função que mapeia cada vértice de $U_{po}$ em um vértice de $V_{cv}$, definido por $g(u_i) = [(i-1)mod|V_{cv}|] + 1$.

Esta transformação é ilustrada na figura 3.9. A figura 3.9(a) mostra o grafo $G_{cv}$. A figura 3.9(b) mostra o grafo $G_{po}$ criado a partir de $G_{cv}$. Note que $G_{po}$ tem o número de camadas igual ao número de vértices de $G_{cv}$, e cada camada de $G_{po}$ possui vértices associados a cada vértice de $G_{cv}$.

Uma solução para o grafo $G_{po}$ contém apenas um caminho, cujo peso é definido por:
$\infty$ se $g(path_i) = g(path_j), i \neq j$ e

$w_{po}(p) = \sum_{i=1}^{k-1} w_{tsp}[g(path_i), g(path_{i+1})]$, caso contrário.

O peso inifinito evita que dois vértices de $U_{pp}$ sejam mapeandos para o mesmo vértice do conjunto $V_{cv}$. É trivial averiguar que um caminho com mínimo custo da primeira até a última camada não possui vértices mapeados para o mesmo vértices de $V_{cv}$. Deve notar-se que esta definição de peso é permitida pela definição do problema de oclusão. Dado que o tamanho do caminho é igual ao número de vértices de $G_{cv}$, este caminho representa um circuito Hamiltoniano do *CV*.

Pela equação acima, sabemos que o peso de um caminho é igual ao peso do circuito que ele representa. Assim, encontrar o caminho de máximo peso no grafo $G_{po}$ é equivalente a encontrar o circuito de custo máximo em $G_{cv}$. Logo, o problema da oclusão, como definido neste documento, é *NP-difícil* □.

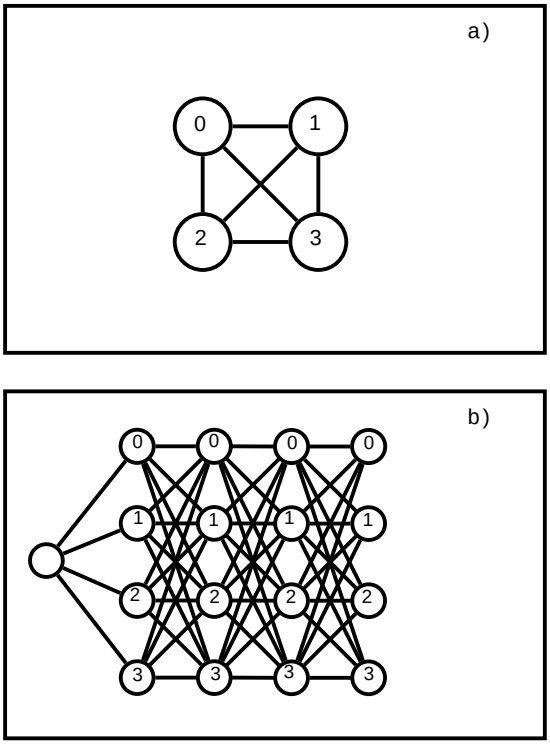



Figura 3.9: a) Grafo de entrada para o problema do caixeiro viajante. b) Grafo gerado como entrada para o problema da oclusão.

# Capítulo 4

# Resultados

Esta seção apresenta os resultados experimentais de nosso trabalho. Avaliam-se três aspectos do sistema: qual é a melhor heurística de clique para ser usada na decomposição em cliques de um grafo, se os rastreadores implementados respeitam as suposições do modelo (seção 2.3.5) e a eficiência do método de resolução do problema de oclusão. A seção 4.1 apresenta os resultados para as heurísticas de clique máxima. A seção 4.2 apresenta os resultados sobre os rastreadores. Finalmente, a seção 4.3 apresenta os resultados da heurística para resolução de oclusões.

## 4.1 Heurística para cobertura por cliques

A seção 3.2 apresentou uma heurística de cobertura por cliques para a otimização do grafo de oclusões, cujo procedimento mais importante é outra heurística que deve encontrar cliques máximas em um grafo. Como mostrado na seção 3.2, foram implementadas $3$ heurísticas para busca de cliques maximais: heurística gulosa crescente, gulosa decrescente e genética.

Realizamos testes sobre dois aspectos importantes: o tamanho das cliques encontradas e o tempo de execução das heurísticas. Esta escolha foi feita levando-se em conta que quanto maior as cliques encontradas, mais simples e menor é o grafo de oclusões. Além disso, a heurística para cobertura de cliques deve ter um baixo tempo de execução, pois deve ser executada muitas vezes. Os testes foram realizados em um *Pentium 1700 Mhz* com $256$ *Mbytes* de memória *RAM*. Um conjunto de grafos com $450$ vértices e densidade média de arestas de $50\%$ foi utilizado. Os resultados são mostrados nas tabelas 4.1 e 4.2. A tabela 4.1 mostra o tamanho das cliques encontradas nos $5$ grafos pelas heurísticas gulosa crescente, gulosa decrescente e genética, respectivamente. A tabela 4.2 mostra o tempo médio de execução destas três heurísticas.

As heurísticas gulosas crescente e decrescente tiveram resultados claramente inferiores aos da heurística genética, levando em conta a o tamanho das cliques encontradas. A heurística gulosa decrescente, contudo, obteve resultados muito superiores ao da heurística gulosa crescente. Levando-se em conta o tempo de execução, a heurística genética tem um desempenho de duas

| Grafo | Crescente | Decrescente | Genética |
|---|---|---|---|
| 1 | 9 | 21 | 27 |
| 2 | 9 | 19 | 27 |
| 3 | 9 | 21 | 26 |
| 4 | 8 | 21 | 26 |
| 5 | 9 | 20 | 27 |

Tabela 4.1: Tamanho das cliques encontradas.

| Heurística | Tempo médio |
|---|---|
| Crescente | 0.0374 |
| Decrescente | 0.0456 |
| Genética | 2.7490 |

Tabela 4.2: Tempo de execução das heurísticas.

ordens de grandeza inferior ao das heurísticas gulosas. Dado isso, a heurística gulosa decrescente foi escolhida para ser utilizada no projeto em questão, já que gera cliques de tamanhos bastante razoáveis e consome pouco tempo de processamento.

A figura 4.1 mostra um pequeno grafo de oclusões sem qualquer otimização, seguido pelo grafo equivalente otimizado. Nota-se que na região onde ocorrem oclusões, o grafo não otimizado tem uma explosão de vértices e arestas, problema que não acontece com o grafo otimizado.

## 4.2 Resultados dos rastreadores

Nesta seção discutem-se os resultados do módulo de rastreamento em alguns vídeos de experimentos biológicos reais. O objetivo destes testes é verificar se os rastreadores implementados respeitam as suposições do modelo de rastreamento (seção 2.3.5) que, resumidamente, são: todo alvo deve ser rastreado por ao menos um rastreador, um rastreador não deve perder-se e os rastreadores podem trocar de alvo apenas em um evento de oclusão.

Além disso, como base de comparação do desempenho dos métodos implementados, realizaram-se os rastreamentos com os algoritmos aqui propostos e com uma implementação do conhecido rastreador por fluxo ótico Lucas Kanade, *KLT* [5], implementado pela biblioteca *OpenCV*. Basicamente, o *KLT* é um algoritmo que efetua o rastreamento de pontos calculando o deslocamento entre dois quadros de cada ponto sendo seguido. Verificou-se também se o *KLT* atende as restrições de rastreamento deste sistema.

Para os testes, consideram-se 5 vídeos de diferentes experimentos (listados na seção 1.2), com as seguintes características.

1. Campo aberto (rato) : um experimento de campo aberto com um único rato. O vídeo tem

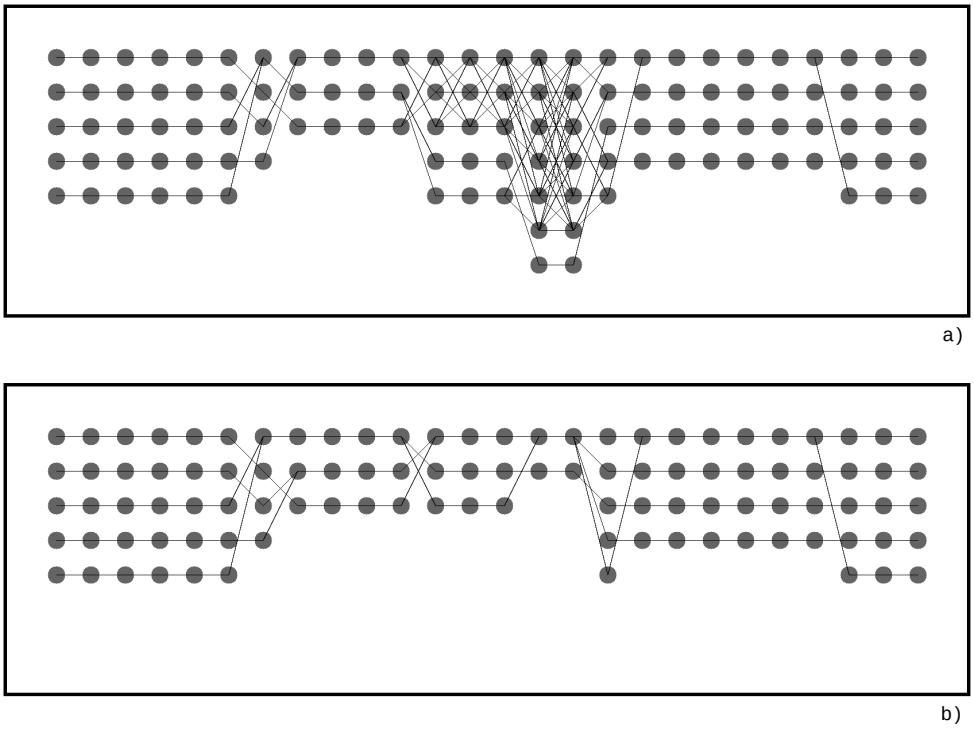



Figura 4.1: Grafo normal (a) e grafo otimizado (b).

717 quadros de boa qualidade, mas com a presença de ruído branco na gravação.

2. Labirinto aquático de Morris: um experimento com um camundongo nadando no labirinto aquático de Morris. Tem 1947 quadros de qualidade regular.

3. Campo aberto (3 formigas): um experimento de campo aberto com 3 formigas. O vídeo tem 1598 quadros de boa qualidade.

4. Campo aberto (20 formigas): um experimento de campo aberto com 20. O vídeo tem 299 quadros, com ruído gerado por compactação de vídeo com perdas.

5. Rastreamento de células: um experimento com 32 sendo rastreadas. O vídeo tem apenas 40 quadros e possui muito ruído e uma taxa baixíssima de quadros por segundo.

As figuras 4.2, 4.4, 4.6, 4.8 e 4.10 mostram os resultados do módulo de rastreamento do sistema proposto. Apresentamos a seguir, comentários gerais sobre os resultados obtidos.

- No primeiro vídeo (figura 4.2) foi utilizado um rastreador por remoção de fundo (seção 2.3.4) por poder lidar com o ruído branco presente nestes vídeo. O roedor foi rastreado com precisão em todos os quadros.

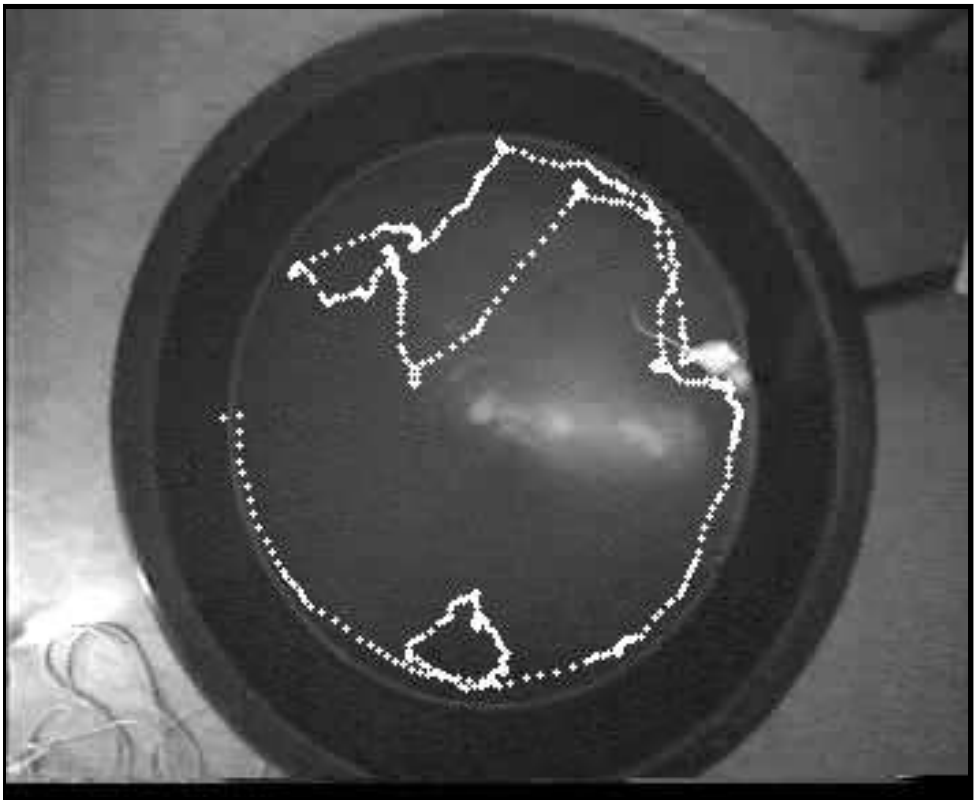


Figura 4.2: Resultados do rastreamento para o experimento de campo aberto utilizando o sistema proposto.

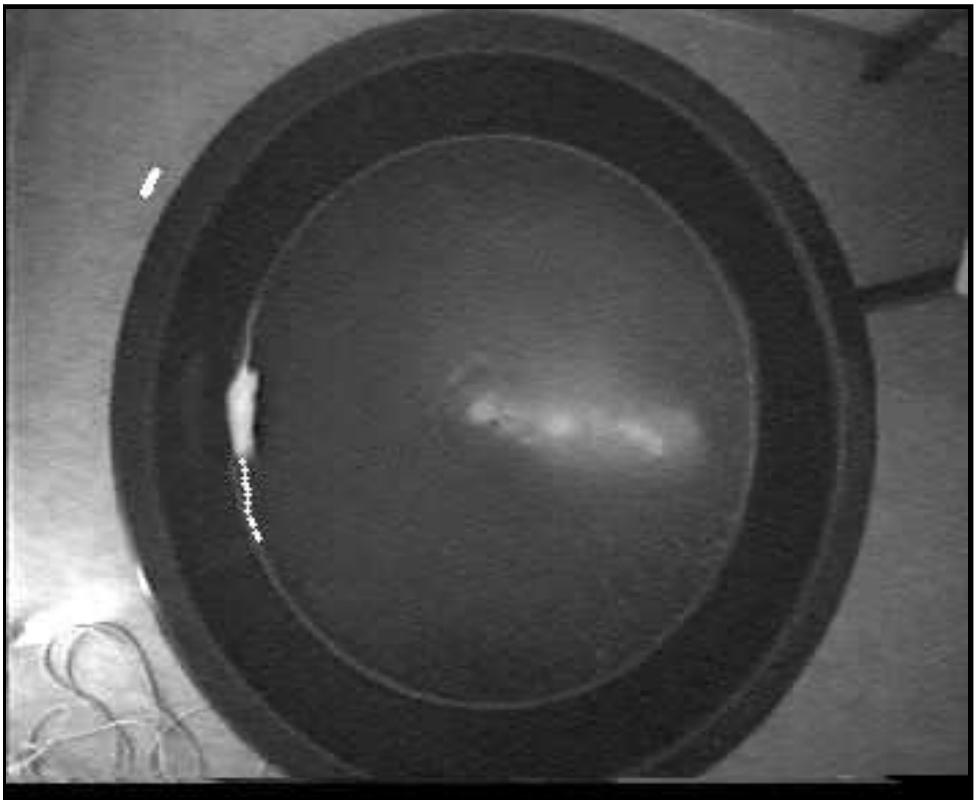


Figura 4.3: Resultados do rastreamento para o experimento de campo aberto utilizando o *KLT*.

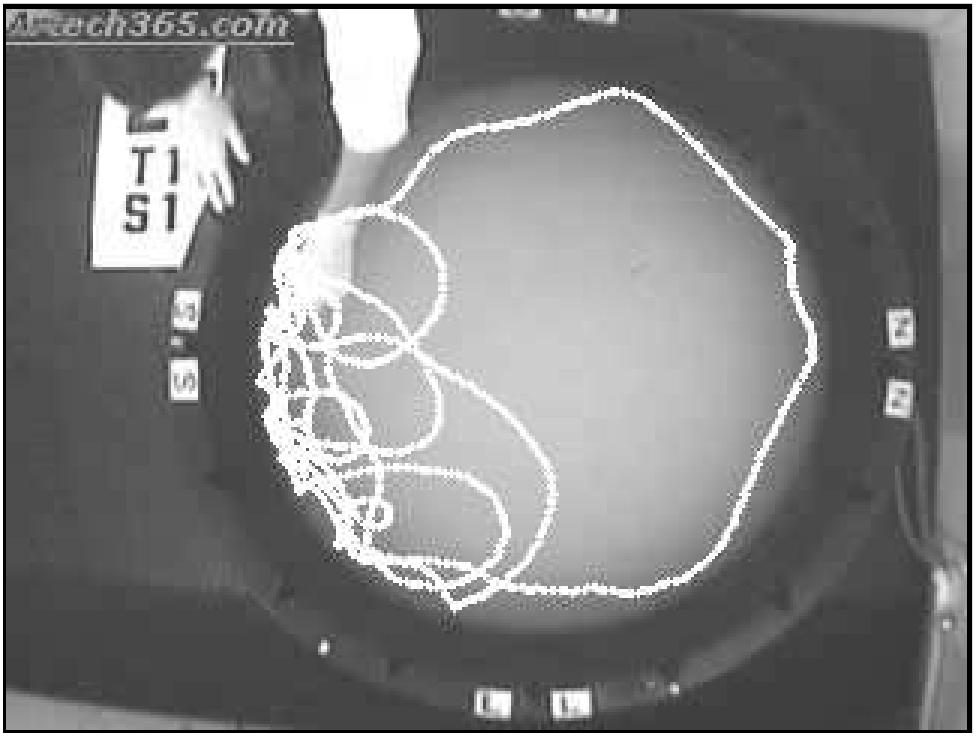



Figura 4.4: Resultados do rastreamento para o experimento do labirinto aquático de Morris utilizando o sistema proposto.

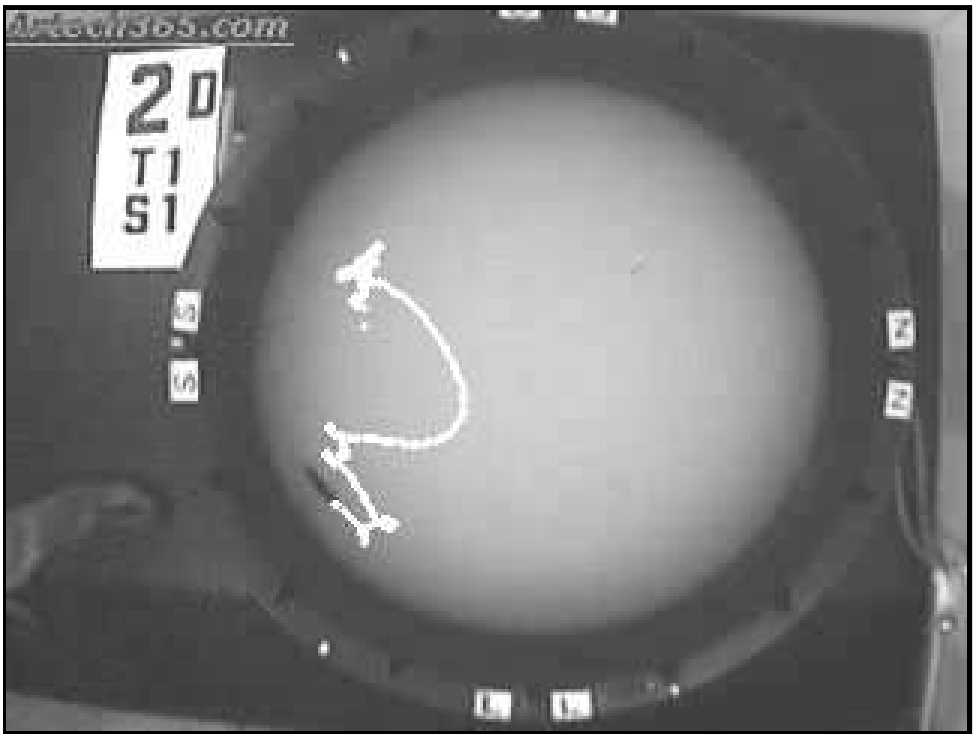



Figura 4.5: Resultados do rastreamento para o experimento do labirinto aquático de Morris utilizando o *KLT*.

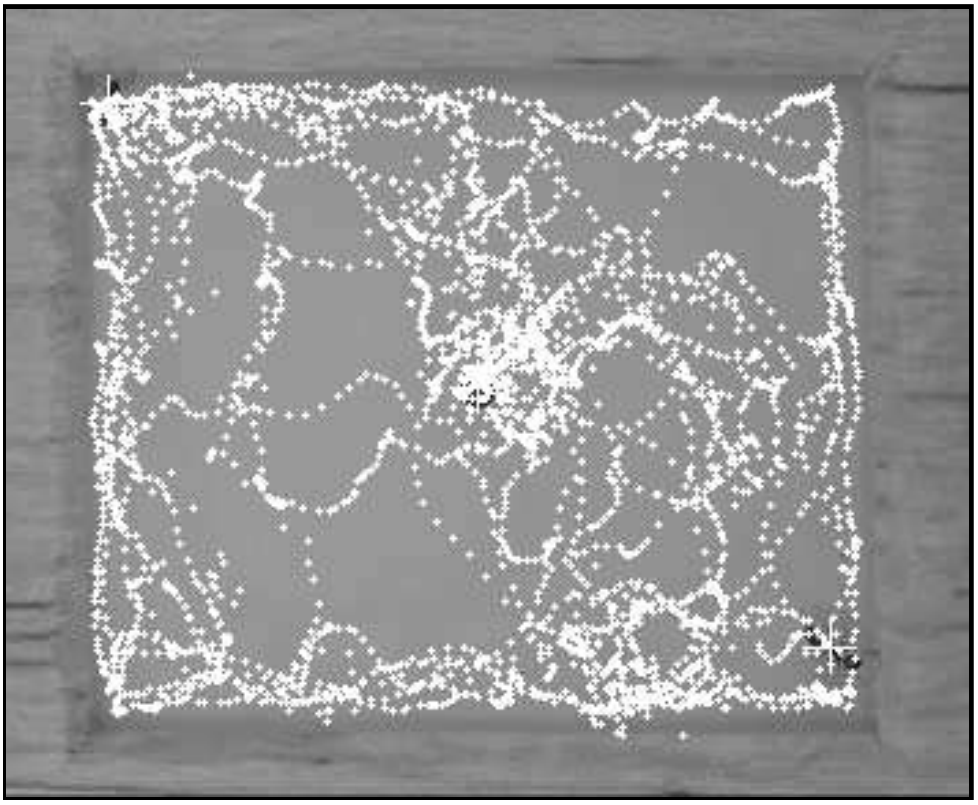


Figura 4.6: Resultados do rastreamento para o experimento de campo aberto com 3 formigas utilizando o sistema proposto.

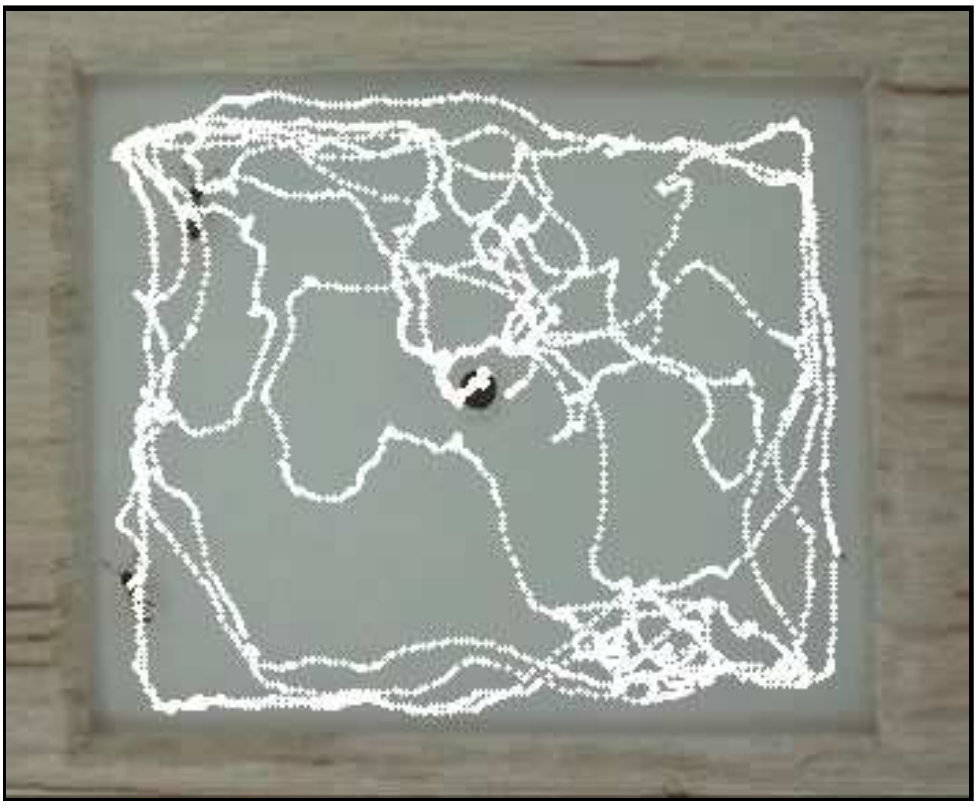


Figura 4.7: Resultados do rastreamento para o experimento de campo aberto com 3 formigas utilizando o *KLT*.

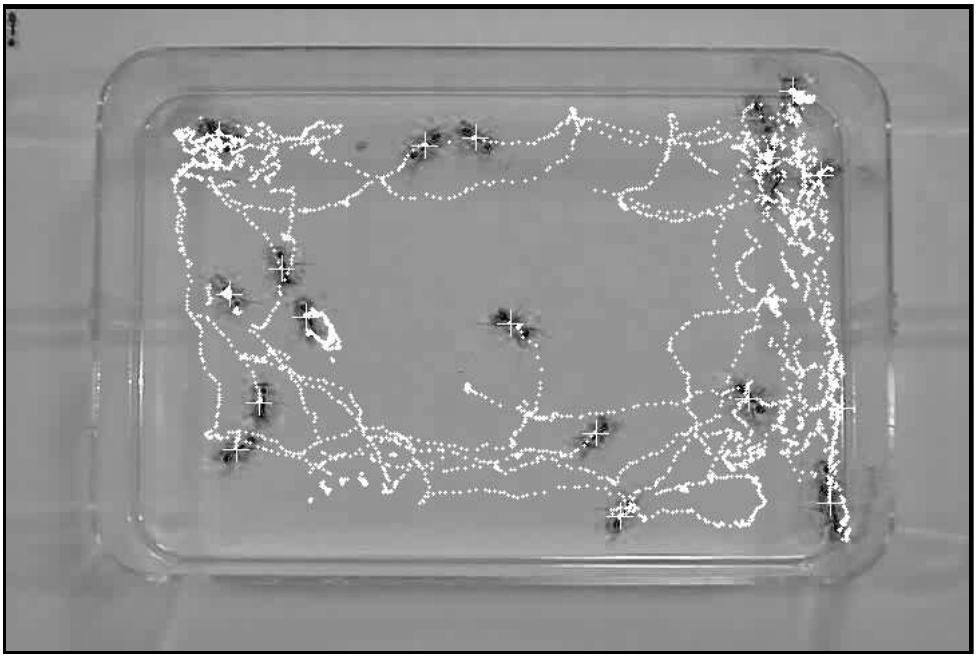


Figura 4.8: Resultados do rastreamento para o experimento de campo aberto com 20 formigas utilizando o sistema proposto.

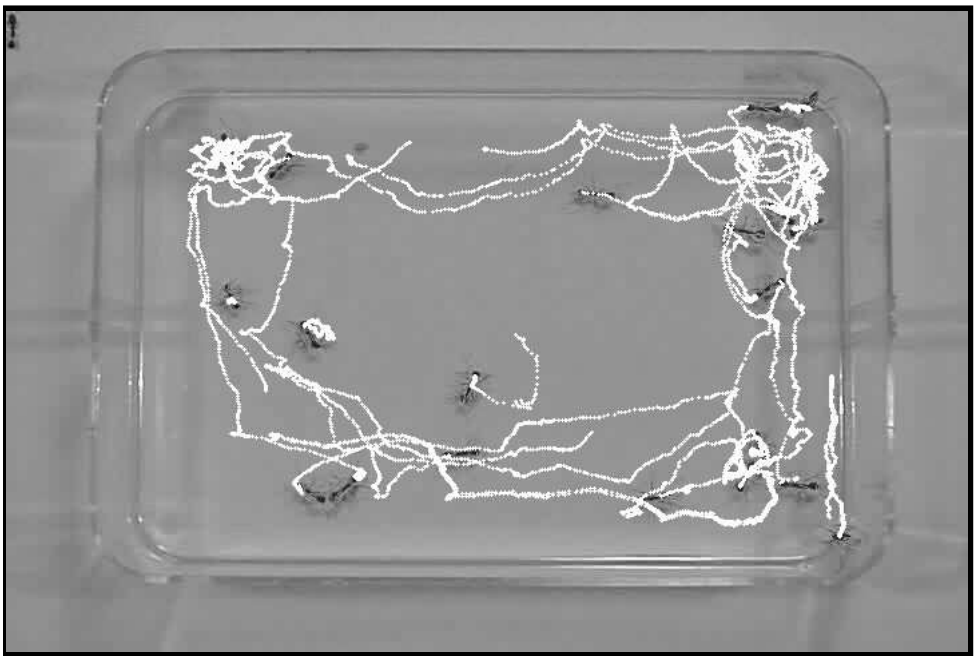


Figura 4.9: Resultados do rastreamento para o experimento de campo aberto com 20 formigas utilizando o *KLT*.

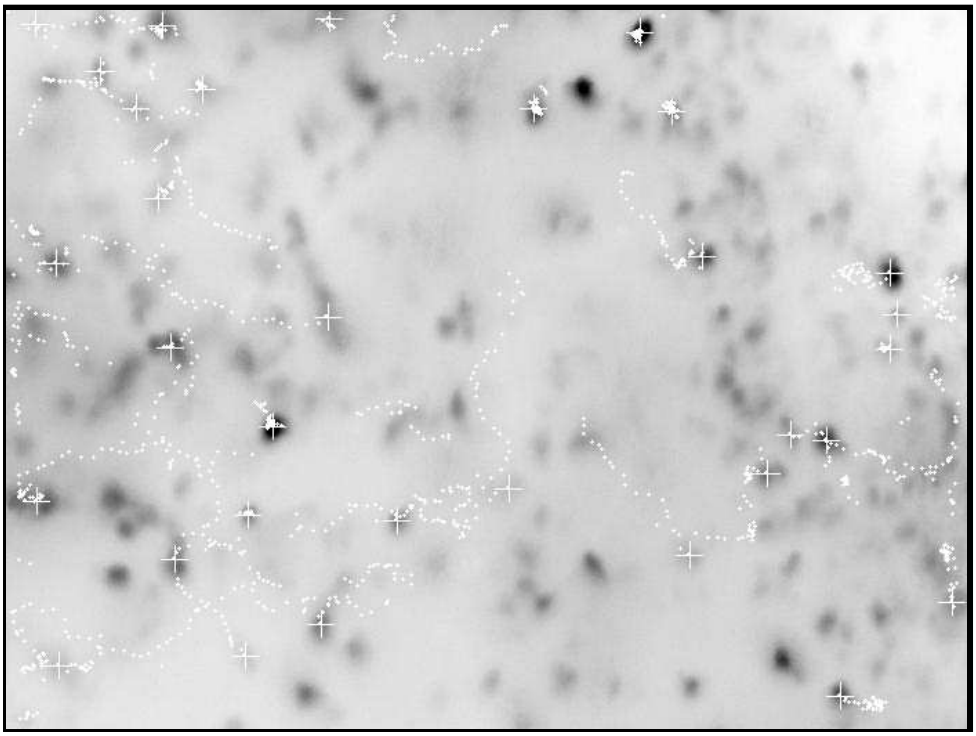


Figura 4.10: Resultados do rastreamento para o experimento com células utilizando o sistema proposto.

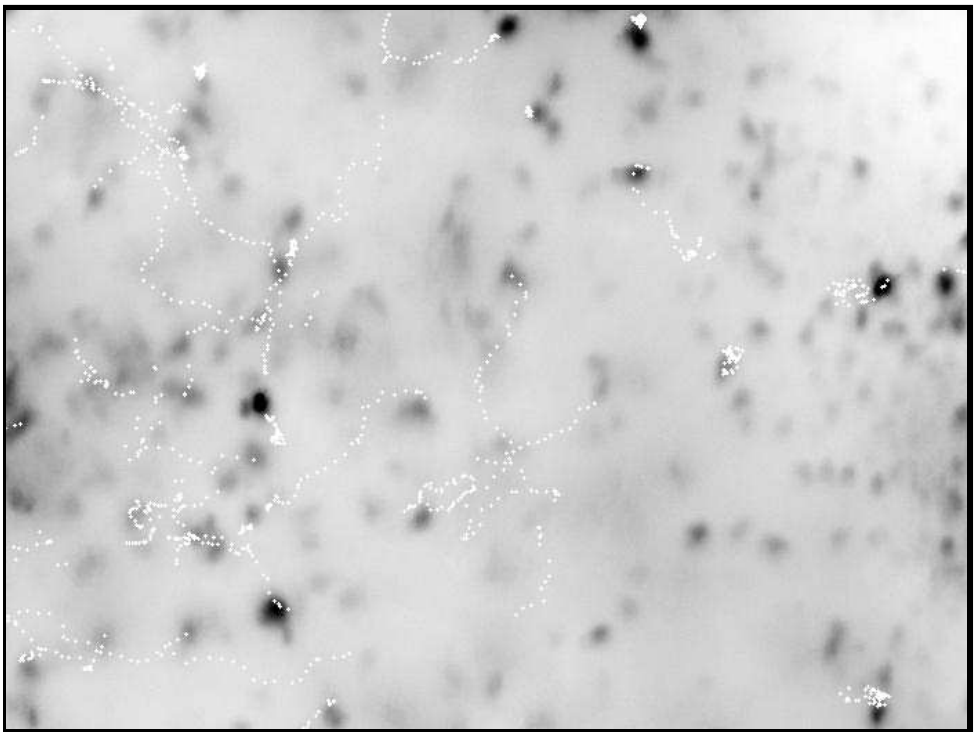


Figura 4.11: Resultados do rastreamento para o experimento com células utilizando o *KLT*.

- No segundo vídeo (figura 4.4) foi utilizado o rastreador morfológico (seção 2.3.3), para contornar o problema da baixa qualidade deste tipo de vídeo. O rato foi seguido mesmos em momentos em que ele era invisível a olho nu, devido à baixa iluminação do experimento.

- No terceiro vídeo (figura 4.6) foi utilizado o rastreador de múltiplos alvos por remoção de fundo (seção 2.3.6), já que este rastreador é rápido e se beneficia da boa qualidade do vídeo, que não apresenta problemas de iluminação ou reflexo. Como uma das formigas sai de um pequeno buraco no meio da arena depois do começo do vídeo, e para que a restrição de rastreamento fosse cumprida, considerou-se o buraco como sendo um possível alvo e as formigas eventualmente em oclusão com ele. A partir desta consideração, as formigas foram rastreadas corretamente.

- No quarto vídeo (figura 4.8) foi utilizado um rastreador por casamento de regiões (seção 2.3.7) com remoção de fundo (seção 3.1.5), escolhido pela presença de ruído na imagem segmentada. As formigas foram rastreadas adequadamente, exceto em alguns quadros, em que uma das 20 formigas, devido a ruído, não foi devidamente rastreada.

- No quinto vídeo (figura 4.10), o melhor resultado foi obtido utilizando-se o rastreador por casamento de regiões (seção 2.3.7) com segmentação por limiarização de bacias de

watershed (seção 3.1.6), um algoritmo mais complexo escolhido pela baixíssima qualidade do vídeo. Este vídeo obteve os piores resultados, com várias células sendo perdidas, devido ao ruído e à baixa taxa de amostragem dos quadros. Contudo, pode-se obter uma visão geral do padrão de movimentação das células que, neste caso, apresentam movimento browniano.

As figuras 4.3, 4.5, 4.7, 4.9 e 4.11 mostram os resultados do rastreamento usando o método *KLT* [5]. O *KLT* tentou seguir um ponto pertencente a cada alvo. Abaixo apresentam-se os resultados obtidos.

- No primeiro vídeo (figura 4.3), o *KLT* rapidamente perde o roedor devido ao ruído branco presente na gravação.
- No segundo vídeo (figura 4.5), o *KLT* também perde o roedor devido a presença de ruído e má iluminação.
- No terceiro vídeo (figura 4.7) , um vídeo de melhor qualidade, o *KLT* consegue rastrear os alvos com precisão enquanto estes não se encontram em oclusão. Contudo, depois de algumas oclusões, todas as formigas deixam de ser rastreadas.
- No quarto vídeo (figura 4.9), o mesmo problema do vídeo anterior acontece. As formigas são bem rastreadas, até o momento que acontecem as oclusões, quando $4$ delas são perdidas.
- No quinto vídeo (figura 4.11), o *KLT* se saiu admiravelmente bem enquanto as células estavam posicionadas fora de oclusão. Ao todo $5$ células foram perdidas, um número menor que o do método aqui proposto. Isso se deve ao fato de que, para o modelo de rastreamento implementado pelo *KLT*, as células não rastreadas não representam ruído.

Os resultados discutidos acima são sintetizados na tabela 4.3 que apresenta, para cada vídeo, o número de alvos que nele devem ser rastreados, o número de alvos perdidos pelos métodos de rastreamento do sistema proposto e o número de alvos perdidos pelo *KLT*. Os dados foram obtidos através da inspeção visual do desempenho dos rastreadores.

Tendo em vista estes resultados, conclui-se que a implementação de rastreadores que seguem as restições presentes na seção 2.3.5 é factível. O *KLT*, um método muito conhecido de rastreamento de múltiplos alvos, não se mostrou adequado para lidar com esta classe de vídeos, devido a ruídos e presença de oclusão, entre outros problemas.

| Vídeo | alvos | sistema | *KLT* |
|---|---|---|---|
| 1 | 1 | 0 | 1 |
| 2 | 1 | 0 | 1 |
| 3 | 3 | 0 | 3 |
| 4 | 20 | 1 | 4 |
| 5 | 32 | 7 | 5 |

Tabela 4.3: Resultados dos rastreamentos.

## 4.3 Solução de oclusões

Para a solução do problema de oclusão, são necessárias as coordenadas reais de cada alvo atráves de todos os quadros do vídeo. Contudo, tal informação não estava previamente disponível para nenhum dos vídeos utilizados neste projeto. Além disso, registrar as posições manualmente demandaria um esforço muito grande. Por exemplo, para os vídeos com multiplos alvos utilizados no projeto, aproximadamente $10.000$ posições deveriam ser registradas manualmente. Além disso, tal quantidade de trabalho manual poderia gerar muitos erros nos testes. Para evitar isso, desenvolveu-se um sintetizador de videos de teste.

Este sintetizador gera vídeos com várias regiões de diferentes formas, tamanhos, cores e padrão de movimento. Cada alvo pode ser diferenciado pelo seguinte conjunto de características:

- forma: as regiões podem ser retangulares, circulares ou elípticas.
- tamanho: a área de cada alvo pode variar dentro de um intervalo, fornecido como parâmetro.
- cor: cada alvo é pintado com um tom de cinza.
- padrão de movimento: cada alvo pode apresentar movimento aleatório, com igual probabilidade para todas as direções, ou movimento linear em uma dada direção, ricocheteando nas bordas da imagem.
- velocidade: cada alvo se movimenta com velocidade constante.

Estas características são escolhidas aleatoriamente para cada um dos alvos. Todas elas (exceto forma e cor, para facilitar a segmentação) são corrompidos por ruído gaussiano cujas características são parâmetros do sintetizador. Assim, a cada quadro, o tamanho dos alvos, sua direção de deslocamento e sua velocidade podem sofrer variações. A figura 4.12 mostra alguns quadros dos vídeos sintetizados utilizados nos testes. Pode-se ver nas imagens 4.12.d e 4.12.f momentos de oclusões. Os dados de rastreamento destes vídeos foram gerados pelo rastreador por casamento de regiões (seção 2.3.7), utilizando uma limiarização simples devido às características dos vídeos sintéticos.

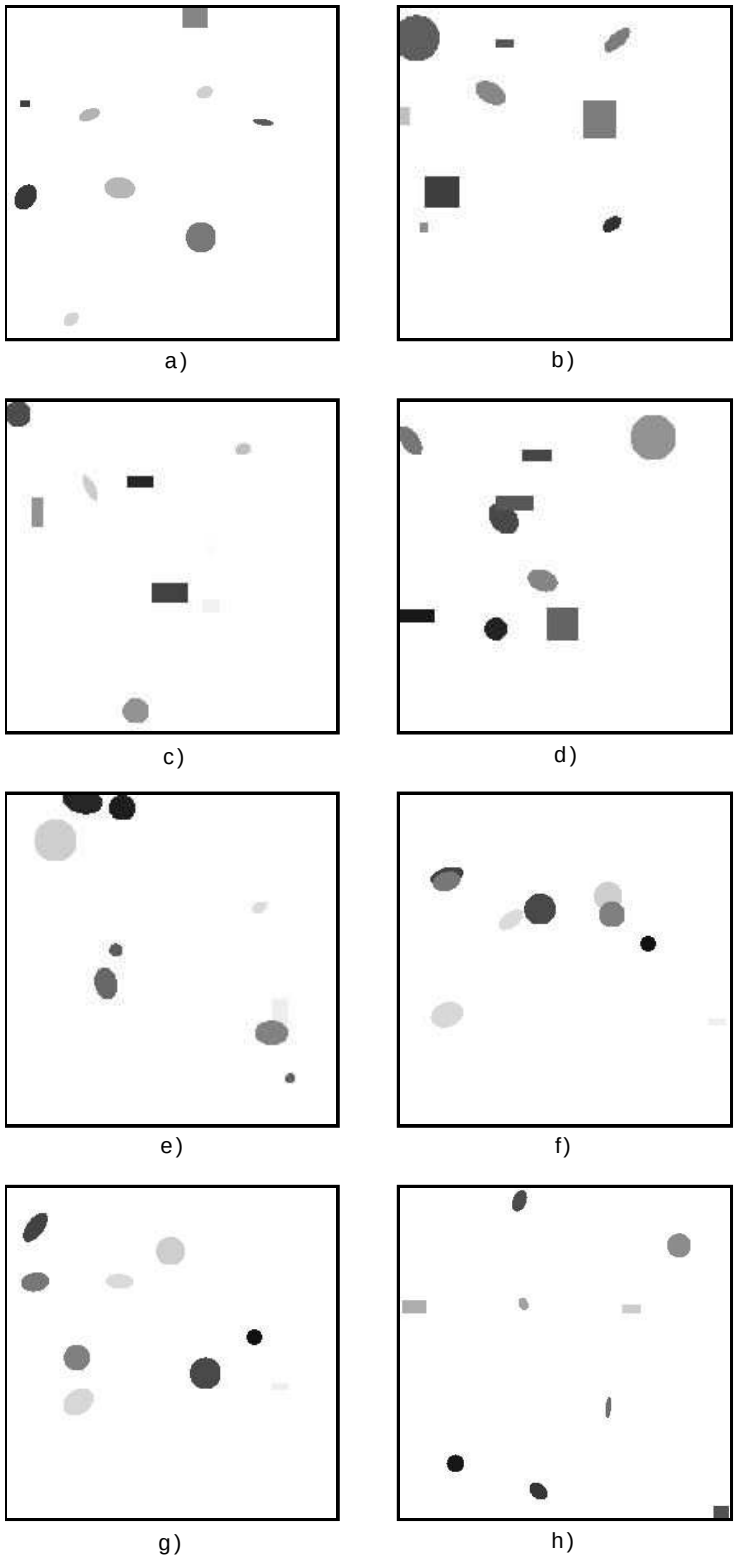



Figura 4.12: Exemplos de imagens dos vídeos sintetizados.

| Video | # de quadros | # de alvos | # oclusões | Erro AC | Erro HG |
|---|---|---|---|---|---|
| 1 | 50 | 8 | 10 | 0% | 30% |
| 2 | 50 | 9 | 5 | 0% | 0% |
| 3 | 70 | 8 | 6 | 0% | 16% |
| 4 | 100 | 9 | 7 | 14% | 28% |
| 5 | 80 | 9 | 10 | 10% | 20% |
| 6 | 100 | 9 | 7 | 14% | 42% |
| 7 | 100 | 9 | 5 | 20% | 40% |
| 8 | 150 | 9 | 8 | 12,5% | 50% |

Tabela 4.4: Resultados da resolução de oclusões, com a porcentagem de erro de cada método.

Para verificar a eficiência da heurística *Ant Colony*(*AC*) na solução do problema de oclusões, foram gerados $8$ vídeos. Comparamos a heurística *ant colony* com uma heurística gulosa (*HG*) cujo funcionamento é identico ao da *Ant Colony*, com a exceção que as formigas sempre escolhem o caminho com o maior peso e desconsideram a informação de feromônio. Com esta comparação, pretende-se mostrar que a meta-heurística implementada é mais apropriada para a aplicação do que uma simples heurística gulosa.

Os resultados são mostrados na tabela 4.4. Nela, para cada vídeo são apresentados, respectivamente: o número de quadros do vídeo sintético, o número de alvos presentes, o número de oclusões envolvendo estes alvos, a porcentagem dos erros na solução das oclusões pela heurística *Ant Colony* e heurística gulosa.

Como se pode ver, a heurística *ant colony* consegue encontrar as soluções com uma taxa de erros inferior ao da heurística gulosa descrita nesta seção. A razão do maior número de erros é que a heurística gulosa se prende a ótimos locais, enquanto a *Ant Colony* tende a realizar uma exploração ampla do espaço das soluções.

Os erros apresentados acontecem por uma falha no método de combinação de quantificadores (seção 3.5). Este método define uma função objetivo que atribui pesos a cada possível solução do rastreamento (seção 2.5.5). Contudo, nos casos em que os erros ocorreram, a soluções corretas do rastreamento, isto é, as que resolvem corretamente todas as oclusões, não recebem os valores máximos da função. Isto faz com que uma solução com erros seja preferida pela heurística *Ant Colony*. Para uma solução geral deste problema, seria necessária a criação de um método inteligente de se determinar qual é a melhor forma de combinar os quantificadores para a obtenção do resultado correto. Tais métodos serão abordados em futuras pesquisas.

O próximo capítulo apresenta as conclusões gerais sobre o trabalho proposto.

# Capítulo 5

# Conclusões

Este trabalho abordou o problema do rastreamento automático de múltiplos alvos aplicado a vídeos de experimentos de laboratório. Foi apresentado um conjunto de técnicas baseadas em processamento de imagens, visão computacional, grafos, inteligência artificial e otimização combinatória.

Tais técnicas foram implementadas visando à flexibilidade de serem combinadas de diferentes formas para a criação de sistemas de coletas de parâmetros em experimentos de biologia. Estes sistemas são de grande utilidade nesta área dado que os experimentos, em geral, possuem uma quantidade significativa de parâmetros difíceis de serem extraídos sem a ajuda de métodos computacionais.

Os métodos aqui discutidos lidam com o rastreamento de alvos isolados, detecção e tratamento de oclusões, estas consideradas quando dois ou mais alvos se aproximam ou sobrepõem-se, fazendo com que seja impossível para os algoritmos rastreadores diferenciá-los corretamente. Assim, há uma potencial "troca de identidade"entre os alvos, o que gera inconsistência nos dados coletados. Para lidar com isso, foram criados um modelo para representar as oclusões e um método para resolvê-las, ou seja, corrigir a inconsistência dos dados. Foi apresentada uma modelagem das oclusões utilizando-se um grafo. Foram apresentados métodos para a otimização deste grafo, já que seu tamanho e complexidade são importantes para o bom desempenho do sistema. Demontra-se que a resolução das oclusões, da forma com se define neste trabalho, é um problema *NP-difícil*, indicando que não é conhecido nenhum algoritmo com tempo de execução sub-exponencial que resolva este problema. Sendo assim, foi realizada uma implementação baseada na meta-heurística *Ant colony*, método polinomial que tende a encontrar boas soluções para as oclusões. Para um dado rastreamento, pode haver várias soluções diferentes para as oclusões e de diferentes qualidades. Sendo assim, foram implementados métodos para a determinação da qualidade destas soluções, os *quantificadores*.

Para validação dos modelos, os métodos foram aplicados a vídeos de diferentes experimentos. Além disso, utilizaram-se ainda vídeos sintéticos para a validação da resolução das oclu-

sões. Foram testados igualmente os métodos para a otimização do grafo de oclusões. Para que as oclusões possam ser registradas e tratadas, todos os alvos devem ser seguidos corretamente e o momento em que entram em oclusão deve ser detectado. Nos vídeos de experimentos, os métodos implementados realizaram esta tarefa corretamente, exceto em um dos vídeos, devido a ruídos e baixa taxa de amostragem. O método para resolução de oclusões obteve um desempenho satisfatório, mas não conseguindo encontrar a melhor solução na maioria dos testes. Isso se deve ao fato de que a função objetivo definida para medir a qualidade de uma solução não atribui corretamente o valor máximo a melhor destas soluções. Não é fácil encontrar um método geral que consiga definir esta função de maneira adequada.

Alguns exemplos de extensões a este trabalho são os seguintes:

- Implementação de uma combinação de quantificadores baseado em aprendizado de máquina:

  O ponto mais difícil de adaptação do sistema é fazer com que a função de peso de um conjunto de caminhos do grafo de oclusões tenha como máximo global a correta solução de oclusão. Para isso, os quantificadores devem ser corretamente combinados. Métodos de aprendizado de máquina podem ser bastante adequados a este fim.

- Implementação de novos quantificadores:

  Cenários diferentes de rastreamento podem requerer tipos diferentes de quantificadores. Eles devem ser implementados para que o sistema seja mais abrangente.

- Adaptação do rastreamento para o uso de várias câmeras:

  Existem muitas aplicações de rastreamento em que se buscam as posições tridimensionais de um alvo. Para este tipo de aplicação, o sistema deve ser adaptado para lidar com todas as complicações que o uso de múltiplas câmeras pode trazer.

- Adaptação do rastreamento para alvos que entram e saem do campo de visão.

  Em vários experimentos de biologia pode haver a ocorrência de alvos que entram e saem do campo de visão. Um rato pode se esconder em sua toca e uma célula pode se dividir em duas por mitose, por exemplo. O sistema deve ser adaptado para poder lidar com estes casos.

- Adaptação para um rastreamento em tempo real:

  Em muitas aplicações, pode ser de interesse realizar-se o rastreamento em tempo real. Para tanto, o sistema aqui descrito necessita sofrer algumas adaptações para atender a esta restrição. Os algoritmos rastreadores são métodos rápidos que podem ser executados em tempo real. A maioria das adaptações devem ser feitas no módulo resolvedor de oclusões, já que este apresenta um conjunto de algoritmos que demanda maior tempo computacinal.

Por outro lado, como apresentado em [9], uma heurística *Ant colony* pode ser adaptada para trabalhar com grafos que se modifiquem dinamicamente, possibilitando, assim, uma atualização rápida do grafo de oclusões.

# Referências Bibliográficas


[1] Tucker Balch, Zia Khan, and Manuela Veloso. Automatically tracking and analyzing the behavior of live insect colonies. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 521–528, New York, NY, USA, 2001. ACM Press.

[2] Gerald J. F. Banon and Júnior Barrera. *Bases da Morologia Matemáica para Análise de Imagens binárias.* IX Escola de Computação, 1994.

[3] S F Barrett. An improved morris water maze tracking algorithm for psychophysical studies. *Biomed Sci Instrum*, 8:36–263, 2000.

[4] S. Beucher and F. Meyer. *The morphological approach to segmentation: the watershed trasformation*. Marcel Decker, Inc, New York, 1992.

[5] J. Y. Bouguet. Pyramidal implementation of the lucas kanade feature tracker. description of the algorithm, 2000.

[6] Y. Cheng, V. Wu, R. Collins, A. Hanson, and E. Riseman. Maximum-weight bipartite matching technique and its application in image feature matching, 1996.

[7] Patrick de Smet and Rui Pires. Implementation and analysis of an optimized rainfalling watershed algorithm (post paper). *IS and T/SPIES's 12th Annual Sympposium Electronic Imaging 2000: Science and Technology, Conference: Imageand Video Communications and Processing (ei26)*, 23–28, January 2000.

[8] R. H. Castanheira de Souza and Neucimar J. Leite. Sistema automático de rastreamento para o labirinto aquático de morris. In *WIC – Sibgrapi*, 2005.

[9] Marco Dorigo and Gianni Di Caro. The ant colony optimization meta-heuristic. pages 11–32, 1999.

[10] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 26(1):29–41, 1996.

[11] D. Eilam. Open-field behavior withstands drastic changes in arena size. *Behavioural Brain Research*, 142:53–62, 2003.

[12] P. Figueroa, N. Leite, R.M.L. Barros, I. Cohen, and G. Medioni. Tracking soccer players using the graph representation. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 4, pages 787– 790, 2004.

[13] S. Goldenstein. A gentle introduction to predictive filters. *Revista de Informatica Teorica e Aplicada*, 11(1):61–89, 2004.

[14] Jens Gramm, Jiong Guo, Falk Huffner, and Rolf Niedermeier. Data reduction, exact, and heuristic algorithms for clique cover. In .SIAMress, editor, *8th Workshop on Algorithm Engineering and Experiments (ALENEX 06)*, pages 86 – 94, 2006.

[15] Michael Grimaud. A new measure of contrast: Dynamics. *Proc. Image Algebra and Morphological Image Processing III*, pages 294–305, 1992.

[16] G. Gutin and A. Punnen. Traveling salesman problem and its variations, 2002.

[17] Henk J. A. M. Heijmans. *Morphological Image Operators*. Academic Press, 1994.

[18] HVS Image. Water 2020 — software specifications. *http://www.hvsimage.com/information/specifications/*, 2004.

[19] Z. Kalafatic. Model-based tracking of laboratory animals. In *EUROCON 2003. Computer as a Tool. The IEEE Region 8*, volume 2, pages 22–24, 2003.

[20] Z. Kalafatic, S. Ribaric, and V. Stanisavljevic. A system for tracking laboratory animals based on optical flow and active contours. In *Image Analysis and Processing, 2001. Proceedings. 11th International Conference on*, pages 334–339, 2001.

[21] A. J. Spink L. P. J. J. Noldus and R. A. J. Tegelenbosch. Ethovision: A versatile video tracking system for automation of behavioral experiments. *Behavior Research Methods, Instruments and Computers*, 3:398–414, 2001.

[22] Morris R. G. M. Developments of a water-maze procedure for studying spatial learning in the rat. *Jounal of Neuroscience Methods*, 11:47–60, 1984.

[23] V Hlavac M Sonka and R Boyle. *Image processing, analysis, and machine vision*. PWS Publishing, 1999.

[24] Elena Marchiori. Genetic, iterated and multistart local search for the maximum clique problem. In *Proceedings of the Applications of Evolutionary Computing on EvoWorkshops 2002*, pages 112–121, London, UK, 2002. Springer-Verlag.

[25] Paul Martin and Patricki Bateson. *Measuring Behaviour: an introductory guide*. Cambridge University Press, 1988.

[26] Morris. Spatial localization does not require the presence of local cues, learning and motivation. *Learning and Motivation*, 12:239–260, 1981.

[27] Tatiana Rodrigues Nahas. O teste do campo aberto. http://www.ib.usp.br/ gfxavier/cap11.html.

[28] P. Soille. *Morphological Image Analysis*. Springer, 2003.

[29] Noldus Information Technology. Noldus ethovision 3.0 — technical specifications. *http://www.noldus.com/download/ev_ techn_ specs_ v30.pdf*, 2002.

[30] Wikipedia. Mahalanobis distance — wikipedia, the free encyclopedia, 2008. [Online; accessed 13-April-2008].