UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

YUMI MONMA

ALGORITMO RÁPIDO PARA SEGMENTAÇÃO DE VÍDEOS UTILIZANDO AGRUPAMENTO DE CLUSTERS

Porto Alegre
2014

YUMI MONMA

ALGORITMO RÁPIDO PARA SEGMENTAÇÃO DE VÍDEOS UTILIZANDO AGRUPAMENTO DE CLUSTERS

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Rio Grande do Sul como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica.
Área de concentração: Engenharia da Computação - Processamento de Sinais

ORIENTADOR: Prof. Dr. Jacob Scharcanski

Porto Alegre
2014

YUMI MONMA

ALGORITMO RÁPIDO PARA SEGMENTAÇÃO DE VÍDEOS UTILIZANDO AGRUPAMENTO DE CLUSTERS

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica e aprovada em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____
Prof. Dr. Jacob Scharcanski, UFRGS
Doutor pela University of Waterloo - Waterloo, Canadá

Banca Examinadora:

Profa. Dra. Letícia Vieira Guimarães, UERGS
Doutora pelo Muroran Institute of Technology - Muroran, Japão

Prof. Dr. Wilson Gavião Neto, UniRITTER
Doutor pela Universidade Federal do Rio Grande do Sul - Porto Alegre, Brasil

Prof. Dr. Walter Fetter Lages, UFRGS
Doutor pelo Instituto Tecnológico de Aeronáutica - São José dos Campos, Brasil

Coordenador do PPGEE: _____
Prof. Dr. Alexandre Sanfelice Bazanella

Porto Alegre, outubro de 2014.

AGRADECIMENTOS

Ao Prof. Dr. Jacob Scharcanski, pela excelente metodologia de orientação e valiosos conselhos.

Ao Prof. Dr. Luciano Silva, por disponibilizar sua pesquisa de doutorado como base para este trabalho e pela ajuda prestada.

Aos professores que me orientaram durante a graduação.

RESUMO

Este trabalho propõe um algoritmo rápido para segmentação de partes móveis em vídeo, tendo como base a detecção de volumes fechados no espaço tridimensional. O vídeo de entrada é pré-processado com um algoritmo de detecção de bordas baseado em linhas de nível para produzir os objetos. Os objetos detectados são agrupados utilizando uma combinação dos métodos de *mean shift clustering* e meta-agrupamento. Para diminuir o tempo de computação, somente alguns objetos e quadros são utilizados no agrupamento. Uma vez que a forma de detecção garante que os objetos persistem com o mesmo rótulo em múltiplos quadros, a seleção de quadros impacta pouco no resultado final. Dependendo da aplicação desejada os grupos podem ser refinados em uma etapa de pós-processamento.

Palavras-chave: Processamento digital de sinais, segmentação de vídeos, segmentação baseada em movimento, segmentação baseada em objetos, composição de clusters.

ABSTRACT

This work presents a very fast algorithm to segmentation of moving parts in a video, based on detection of surfaces of the scene with closed contours. The input video is preprocessed with an edge detection algorithm based on level lines to produce the objects. The detected objects are clustered using a combination of mean shift clustering and ensemble clustering. In order decrease even more the computation time required, two methods can be used combined: object filtering by size and selecting only a few frames of the video. Since the detected objects are coherent in time, frame skipping does not affect the final result. Depending on the application the detected clusters can be refined using post processing steps.

Keywords: Digital Signal Processing, Video Segmentation, Movement-based Segmentation, Object-based segmentation, Ensemble Clustering.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

CSPA Cluster-based Similarity Partitioning Algorithm

HGPA HyperGraph Partitioning Algorithm

MCLA Meta-Clustering Algorithm

LISTA DE SÍMBOLOS

p	Posição *(x,y)* na imagem
$I(p)$	Brilho na posição *p*
$R(p)$	Valor do canal vermelho na posição *p*
$G(p)$	Valor do canal verde na posição *p*
$B(p)$	Valor do canal azul na posição *p*
N	Número de níveis para detecção de bordas
Λ	Conjunto de limiares para detecção de bordas
λ_n	Limiar associado ao nível *n*
$X_{\lambda n}(p)$	Imagem onde $I(p) \geq \lambda_n$
R	Raio do círculo para operação morfológica de dilatação
$C_{\lambda n}(p)$	Imagem com contornos das áreas presentes em $X_{\lambda n}(p)$
$S(p)$	Imagem com a contagem de contornos em cada posição
δ	Limiar para detecção de bordas
$L(p)$	Imagem contendo as bordas detectadas
ΔF	Intervalo entre dois quadros selecionados para agrupamento
N_{obj}	Número de objetos
$N_{quadros}$	Número de quadros
$N_{feicoes}$	Número de feições
T	Tamanho da janela temporal
$\Delta_{feicoes}(f, T)$	Subconjunto da matriz de dados para particionamento inicial
C	Estrutura de dados para armazenar particionamentos iniciais
$Data_{in}$	Dados de entrada do *mean shift clustering*
b	Largura de banda do *mean shift clustering*
λ_1	Grupo detectados no *mean shift clustering*
H	Hiper-grafo dos grupos
h_n	Hiper-aresta

Z	Árvore hierárquica de grupos
Z_n	Nível n na árvore Z
$P(objeto, grupo)$	Probabilidade de um objeto pertencer a um meta-grupo

1 INTRODUÇÃO

Segmentação é a tarefa de analisar automaticamente uma cena em vídeo e identificar quais pixels pertencem a uma região de interesse, e quais são irrelevantes (JAHNE, 2005). Essa identificação tem por objetivo entregar dados significativos para algoritmos de análise e compressão.

As abordagens tradicionais de segmentação de vídeos tratam os mesmos como um conjunto de imagens em duas dimensões, onde a maior distância temporal considerada para análise é a diferença entre dois quadros subseqüentes.

Recentemente, abordagens tridimensionais, onde além das dimensões espaciais é feita a análise no domínio temporal tem sido objeto de estudo. A análise tridimensional permite uma visão ao longo de todo o tempo percorrido do vídeo, possibilitando a extração de informações que vão além do que é visto na abordagem tradicional, por exemplo movimento.

Em termos de abordagens tridimensionais, a análise da evolução do estado da arte, apresentada no capítulo a seguir, indica que enquanto houve um grande aumento na qualidade dos resultados de segmentação de vídeo, não houve redução significativa no tempo de processamento. Isso se deve ao fato da maioria das abordagens ser baseada em acompanhamento de trajetória de partículas.

Em seu trabalho intitulado *Video Segmentation Based on Motion Coherence of Particles in a Video Sequence* (SILVA; SCHARCANSKI, 2010), Silva e Scharcanski apresentam um método de segmentação de vídeos baseado em movimentos coerentes de partículas. O método de citado possui taxa de acerto superior a 99% em vídeos onde as partes em movimento se movem de forma diferente do fundo. O tempo de processamento exigido é da ordem de diversas horas, o que o torna bastante custoso para aplicações práticas. O gargalo no processamento são as etapas iniciais de fluxo óptico e acompanhamento das partículas. Estas etapas representam de 35 a 45% do tempo total de processamento.

A contribuição proposta deste trabalho é uma modificação do método de (SILVA; SCHARCANSKI, 2010), mantendo seu desempenho porém apresentando tempo de processamento reduzido. A modificação proposta é a substituição do uso de partículas por túneis tridimensionais delimitadas por contornos fechados. Cada túnel representa regiões do vídeo que seriam representadas por muitas partículas, reduzindo o volume de dados para o agrupamento. Também é utilizada outra técnica de redução do volume de dados, a seleção de quadros, onde é selecionado para processamento somente um quadro a cada intervalo fixo.

O restante desta dissertação está organizada em quatro partes. O capítulo 2 apresenta uma análise de publicações acadêmicas sobre segmentação de vídeos baseada em movimento, buscando demonstrar a evolução dos métodos utilizados e estabelecer o estado da arte nesse tópico. Também é apresentado em mais detalhes o método de segmentação

baseado em movimento coerente de partículas de (SILVA; SCHARCANSKI, 2010).

O capítulo 3 apresenta o método proposto e as etapas envolvidas em sua execução: pré-processamento, agrupamento e pós-processamento. A etapa de pré-processamento foi desenvolvida e implementada para este método, e consiste na detecção de bordas dos quadros utilizando um algoritmo de linhas de nível de (NEGRI; LOTITO, 2012); detecção dos túneis tridimensionais (chamados de *objetos*) através de um algoritmo desenvolvido para este trabalho; seleção de quadros e criação de uma estrutura de dados utilizada na etapa seguinte do método. A etapa de agrupamento segue o mesmo esquema utilizado em (SILVA; SCHARCANSKI, 2010), onde os dados são particionados em subgrupos, segmentados individualmente e a seguir é calculado o consenso entre as partições utilizando o algoritmo de meta-agrupamento (STREHL; GHOSH, 2003). Já a etapa de pós-processamento foi desenvolvida para este trabalho a fim de refinar o resultado de segmentação e remover possíveis erros.

O capítulo 4 apresenta os resultados experimentais obtidos pelo método proposto. Foram realizados testes em vídeos adquiridos em ambiente controlado e vídeos de bases públicas de dados. São apresentadas comparações de desempenho e tempo de processamento entre o método proposto e o de (SILVA; SCHARCANSKI, 2010).

Finalmente o capítulo 5 apresenta as conclusões, limitações e sugestão de trabalhos futuros.

2 SEGMENTAÇÃO DE VÍDEOS: CONCEITOS E O ESTADO DA ARTE

Na segmentação de vídeos regiões de interesse são localizadas e identificadas para uso posterior. Duas abordagens clássicas utilizadas são segmentação espacial e segmentação temporal.

Segmentação espacial ocorre quando os quadros do vídeo são analisados de forma individual e particionados de acordo com as feições que sua imagem apresenta. Usualmente o processo de particionamento é baseado em regiões ou em limites (KONRAD, 2007). O particionamento baseado em regiões busca encontrar áreas semelhantes na imagem. Já o particionamento baseado em limites busca encontrar descontinuidades que dividam a imagem.

Já a segmentação temporal de vídeos ocorre quando procura-se agrupar quadros que pertençam às mesma cenas. Usualmente isso é feito através da detecção dos quadros delimitadores, tendo como base uma análise temporal de cada quadro do vídeo.

A combinação de segmentação temporal e espacial fez surgir a segmentação espaço-temporal de vídeos, uma nova abordagem onde acompanha-se através do tempo o comportamento das regiões segmentadas no espaço. Um exemplo de segmentação espaço-temporal é detectar os contornos das partes presentes em um quadro do vídeo e identificar onde elas se localizam nos quadros seguintes. Quando os quadros contendo contornos são vistos com uma perspectiva de seqüência de imagens, os contornos passam a delimitar volumes fechados, também chamados de túneis espaço-temporais (RISTIVOJEVIC; KONRAD, 2006).

Os métodos de segmentação que utilizam túneis são considerados tridimensionais. Isso porque eles não somente analisam o vídeo no domínio espacial *(x,y)* em cada quadro como também no domínio da seqüência dos quadros, ou temporal *(t)*. A principal variável utilizada nessa análise é o movimento detectado nos quadros do vídeo. Uma visualização da análise tridimensional de vídeos baseada em movimento é apresentada na Figura 1.

Analisando na literatura o progresso da área de pesquisa de segmentação de vídeos baseada em movimento, observa-se que o escopo se torna cada vez mais amplo e os resultados mais refinados, e com mais aplicações. Algumas aplicações relevantes atualmente são detecção automática de eventos, compreensão de vídeos e codificação de vídeos baseada em objetos.

A seguir é apresentada a análise de trabalhos que representam a evolução dos algoritmos de segmentação tridimensional baseada em movimento.

Em (MANSOURI; KONRAD, 2003) é considerado que a abordagem padrão até o momento é a segmentação do vídeo entre regiões estáticas e regiões em movimento. O trabalho sugere uma metodologia utilizando *conjuntos de nível*, que se trata da obtenção

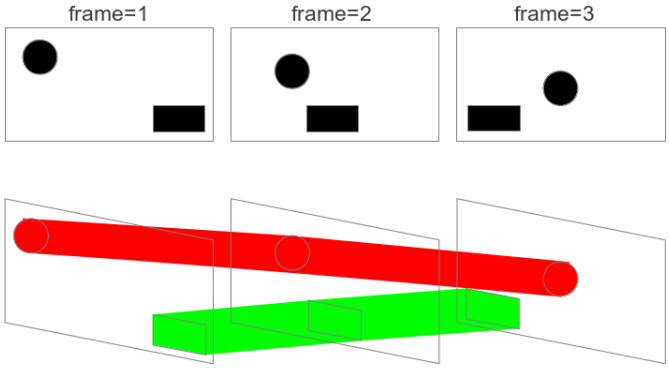
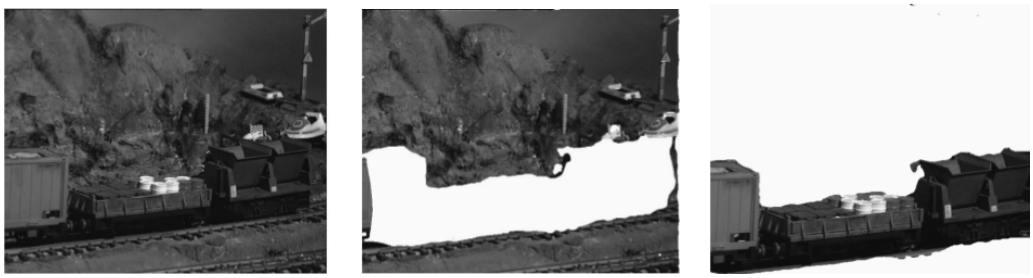


Figura 1: Túneis tridimensionais obtidos a partir do acompanhamento das posições (movimento) de duas partes móveis em três quadros. Cada túnel é representado por uma cor.

de uma curva fechada que possua a maior probabilidade de delimitar a região em movimento em cada quadro. Essa análise leva em conta somente movimento e não intensidade ou cor dos pixels do quadro. Através da comparação da curva fechada entre quadros subseqüentes chega-se em um modelo de transformação afim da região em movimento. No mesmo trabalho são apresentadas duas expansões para a metodologia: a primeira é o uso da intensidade dos pixels como auxiliar em casos onde as bordas de intensidade coincidem com as bordas das regiões de movimento, e a segunda é a detecção de múltiplas regiões em movimento. A Figura 2 mostra o resultado de segmentação da seqüência *Train*. Em experimentos, o tempo de processamento da sequencia *Train*, de tamanho 720x576 pixels, em PC com processador Pentium II na frequencia de 450MHz foi superior a vinte horas.

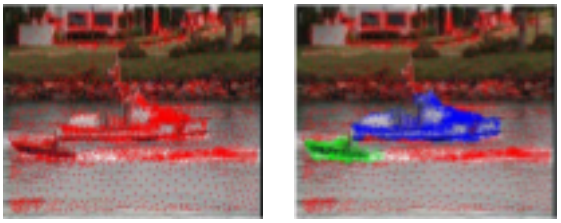


(a) (b) (c)

Figura 2: Resultado de segmentação de (MANSOURI; KONRAD, 2003) na seqüência *Train*. (a) Quadro do vídeo original; (b) Região de fundo; (c) Região em movimento.

O trabalho de (SILVA; SCHARCANSKI, 2010) apresenta um método de segmentação baseado em movimento coerente de partículas, bastante diferente da abordagem de curva fechada de (MANSOURI; KONRAD, 2003). Partículas são pontos amostrados adaptativamente nos quadros do vídeo. A sua movimentação entre os quadros é estimada com uma versão modificada do método de estimativa de trajetórias de (SAND; TELLER, 2008), composta de quatro passos executados a cada quadro: adição de partículas, onde partículas são adicionadas ao quadro nas regiões onde a análise de textura indica haver detalhes, de forma que regiões homogêneas recebem poucas partículas; propagação de partículas, onde partículas existentes em um dado quadro t são propagadas para o quadro seguinte $t+1$; remoção de partículas, onde são removidas partículas que se tornam oclusas ou saem do campo de visão do quadro; otimização da localização das partículas, onde as

partículas que têm como origem propagação do quadro anterior têm sua localização corrigida para evitar propagação de erros. Após a estimativa de movimento das partículas, as mesmas são agrupadas utilizando o algoritmo de meta-agrupamento (STREHL; GHOSH, 2003), onde o conjunto total de partículas a agrupar é dividido em subconjuntos (representando diferentes pontos de vista do conjunto total). Os subconjuntos são agrupados individualmente utilizando *mean shift clustering* e a seguir é calculado um consenso entre os resultados, chegando-se aos grupos finais (meta-grupos). Uma vez que a aplicação do algoritmo de meta-agrupamento utilizada por (SILVA; SCHARCANSKI, 2010) é utilizada neste trabalho, ela é explicada com mais detalhes na seção 3.2. Após o meta-agrupamento é extraída a segmentação densa, onde os pixels do vídeo são designados ao grupo de uma das partículas próximas. Essa escolha leva em conta localização espacial, movimento e um coeficiente de suavização. A Figura 3 mostra o resultado da detecção de partículas e as mesmas após meta-agrupamento em um quadro da seqüência *coastguard*. O método foi implementado em código MATLAB não otimizado, e o tempo de processamento de um vídeo de 50 quadros com resolução 352x288 pixels foi cerca de 7 horas em um PC com processador *Pentium III* rodando a uma freqüência de 1.6 GHz.



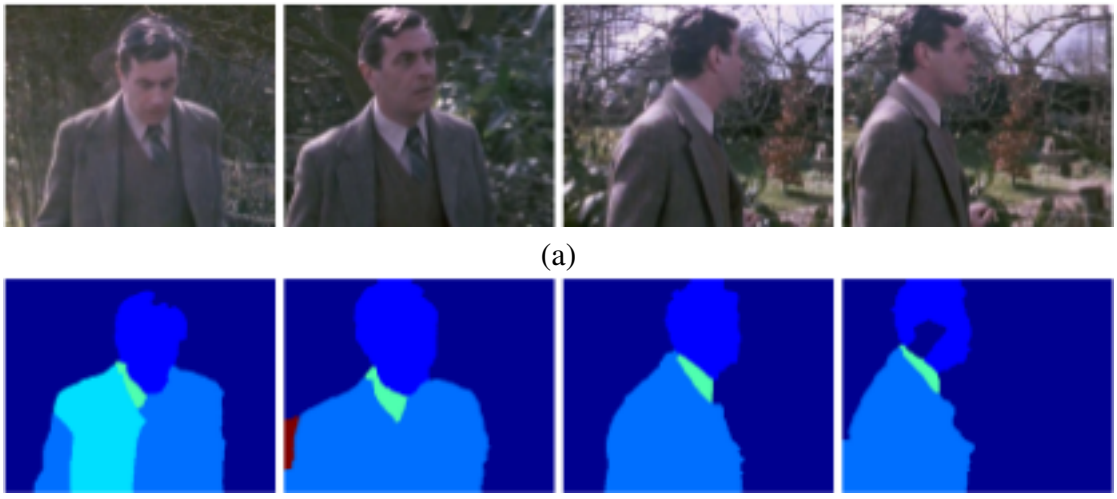
(a) (b)

Figura 3: Resultado de segmentação de (SILVA; SCHARCANSKI, 2010) na seqüência *coastguard*. (a) Partículas detectadas; (b) Partículas após meta-agrupamento.

Já o trabalho de (SUNDARAM; KEUTZER, 2011) difere dos anteriores por não utilizar análises probabilísticas, preferindo abordar o problema de segmentação de vídeos como uma generalização de segmentação de imagens. É utilizado o algoritmo de detecção de bordas gPb (MAIRE et al., 2008) para segmentação individual dos quadros. Em seguida, são utilizados algoritmos de fluxo óptico e agrupamento espectral para decidir a afinidade dos pixels entre os quadros, levando em consideração intensidade, cor, textura e movimento. Para encontrar pontos onde o cálculo do fluxo óptico pode apresentar um resultado incorreto é calculada uma medida de confiança. A medida parte da premissa que o fluxo óptico entre um dado quadro *i* e o quadro *i+1* deve ser semelhante ao fluxo entre os quadros *i* e *i-1*. Esta abordagem de segmentação requisita uma grande quantidade de processamento. Para compensar isso o equipamento utilizado foi um *grid* de 34 unidades de processamento gráfico contendo cada uma 3GB de memória RAM e 14 processadores. Nesse equipamento, uma seqüência de 200 quadros e tamanho 352x288 pixels era processada em cinco minutos. A Figura 4 mostra o resultado de segmentação do algoritmo.

Em (DIMITRIOU; DELOPOULOS, 2013), é introduzida uma abordagem de segmentação baseada em movimento especialmente adaptada para vídeos longos. O vídeo é dividido em N janelas temporais $\{w_i\}_{i=1..N}$. As janelas são escolhidas nos quadros de forma que haja uma sobreposição de quadros, por exemplo 2, conforme a Figura 5.

Em cada janela são identificadas partículas. É utilizado um modelo de transformação afim para obter a trajetória completa de cada partícula dentro da janela. As partículas



(a)

(b)

Figura 4: Resultado de segmentação de (SUNDARAM; KEUTZER, 2011). (a) Quadros dos vídeos originais; (b) regiões segmentadas.

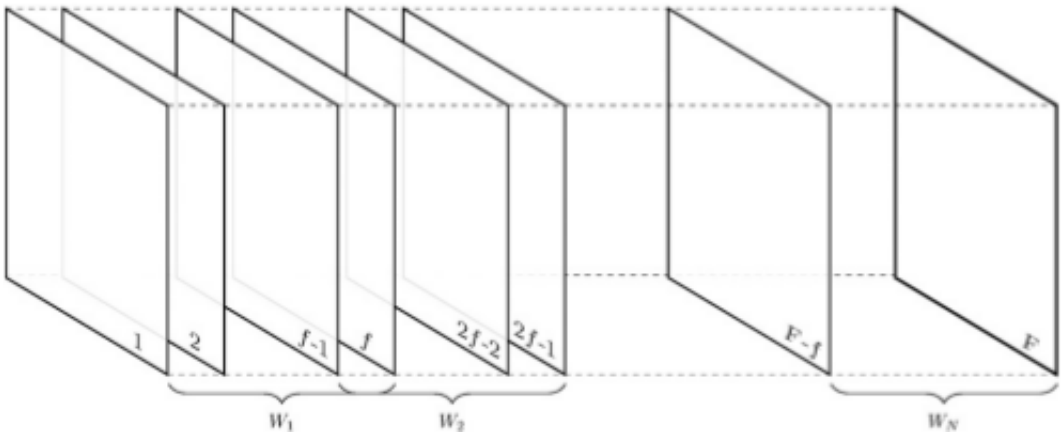
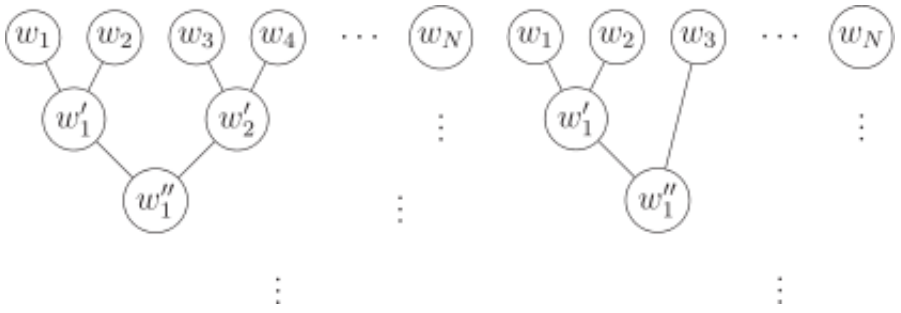


Figura 5: Esquema de escolhas dos quadros pertencente a cada janela temporal $\{w_i\}_{i=1..N}$ em (DIMITRIOU; DELOPOULOS, 2013). As janelas são escolhidas de forma que haja sobreposição de dois quadros.

são agrupadas em cada janela de acordo com seu movimento utilizando o algoritmo de núcleo ordenado residual (CHIN; WANG; SUTER, 2009). É possível que haja sobre-segmentação após o agrupamento nas janelas, mas isso é compensando na fusão dos resultados. A fusão pode ser realizada de duas maneiras: fundindo duas janelas vizinhas por vez a cada passo ou fundindo somente as duas primeiras e adicionando a janela seguinte a cada passo. A primeira forma é mais indicada para vídeos onde toda a aquisição já foi realizada e a segunda para vídeos que são processados conforme são adquiridos. Uma representação gráfica dos dois esquemas pode ser visto na Figura 6.



(a) (b)

Figura 6: Esquemas de fusão de janelas em (DIMITRIOU; DELOPOULOS, 2013). (a) Fusão de duas janelas vizinhas a cada passo. (b) Fusão das duas primeiras janelas e adição da janela subseqüente a cada passo.

O algoritmo foi avaliado com as bases de dados *Berkeley motion segmentation benchmark* (BROX; MALIK, 2010) (base que possui os vídeos da base Hopkins155 (TRON; VIDAL, 2007) e adiciona outros). Em testes com vídeos contendo de 19 a 800 quadros, apresenta um tempo de processamento de pelo menos 50h. A Figura 7, extraída de (DIMITRIOU; DELOPOULOS, 2013), apresenta o esquema geral do método.

O trabalho de (PALOU; SALEMBIER, 2013) explora o diferença no comportamento através do tempo quando se utiliza segmentação baseada em cor e textura ou baseada em movimento. Enquanto cor e textura diferenciam bastante as regiões no vídeo e são mais simples de segmentar no contexto de um quadro, a segmentação baseada em movimento identifica regiões mais densas e coerentes ao longo de vários quadros. Dessa forma o trabalho propõe utilizar cor, textura e movimento para segmentação, abordando cada informação de forma apropriada. O movimento das partículas é estimado utilizando algoritmo de fluxo óptico descrito em (SUNDARAM; BROX; KEUTZER, 2010). Já as cores são utilizadas para segmentar o vídeo considerando um número máximo de oito cores dominantes no sistema de cores *Lab*, agrupadas utilizando algoritmo de *k-means*. Ao final as partículas são agrupadas considerando a distância entre suas trajetórias e regiões de cor. Uma representação dessa abordagem é apresentada na Figura 8.

Após a segmentação do vídeo é construída uma estrutura hierárquica chamada *BPT - Binary Partition Tree*, onde no primeiro nível se encontram as regiões detectadas no vídeo e a cada nível seguinte os pares de regiões mais similares entre si (em termos de movimento e cor) são fundidas. Esse processo se repete até o nível onde todas as regiões foram fundidas. Essa estrutura tem por objetivo ser uma forma prática de se analisar os vídeos posteriormente, uma vez que ela pode ser cortada em regiões que atendam a determinadas características. A Figura 9 demonstra a criação da BPT. Em termos de desempenho, esta proposta exige mil segundos e 20GB de memória RAM em vídeo de três milhões de voxels (três quadros em resolução 640x480 pixels).

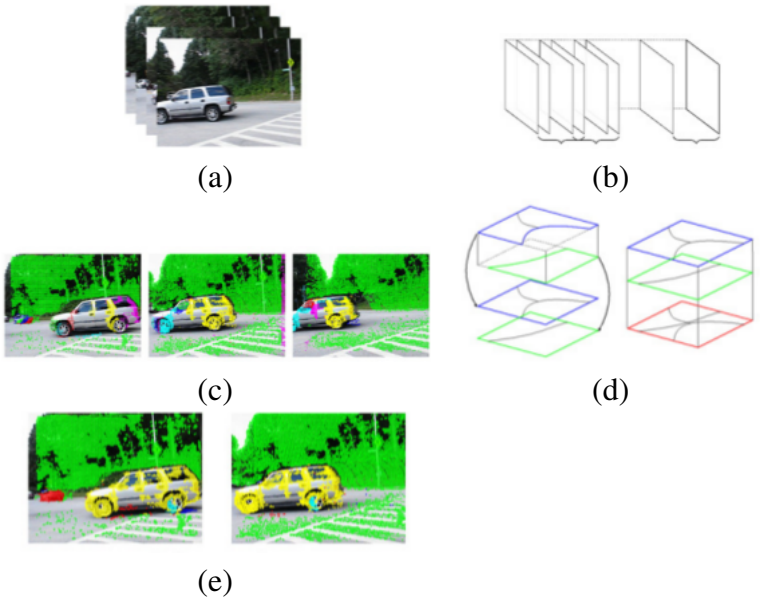


Figura 7: Esquema do método apresentado em (DIMITRIOU; DELOPOULOS, 2013); (a) Quadros do vídeo original; (b) são selecionadas janelas temporais entre os quadros; (c) as janelas são segmentadas independentemente; (d) é feita a fusão dos resultados de segmentação das janelas temporais; (d) segmentação final.

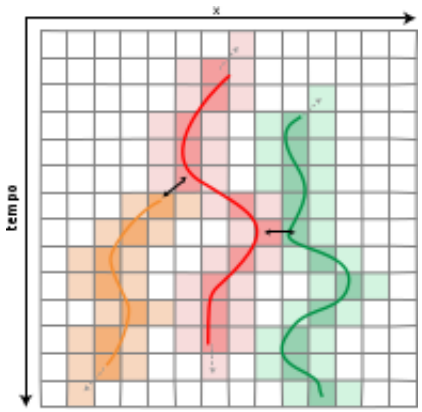


Figura 8: Representação do esquema de segmentação através de cor e trajetória de (PALOU; SALEMBIER, 2013).

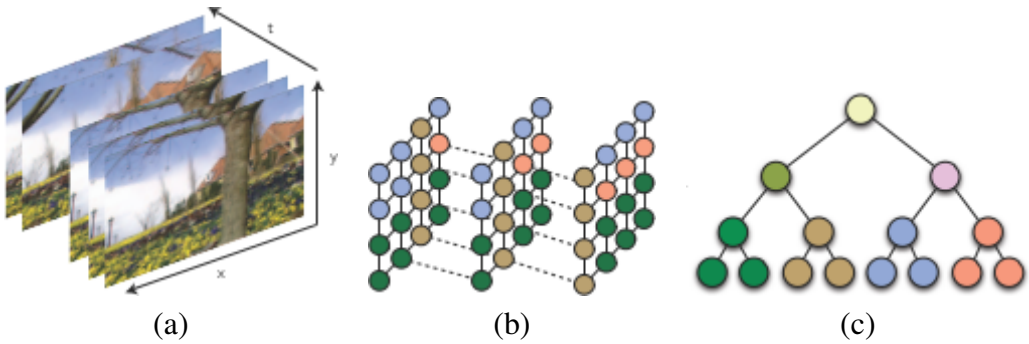


Figura 9: Criação da representação hierárquica proposta por (PALOU; SALEMBIER, 2013); (a) quadros do vídeo original; (b) é feita uma análise de cor e movimento nos quadros; (c) o vídeo é reorganizado na árvore hierárquica.

Também pode ser encontrado na literatura trabalhos que utilizam segmentação baseada em movimento para aplicações diferentes da de pré-processamento de vídeos. Em (BELAGIANNIS et al., 2012) é proposto o uso de acompanhamento de trajetória de partículas para monitorar o comportamento de um objeto específico (p.ex. um carro ou uma pessoa) ao longo do vídeo. É proposto o uso de partículas em contraste ao uso de um retângulo envolvente, comum para esse tipo de aplicação. O trabalho afirma que o retângulo mistura partes do fundo da cena com o objeto monitorado, criando um erro que se propaga, já as partículas criam um contorno exato do objeto, como pode ser visto na Figura 10.

Figura 10: Contornos dos objetos monitorados em (BELAGIANNIS et al., 2012). Os contorno correspondem exatamente ao tamanho do objeto.

O algoritmo proposto determina automaticamente o contorno do objeto a partir de uma marcação manual retangular no quadro inicial, de forma que os pixels na região externa à marcação são considerados fundo, enquanto os pixels na região interna são considerados indeterminados (entre fundo e objeto). É utilizado o algoritmo de *GrabCut* (ROTHER; KOLMOGOROV; BLAKE, 2004) para determinar o contorno do objeto dentro da região retangular. O *GrabCut* extrai automaticamente objetos dentro de retângulos marcados sem precisão em imagens estáticas, utilizando a marcação para determinar modelos para o objeto e para o fundo. Exemplos de segmentação entre objeto e fundo utilizando *GrabCut* podem ser vistos na Figura 11.

Figura 11: Segmentação entre objeto e fundo a partir de uma marcação manual retangular obtida pelo algoritmo *GrabCut* (ROTHER; KOLMOGOROV; BLAKE, 2004).

Seguindo o método de (BELAGIANNIS et al., 2012), nos quadros seguintes a marcação retangular inicial é feita a partir do formato atual do objeto monitorado, e assim sucessivamente. O trabalho não trata o vídeo como uma variável tridimensional, mas propõe fazê-lo em trabalhos futuros para obter melhores resultados. O tempo de processamento do método é em média 30 milissegundos por quadro. Esse tempo não pode ser comparado com o dos outros métodos de segmentação apresentados pois neste algoritmo é feito monitoramento de somente um objeto por vídeo, e os quadros são processados no entorno da posição do objeto. Nos outros trabalho analisados o processamento é realizado nos quadros inteiros.

3 MÉTODO PROPOSTO PARA SEGMENTAÇÃO ESPAÇO-TEMPORAL DE VÍDEOS

O método proposto é dividido em três etapas: pré-processamento, agrupamento e pós-processamento. A etapa de pré-processamento tem como entrada um vídeo, e consiste nos seguintes passos: conversão para tons de cinza; detecção de bordas, utilizando o algoritmo de linhas de nível (NEGRI; LOTITO, 2012); detecção de objetos, que são os volumes fechados no espaço tridimensional delimitados pelas bordas; seleção de quadros do vídeo, onde somente alguns quadros são amostrados; e criação da matriz de dados, estrutura utilizada na etapa seguinte de agrupamento. O detalhamento de cada passo do pré-processamento é apresentado na seção 3.1.

Na etapa de agrupamente é utilizado o algoritmo de meta-agrupamento (STREHL; GHOSH, 2003). O algoritmo consiste em primeiramente dividir os dados de entrada em subconjuntos, que representam diferentes pontos de vista do conjunto total. Os subgrupos são agrupados individualmente utilizando um algoritmo de deslocamento de média (*mean shift clustering*) (FUKUNAGA; HOSTETLER, 1975). Em seguida é calculado um consenso utilizando o algoritmo de meta-agrupamento. O detalhamento da etapa de agrupamento é apresentado na seção 3.2.

A etapa final de pós-processamento tem por objetivo refinar o particionamento através da remoção do meta-grupo de fundo (*background*), remoção de *outliers* (artefatos que aparecem rapidamente no vídeo, em geral resultado de erro) e aplicação de restrições espaciais e temporais que melhoram o agrupamento em vídeos com características de movimento específicas. Por fim as regiões do vídeo onde estão os meta-grupos detectados são marcados com o menor polígono convexo envolvente. O detalhamento da etapa de pós-processamento é apresentado na seção 3.3.

Das três etapas principais do método, a de agrupamento advém diretamente do método de segmentação baseado em movimento coerente de partículas (SILVA; SCHARCANSKI, 2010). A etapa de pré-processamento é inteiramente nova, e representa a parte principal da proposta deste trabalho. A etapa de pós-processamento é conseqüência dos passos de pré-processamento e é utilizada para adequar o resultado para comparação com outros métodos. A Figura 12 mostra as etapas do método e seus passos intermediários, reforçando qual a contribuição deste trabalho em comparação com (SILVA; SCHARCANSKI, 2010).

3.1 Pré-processamento

Esta seção apresenta os passos que compõe a etapa de pré-processamento. O dado de entrada do pré-processamento é o vídeo a ser segmentado. O primeiro passo é a con-

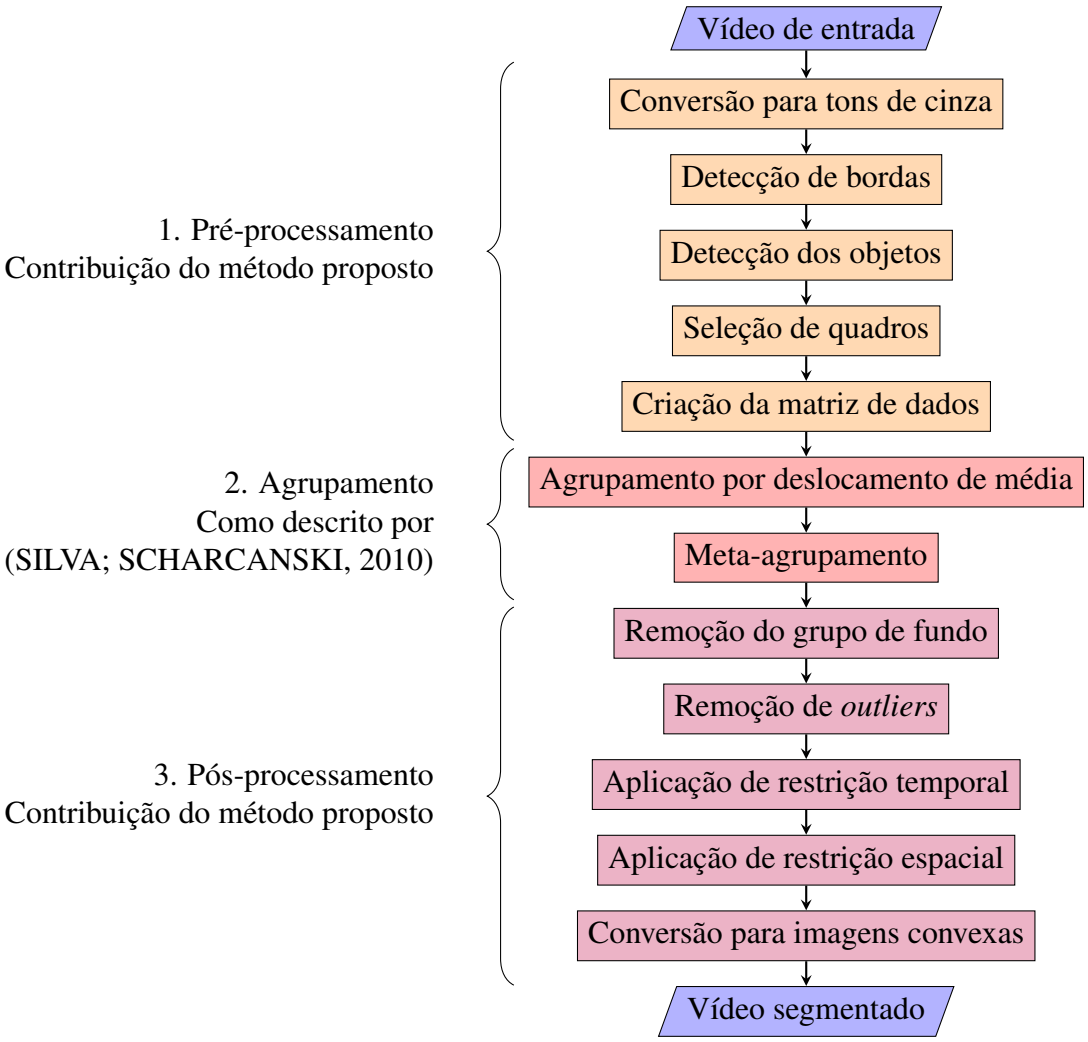


Figura 12: Esquema geral do método proposto

versão dos quadros do vídeo para escala de cinza. Em seguida são detectadas as bordas utilizando algoritmo de linhas de nível (NEGRI; LOTITO, 2012). Após são detectados os objetos, túneis no espaço tridimensional do vídeo delimitados pelas bordas. O quarto passo consiste em selecionar alguns quadros onde os dados serão amostrados para o agrupamento. O último passo é a criação de uma matriz em formato específico contendo os dados amostrados.

3.1.1 Conversão para tons de Cinza

O método proposto se aplica a vídeos com um único canal. No caso do vídeo de entrada ser no formato RGB, é realizada a conversão para o tons de cinza. A fórmula utilizada para conversão é da conversão para formato *YUV* (MOESLUND, 2012), contudo é utilizado apenas o canal de intensidade enquanto cor e saturação são descartados. A fórmula utilizada para cálculo da luminância é apresentada na equação 1, onde $I(p)$ representa a luminância da posição p calculada a partir dos valores dos canais vermelho (R), verde (G) e azul (B) em p.

$$I(p) = 0.2989R(p) + 0.5870G(p) + 0.1140B(p) \tag{1}$$

3.1.2 Detecção de bordas

Para obter os túneis que representaram os objetos presentes no vídeo, as bordas na imagem de cada quadro são detectadas utilizando um algoritmo de linhas de nível (*level lines*) apresentado em (NEGRI; LOTITO, 2012). O motivo da escolha de linhas de nível é devido a sua robustez a variações de iluminação e sua tendência a criar contornos fechados. Os passos para detecção de bordas utilizando linhas de nível são apresentados a seguir em conjunto com suas equações e exemplos.

1. A partir de um número N de níveis, é criado um conjunto de limiares Λ que divide igualmente o espaço de tons de cinza em $N + 1$ partições. Cada limiar recebe o rótulo λ_n:

   $$\Lambda = \{\lambda_1, \lambda_2, ..., \lambda_N\}$$

   Por exemplo, para $N = 3$, os limiares presentes em Λ são:

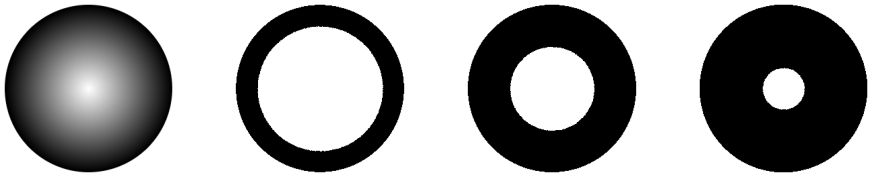
   $$\Lambda = \{\lambda_1 = 64, \lambda_2 = 128, \lambda_3 = 192\}$$

2. Assume-se que a função $I(p)$ indica o brilho em cada posição $p = (x, y)$ da imagem. Para cada λ_n é calculada a imagem $X_{\lambda n}(p)$, que indica os pontos onde $I(p)$ tem um valor mais alto que o limiar associado λ_n:

   $$X_{\lambda n}(p) = p/I(p) \geq \lambda_n$$

   Retomando-se o exemplo anterior, no limiar $\lambda_2 = 128$, todas as posições p com $I(p) \geq 128$ recebem 255 e todas as posições p com $I(p) < 128$ recebem zero. Se para uma dada posição p_1 o valor de $I(p_1)$ é 150, $X_{\lambda_4}(p_1) = 255$. Por outro lado, se em outra posição p_2 o valor de $I(p_2)$ for 50, então $X_{\lambda_4}(p_2) = 0$. A Figura 13 apresenta uma imagem de entrada de exemplo e o valor de cada $X_{\lambda n}(p)$ para $N = 3$.

3. Para cada imagem $X_{\lambda n}(p)$, é calculada a imagem de linhas de nível $C_{\lambda n}(p)$, que contém somente os contornos das regiões marcadas em $X_{\lambda n}(p)$. A obtenção dos



(a) (b) (c) (d)

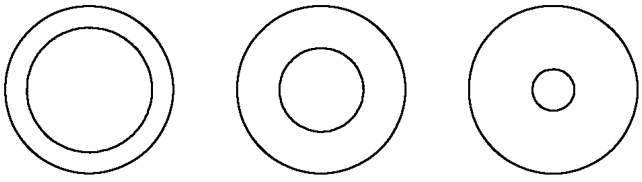
Figura 13: Exemplo de $X_{\lambda n}(p)$: (a) Imagem original; (b) $X_{\lambda 0}(p)$; (c) $X_{\lambda 1}(p)$; (d) $X_{\lambda 2}(p)$;

contornos é realizada através da subtração da imagem após a operação morfológica de dilatação de um círculo de raio R pixels pela imagem original:

$$C_{\lambda n}(p) = (X_{\lambda n}(p) \oplus circle(R)) - X_{\lambda n}(p)$$

Para imagens com contornos bem definidos, $R = 1$ é suficiente, porém, em imagens com problemas de foco ou ruído pode ser necessário um valor de R mais alto. Esta forma de cálculo assegura que a maioria dos contornos detectados serão fechados.

As linhas de nível devem ser calculadas para todos os níveis λ_n. A Figura 14 apresenta $C_{\lambda n}(p)$ para cada $X_{\lambda n}(p)$ mostrado na figura anterior, utilizando $R = 1$.



(a) (b) (c)

Figura 14: Linhas de nível $C_{\lambda n}(p)$: (a) $C_{\lambda 0}(p)$; (b) $C_{\lambda 1}(p)$; (c) $C_{\lambda 2}(p)$;

4. Calcula-se a imagem $S(p)$, onde cada posição possui um valor referente à contagem de linhas de nível sobre a mesma no conjunto $\{C_{\lambda 1}(p), C_{\lambda 2}(p), ..., C_{\lambda N}(p)\}$.

$$S(p) = \sum_{i=1}^{N} C_{\lambda i}(p)$$

O conteúdo de $S(p)$ para o exemplo apresentado pode ser visto na Figura 15.

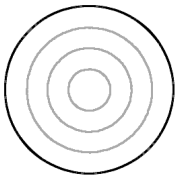
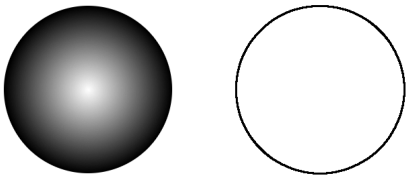


Figura 15: Exemplo de $S(p)$. A linha de cor preta indica $\sum_{i=1}^{N} C_{\lambda i}(p) = 3$ e as linhas de cor cinza $\sum_{i=1}^{N} C_{\lambda i}(p) = 1$

5. Finalmente extrai-se a imagem binária de bordas $L(p)$, que indica as posições de $S(p)$ onde a contagem de linhas de nível é mais alta que um dado limiar δ.

$$L(p) = p/S(p) \geq \delta$$

O resultado final da extração de bordas considerando $\delta = 2$ é apresentado na Figura 16.

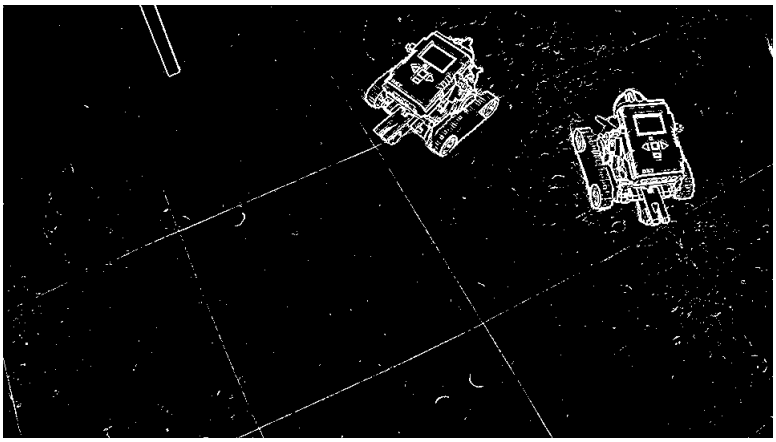


(a) (b)

Figura 16: Resultado final da detecção de bordas. (a) Imagem original; (b) Bordas detectadas $L(p)$.

A escolha dos parâmetros N, R e δ deve ser feita de acordo com as características da imagem. A Figura 17 apresenta o resultado da detecção de bordas em um quadro de vídeo de teste obtido em condições controladas, onde foram utilizados os parâmetros $N = 36$, $R = 3$ e $\delta = 2$:

(a)



(b)

Figura 17: Resultado da extração de bordas em imagem representando quadro de vídeo obtido em ambiente controlado. (a) Imagem original; (b) Bordas obtidas com $N = 36$, $R = 3$ e $\delta = 2$.

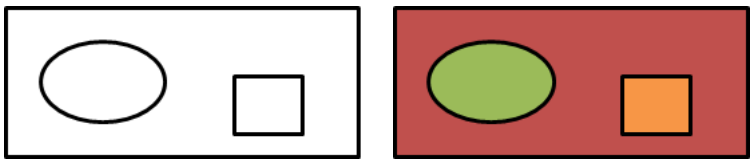
3.1.3 Detecção de objetos

Após a obtenção de bordas, todos os pixels do vídeo podem ser classificados como sendo bordas ou não. A etapa de detecção de objetos tem por objetivo rotular todos os pixels que não pertencem a bordas como pertencentes a algum *objeto*. O uso da palavra *objeto* neste trabalho se refere a um conjunto de pixels no domínio espaço-temporal do vídeo delimitados pelas bordas. Para representar graficamente o estado dos pixels do vídeo durante a detecção de bordas são utilizadas as cores preta para pixels de bordas, branca para pixels não rotulados e demais cores para indicar pixels rotulados como objetos (cada cor indica um objeto único).

O primeiro passo consiste em detectar regiões fechadas de pixels não rotulados no primeiro quadro. Para tanto é utilizado um algoritmo de preenchimento de regiões (BURGER et al., 2009). O algoritmo segue a seguinte ordem:

1. O indicador de próximo rótulo m é inicializado, $m = 1$
2. Para cada ponto p do quadro inicial:
   (a) Se p é não rotulado:
      i. É atribuído a p o rótulo m
      ii. O rótulo m é atribuído a todos os pixels não rotulados vizinhos de p
      iii. Esse processo se repete até que toda o entorno de p delimitado pelas bordas receba o rótulo m
      iv. O indicar de próximo rótulo é incrementado, $m = m + 1$
3. O algoritmo retorna o quadro inicial onde todos os pixels são classificados como bordas ou possuem um rótulo de objeto

Um exemplo gráfico do procedimento de rotulagem do primeiro quadro pode ser visto na Figura 18.

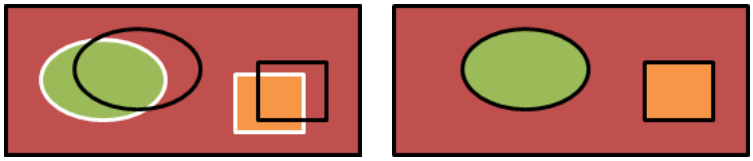


(a) (b)

Figura 18: Detecção de objeto no primeiro quadro. (a) Quadro original, onde os pixels são classificados como bordas (preto) e pixels não rotulados (branco); (b) Quadro após detecção, onde os pixels são classificados como bordas ou objetos (áreas coloridas).

A cada quadro seguinte os seguintes passos são executados:

1. Para cada ponto p não rotulado:
   (a) É detectada sua região delimitada por bordas através de preenchimento de regiões
   (b) É feita uma consulta ao quadro anterior para determinar qual o rótulo cada pixel da região apresentava
   (c) O rótulo mais freqüente é escolhido como rótulo da região



(a) (b)

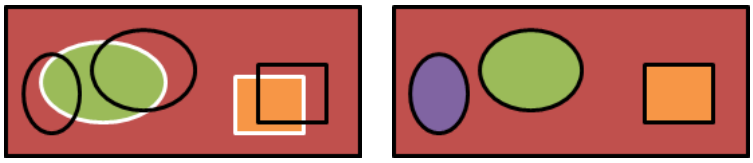
Figura 19: Detecção de objeto nos quadros subseqüentes. (a) Bordas do quadro sobrepostas aos rótulos no quadro anterior; (b) Quadro após detecção, onde cada região fechada recebe o rótulo mais freqüente no quadro anterior.

Uma representação gráfica da detecção de objetos nos quadros subseqüentes ao primeiro é apresentada na Figura 19.

Em etapas iniciais da pesquisa para este trabalho, foi utilizado para detecção de objetos um algoritmo de preenchimento tridimensional. O algoritmo foi trocado pelo método apresentado porque a detecção de regiões tridimensionais permitia a propagação de erros de detecção (a maioria causados por falhas/*gaps* nas bordas em um determinado quadro) para quadros anteriores e seguintes ao quadro onde ocorria o *gap*. Esse efeito é evitado neste método proposto à restrição aplicada que somente exista uma região fechada para cada rótulo a cada quadro. Essa restrição é aplicada da seguinte forma:

1. Ao final do processamento de cada quadro:
   (a) Se duas ou mais regiões fechadas apresentarem o mesmo rótulo:
      i. A região com maior número de pixels mantém o rótulo atual
      ii. As demais regiões recebem um novo rótulo, no valor do indicador de próximo rótulo m
      iii. O indicador de próximo rótulo é incrementado, $m = m + 1$
2. Esse procedimento se repete até que não haja no quadro duas regiões com o mesmo rótulo.

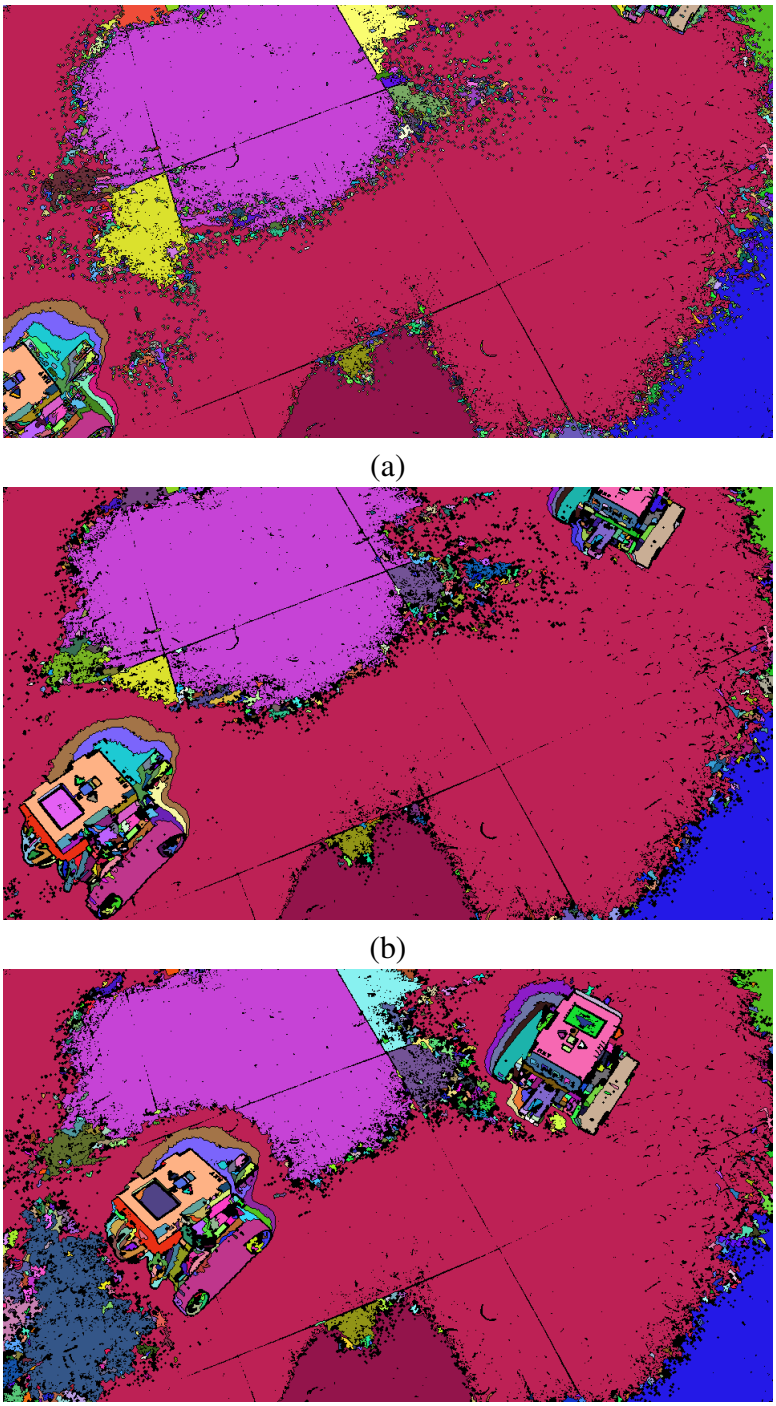
Uma representação gráfica da restrição aplicada após o processamento de cada quadro é apresentada na Figura 20



(a) (b)

Figura 20: Aplicação de restrição ao número de regiões apresentando o mesmo rótulo em um quadro do vídeo. (a) A princípio as duas regiões da esquerda receberiam o rótulo correspondente ao objeto verde (b) Apenas a maior região recebe o rótulo do objeto verde, enquanto a outra recebe um rótulo de novo objeto.

A forma apresentada mostrou-se eficaz para a detecção de objetos persistentes no tempo e representativos às áreas em movimento no vídeo. A restrição de somente um objeto por rótulo em cada quadro evita a propagação de eventuais erros de detecção, ao custo da detecção de um número maior de objetos. A Figura 21 apresenta o resultado da detecção de objetos em três quadros de um vídeo obtido em ambiente controlado.



(a)

(b)

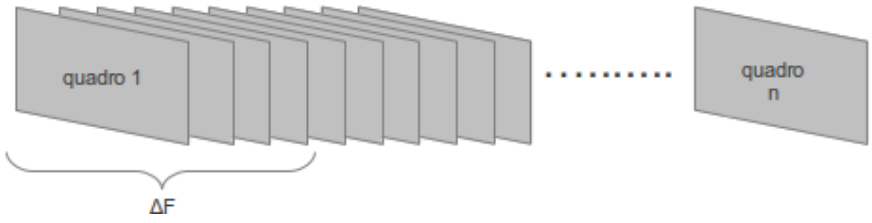
(c)

Figura 21: Detecção de objetos em vídeo de teste. (a) Rótulos atribuídos no quadro 1; (b) Rótulos atribuídos no quadro 30; (c) Rótulos atribuídos no quadro 60.

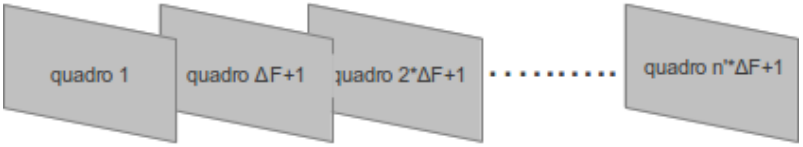
3.1.4 Seleção de quadros

Para seqüências longas de vídeo, é possível diminuir a complexidade do método proposto através da seleção de um subconjunto dos mesmos, escolhidos de forma intercalada. A redução do número de quadros processados irá afetar o tempo total de processamento de duas formas: reduzindo o número de vezes que o cálculo de *mean shift clustering* é realizado e reduzindo a complexidade do meta-agrupamento devido à redução do volume de dados de entrada.

Devido à forma de detecção apresentada na seção anterior, os objetos são túneis espaço-temporais que tendem a persistir em todos os quadros onde exista a parte móvel da cena que eles representam. Essa característica permite que a seleção de quadros seja realizada sem afetar a qualidade do resultado final. A seleção de quadros pode ser vista como uma representação *fast-forward* do vídeo de entrada. A Figura 22 mostra como a seleção de quadros é feita, através da escolha de um quadro a cada intervalo fixo ΔF.



(a)



(b)

Figura 22: Seleção de quadros. (a) Conjunto original de quadros; (b) subconjunto restante após a seleção.

3.1.5 Criação da matriz de dados

Para a execução do agrupamento, os objetos detectados são representados em uma matriz de dados. Essa matriz tem tamanho $[N_{obj}, N_{quadros}, 2]$. Na primeira dimensão são armazenados os rótulos dos objetos, na segunda o índice de cada quadro em que o objeto está presente e na última a posição horizontal e vertical do centroide do objeto no quadro. O centroide em um dado quadro é calculado como a média da posição da cada pixel pertencente ao objeto naquele quadro. A Figura 23 apresenta um exemplo de três objetos detectados e sua movimentação em dois quadros. A Tabela 1 apresenta a matriz de dados obtida para a Figura 23.

3.2 Agrupamento

Para agrupar os objetos que se movem de forma semelhante, é utilizado um esquema de meta-agrupamento (STREHL; GHOSH, 2003). Meta-agrupamento se trata de inicialmente obter partições diferentes para o mesmo conjunto de dados de entrada, e a seguir

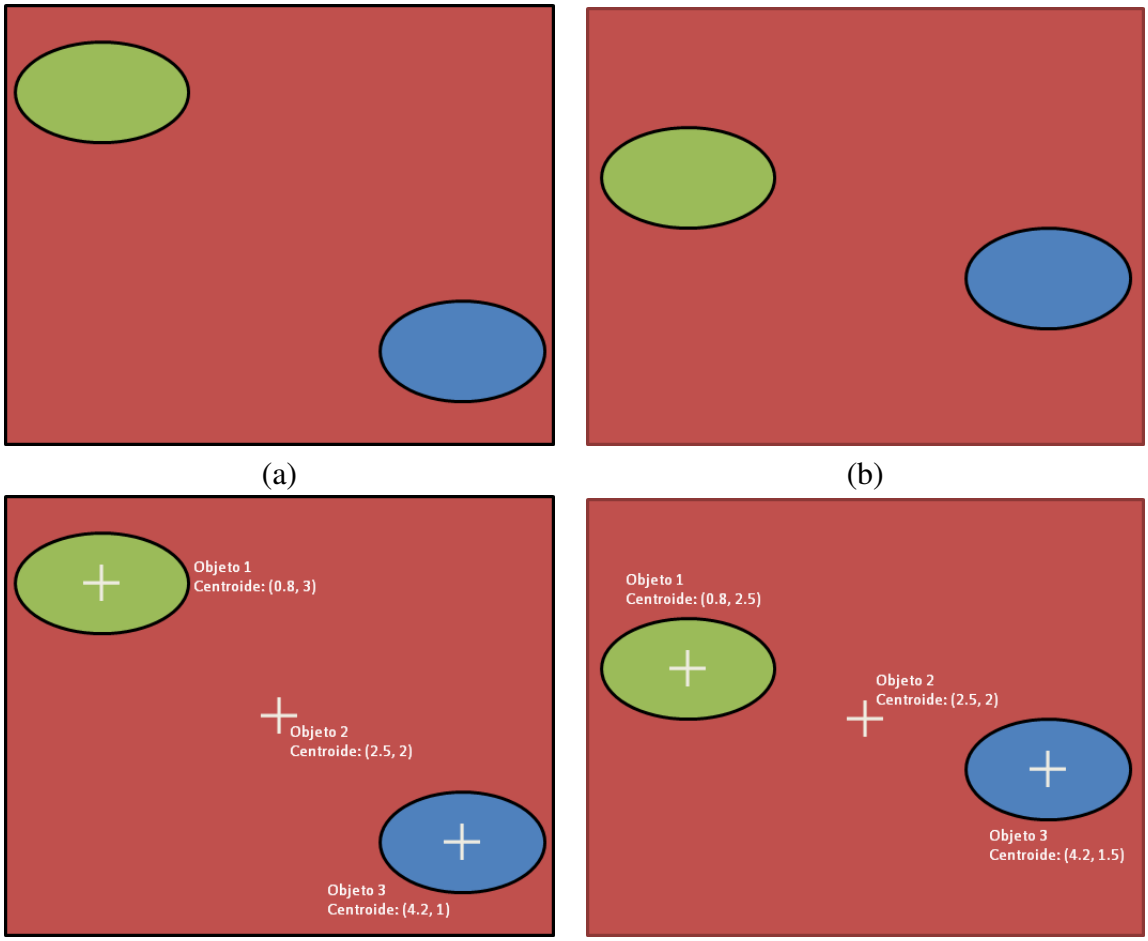


Figura 23: Exemplo de três objetos detectados em dois quadros. (a) Objetos no primeiro quadro; (b) Objetos no segundo quadro; (c) Centroides marcados no primeiro quadro; (d) Centroides marcados no segundo quadro.

Tabela 1: Matriz de dados obtida para a Figura 23

Objeto	Quadro	Posição horizontal do centroide	Posição vertical do centroide
1	1	0.8	3
1	2	0.8	2.5
2	1	2.5	2
2	2	2.5	2
3	1	4.2	1
3	2	4.2	1.5

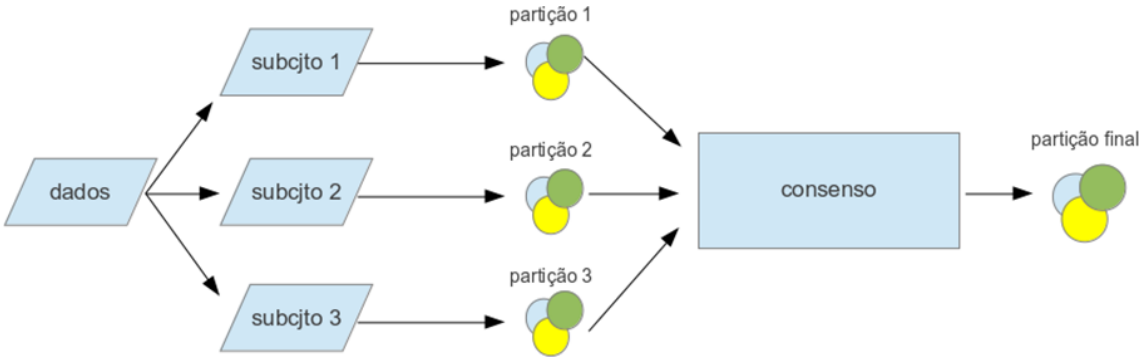


Figura 24: Esquema geral do meta-agrupamento

combiná-las em uma partição que represente um consenso do resultado inicial. A Figura 24 apresenta um esquema geral do meta-agrupamento.

Duas opções são disponíveis para obtenção das diferentes partições iniciais:

- Aplicar diferentes algoritmos de particionamento para o mesmo conjunto de dados de entrada.
- Dividir o conjunto de entrada em subconjuntos, e aplicar a eles o mesmo algoritmo de particionamento.

No segundo caso, os subconjuntos representam visões dos dados de entrada sob diferentes perspectivas, e é a opção utilizada neste método.

Os subconjuntos são obtidos da seguinte forma: Dado um tamanho máximo de janela temporal T, iniciando em cada quadro f, pares de quadros da matriz de dados são selecionados. Os pares iniciam nos dados referentes aos quadros f,$f+1$ até f,$f+T$. É calculada uma matriz com a variação do centroide (movimento) de cada objeto presente nos dois quadros do par, $movimento_{centroide}(f,T) = centroide(f+T) - centroide(f)$. A matriz $movimento_{centroide}(f,T)$ representa o subconjunto de dados que será particionado individualmente. A Figura 25 apresenta uma demonstração visual de como os dados que compõem cada subconjunto são selecionados. O algoritmo de particionamento utilizado para obter grupos em cada subconjunto é o *mean shift clustering*, descrito na seção a seguir.

3.2.1 Agrupamento pelo deslocamento da média - Mean shift clustering

Mean shift clustering, é um algoritmo iterativo apresentado por Fukunaga e Hostetler em (FUKUNAGA; HOSTETLER, 1975) que tem por objetivo agrupar pontos distribuídos no espaço de feições. Na implementação utilizada, os parâmetros de entrada são o conjunto de pontos de entrada $Data_{in}$ e um valor de largura de banda b.

Inicialmente, todos os pontos em $Data_{in}$ são marcados como não-precorridos. Um ponto é selecionado aleatoriamente e marcado como pertencente ao grupo 1. Sua posição no espaço de feições é considerada a média inicial do grupo 1.

Em seguida todos os outros pontos do conjunto de entrada que estão a uma distância b da média do grupo 1 são adicionados a ele, e a nova média é calculada. Esse processo é repetido até que a média do grupo se torne estável, isto é, se mova menos que 0.1% da largura de banda. Em seguida é escolhido de forma aleatória um novo ponto de entrada entre os marcados como não percorridos. Esse novo ponto é marcado como grupo 2. São repetidos os mesmos passos para obtenção dos componentes do grupo 2. Esse sistema se

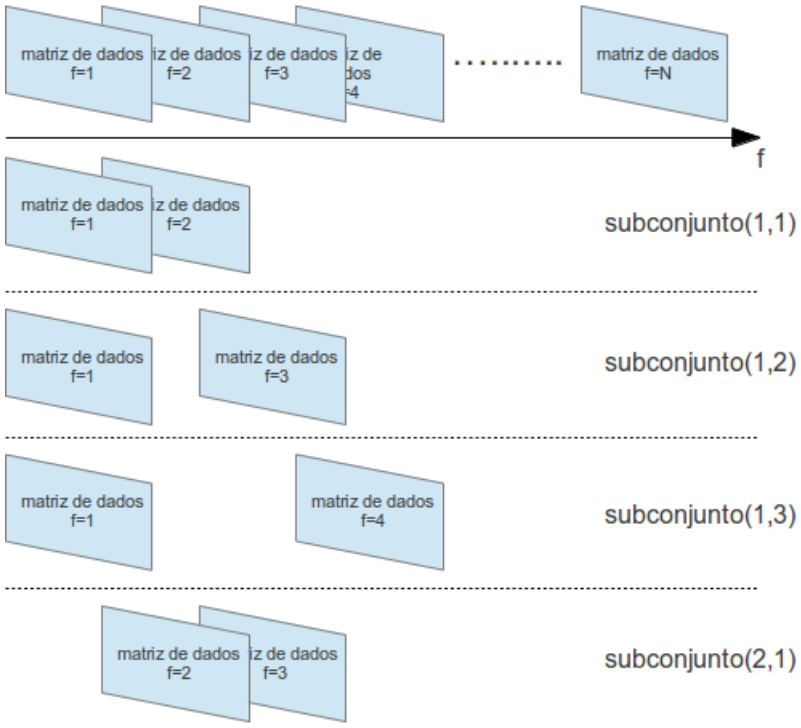


Figura 25: Demonstração da forma como o conjunto de dados de entrada é amostrado para composição dos subconjuntos.

repete até que todos os pontos pertençam a algum grupo. Os passos descritos do *mean shift clustering* são apresentados na Figura 26.

Após obtidas as partições iniciais com o *mean shift clustering*, será feito o cálculo do consenso entre as mesmas, utilizando meta-agrupamento.

3.2.2 Meta-agrupamento

No trabalho intitulado *Cluster Ensembles - A Knowledge Reuse Framework for Combining Multiple Partitions* (STREHL; GHOSH, 2003) são apresentadas três possíveis estratégias para resolver agrupamentos de *clusters*:

- *Cluster-based Similarity Partitioning Algorithm* (*CSPA*): Baseado na medição de similaridade aplicada a pares.
- *HyperGraph Partitioning Algorithm* (*HGPA*): Baseado no particionamento de um hiper-grafo com hiper-arestas.
- *Meta-CLustering Algorithm* (*MCLA*): Baseado na consolidação de *clusters* em grupos de *clusters* (meta-grupos).

O método proposto utiliza a estratégia *MCLA* para consolidar as partições iniciais em meta-grupos. O primeiro passo é criar uma matriz C que consolide os resultados de agrupamento obtidos nos subconjuntos. A Tabela 2 mostra um exemplo de matriz C contendo os índices de objetos nas linhas e resultados de subconjuntos λ_n nas colunas. O número de resultados de subconjuntos pode ser calculado pela fórmula $N_{quadros}.T$-$\Sigma_{n=1}^{T} n$=$N_{quadros}.T$-$T.(T+1)/2$, onde $N_{quadros}$ é o número de quadros e T é o tamanho máximo da janela temporal. Em cada resultado de subconjunto é indicado índice do grupo ao qual o objeto pertence.

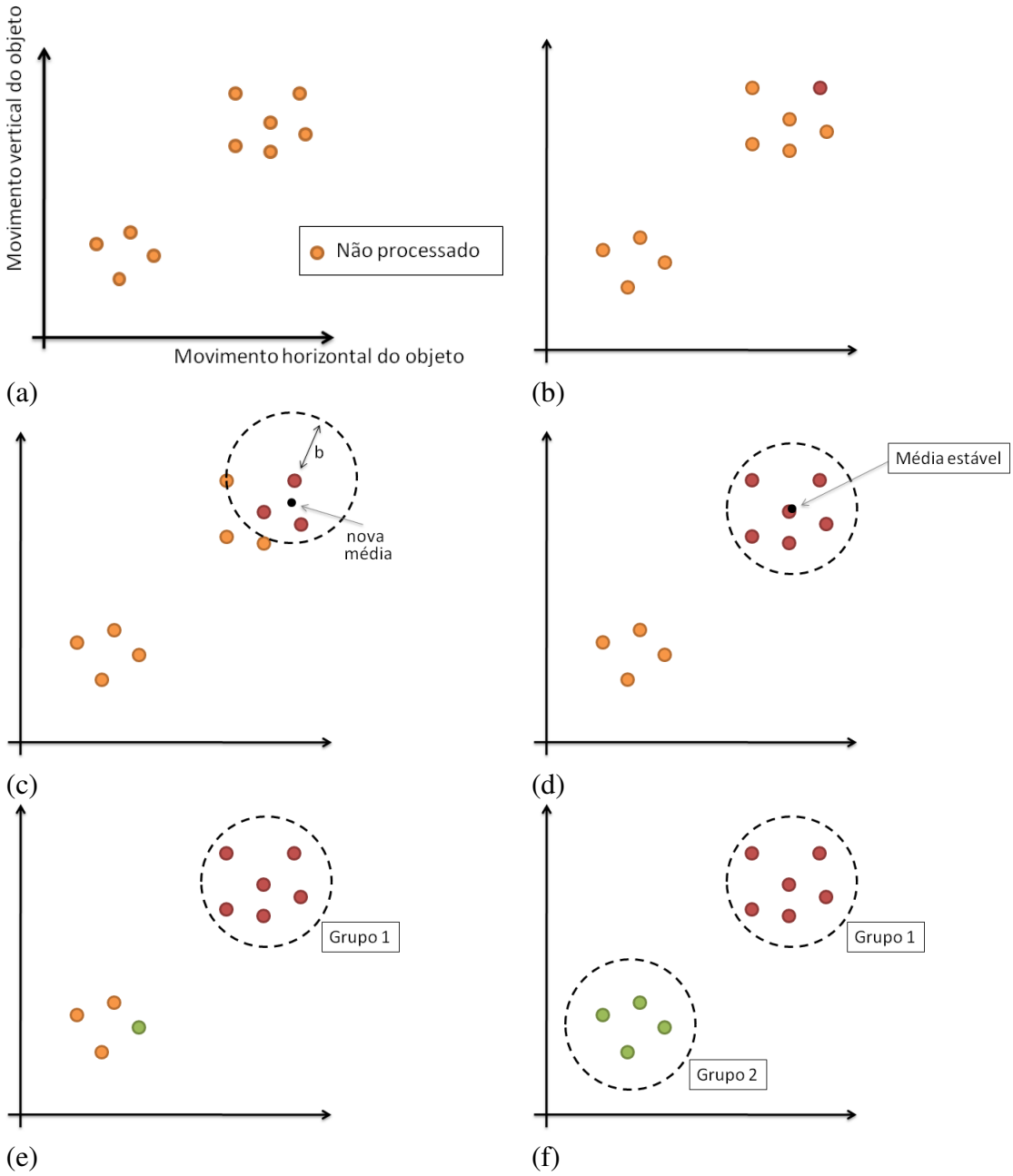


Figura 26: Passos do *mean shift clustering*. (a) Inicialmente todos os pontos são marcados como não-processados. (b) É escolhido um ponto aleatório entre os não-processados. Ele é marcado como um novo grupo e sua posição no espaço de feições é considerada a posição média inicial do grupo. (c) São adicionados aos grupos todos os ponto cuja posição no espaço de feições esteja a uma distância máxima b (largura de banda) da média do grupo. A média é recalculada levando em conta os novos pontos. (d) Esse processo se repete até que a média se torne estável. (e) É escolhido um novo ponto aleatório entre os não-processados, ele recebe um rótulo de novo grupo. (f) Esse processo se repete até que todos os objetos sejam processados.

Tabela 2: Exemplo de matriz C contendo o resultado do agrupamento de sete objetos em três subconjuntos diferentes λ_n

Obj	λ_1	λ_2	λ_3
1	1	2	1
2	1	2	1
3	1	2	2
4	2	3	2
5	2	3	3
6	3	1	3
7	3	1	3

Em seguida a matriz C deve ser convertida para o formato do hiper-grafo H. Essa conversão é feita através da decomposição de cada resultado de subconjunto λ_n em um número de colunas (hiper-arestas) equivalente ao número de grupos contidos no resultado. Cada hiper-aresta possui 1 no índice dos objetos pertencentes ao grupo e 0 nos demais. A Figura 27 apresenta um exemplo de como uma matriz C com resultados de particionamento λ_n de três subconjuntos de dados e sete objetos é convertida para o hiper-grafo H.

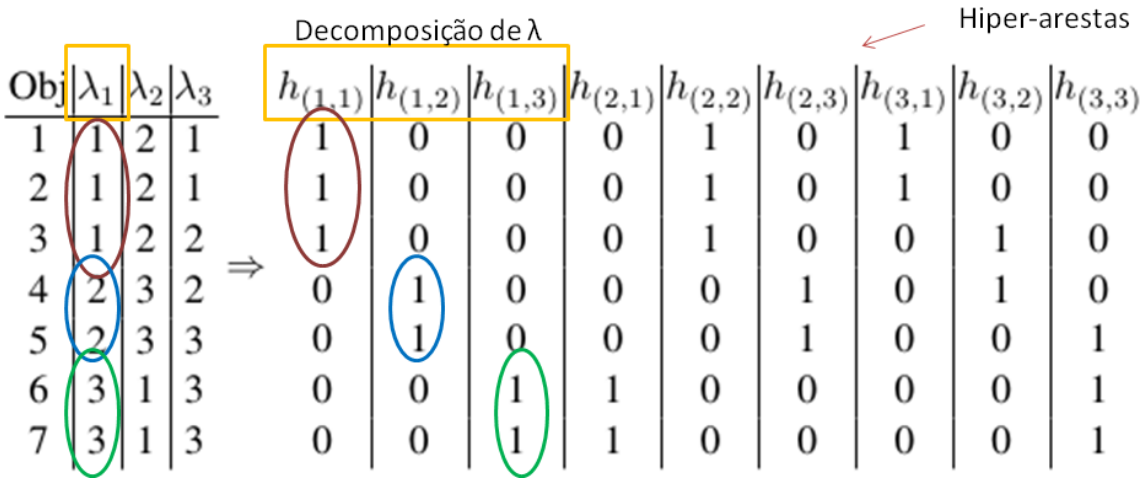


Figura 27: Transformação da matriz C com três subconjuntos, três grupos em cada resultado e sete objetos no hiper-grafo H.

Em seguida é calculada a similaridade entre todas as hiper-arestas no hiper-grafo H. A métrica utilizada é a distância de Jaccard (ALLEN et al., 2009), utilizada como na equação 2, onde h_s e h_t representam qualquer par de hiper-arestas.

$$m(h_s, h_t) = \frac{h'_s.h_t}{||h_s||_2^2 + ||h_t||_2^2 - h'_s.h_t} \tag{2}$$

Após o cálculo da similaridade entre todas as hiper-arestas, é possível construir uma árvore hierárquica de grupos Z. A árvore inicia com o primeiro nível Z_1 contendo todos as hiper-arestas. A cada nível Z_s, o par de nodos com a menor distância entre si são unidos e formam um novo nodo. Esse processo é repetido até que se chegue no nível

onde somente um nodo restar. A cada união de nodos a distância utilizada é tomada e armazenada para o passo seguinte.

Após a união de todos os nodos, é identificado o nível Z_D, onde ocorreu a união com a maior distância. Esse nível é considerado o primeiro onde ocorreu união entre diferentes meta-grupos. Desse nível em diante, considera-se que todas as uniões ocorreram entre meta-grupos. Dito isso, pode-se afirmar que o número de níveis entre Z_D e o último nível da árvore é diretamente relacionado ao número de meta-grupos $Ngrupos$, sendo igual a $Ngrupos - 1$. A Figura 28 apresenta um representação gráfica do processo da criação da árvore até o indicativo do número de meta-grupos. Para evidenciar a relação entre o nível onde ocorre a fusão com maior distância e o número de meta-grupos, é demonstrado na Figura N um exemplo onde as hiper-arestas possuem uma distância uniforme entre si. Uma vez que a maior distância ocorre na primeira fusão, o número de meta-grupos detectados é equivalente ao número de hiper-arestas.

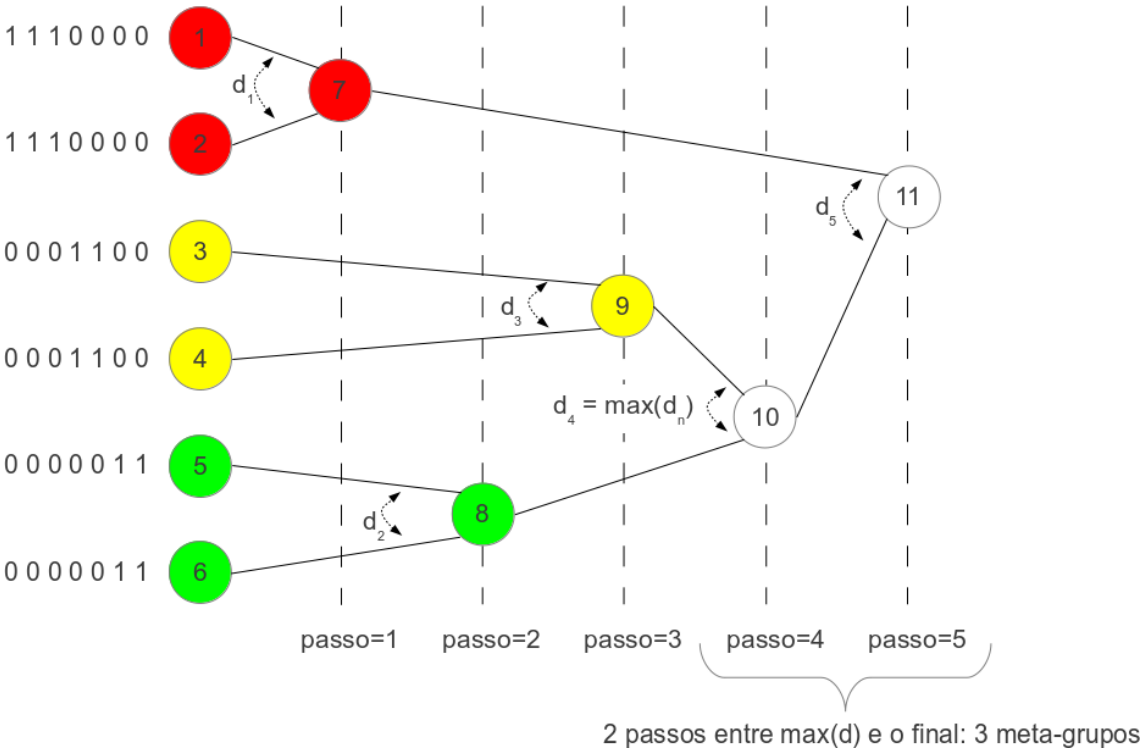


Figura 28: Demonstração da criação da árvore hierárquica Z a partir da união das hiper-arestas. A maior distância $max(d)$ é considerada o limiar para determinar o número de meta-grupos.

Após a definição de N_{grupos}, a árvore Z é cortada no primeiro nível onde existam N_{grupos} meta-grupos ou menos. Os meta-grupos obtidos representam o consenso entre as partições obtidas na etapa de *mean shift clustering*.

Finalmente, é calculada a matriz de probabilidades $P(objeto, grupo)$, contendo a chance de cada objeto pertencer a cada um dos meta-grupos identificados. Os dados de probabilidade vêm fato que um mesmo objeto pode pertencer a meta-grupos diferentes em diferentes hiper-arestas. Um exemplo de matriz de probabilidades é apresentado na Tabela 3.

O resultado final do meta-agrupamento é o número de meta-grupos e um rótulo para cada objeto que indica a qual deles o mesmo pertence. Esse resultado será refinado a seguir na fase de pós-processamento para a obtenção do vídeo segmentado.

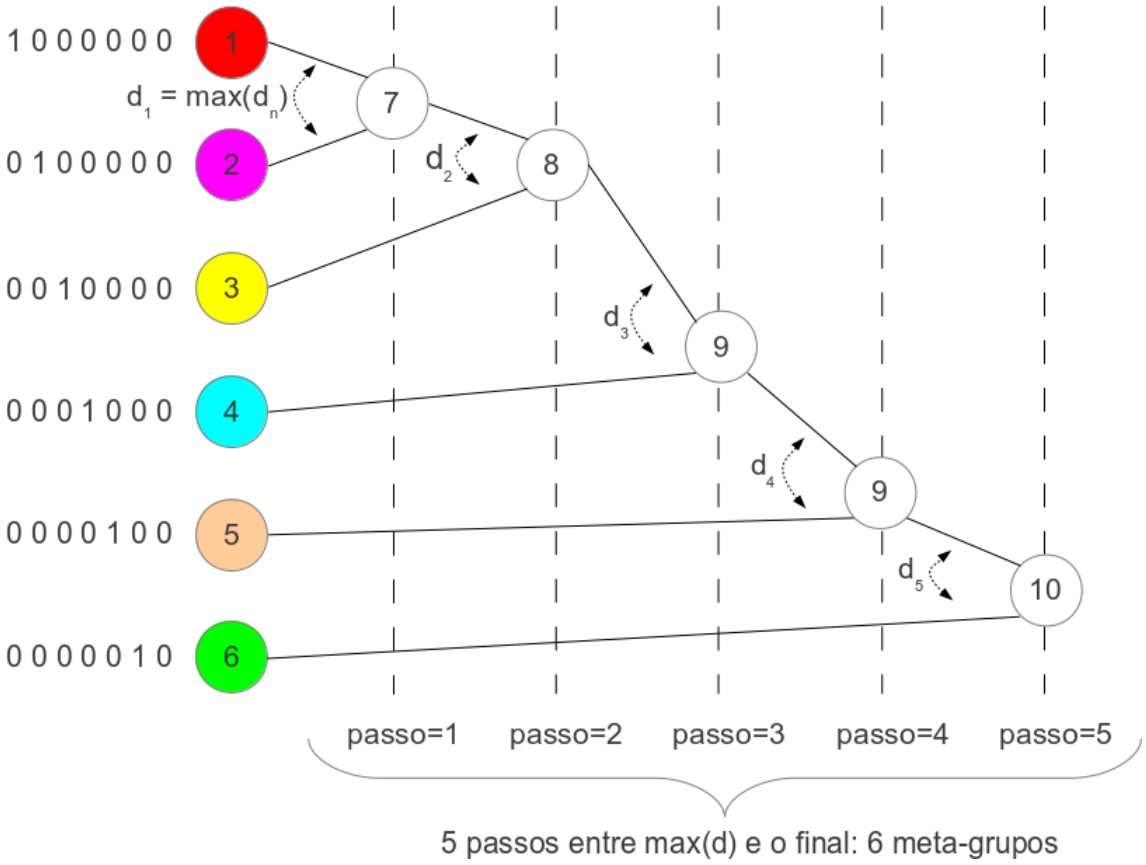


Figura 29: Outro exemplo de árvore hierárquica, visando evidenciar a relação entre o passo onde ocorre a maior distância de fusão max(d) e o número de meta-grupos. Uma vez que os grupo possuem distância uniforme entre si a maior distância ocorre no primeiro nível, resultado em um número de meta-grupos igual ao número de hiper-arestas.

Tabela 3: Exemplo de matriz de probabilidades

Objeto	Grupo 1	Grupo 2	Grupo 3
1	0.4646	0.0054	0.0035
2	0	0.0042	0
3	0.4618	0.0096	0.0017
4	0.0058	0.5400	0
5	0.1854	0.0004	0
6	0.4509	0	0.0043
7	0.0011	0	0

3.3 Pós-processamento

A fase de pós-processamento é composta por duas etapas: refinamento de grupos e conversão dos mesmos para imagens convexas.

3.3.1 Refinamento de grupos

O uso de movimento para segmentação de vídeo traz desvantagens em cenas onde partes diferentes realizam movimentos similares, por exemplo: carros se movendo em uma estrada. Nesse tipo de cenário, diferentes carros que realizam o mesmo movimento podem ser detectados como pertencentes ao mesmo grupo, embora eles sejam espacialmente distantes.

Outra possibilidade de erro de detecção é quando o vídeo possui movimentos esparsos (por exemplo o vídeo passa um longo tempo sem movimento). Nesse caso, mesmo que o algoritmo de *mean shift clustering* segmente corretamente as partes em movimento em diferentes pontos do vídeo, o meta-agrupamento irá considerar os movimentos esparsos como o mesmo grupo.

Dessa forma observa-se que a maioria dos vídeos de situações reais requer refinamento do resultado do agrupamento. O refinamento consiste nos seguintes passos: remoção do grupo de fundo, remoção de *outliers*, aplicação de condição temporal e finalmente aplicação de restrição espacial.

3.3.1.1 Remoção do grupo de fundo

No primeiro passo, remoção de grupo de fundo, determina-se qual grupo representa o plano de fundo e seus objetos são removidos do resultado atual. Isso é feito a partir da premissa que o grupo que possuir mais posições é o plano de fundo. Um exemplo de remoção do grupo de fundo é apresentado na Figura 30.

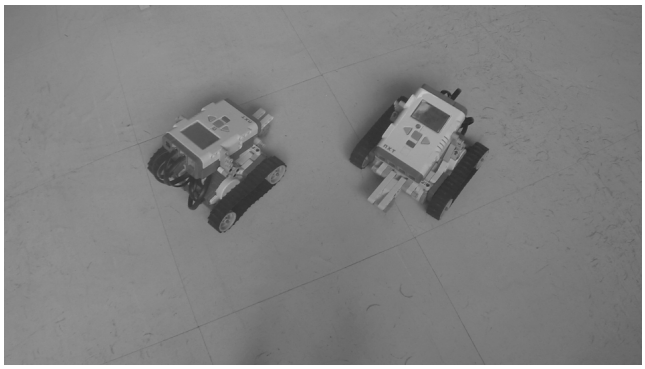
3.3.1.2 Remoção de outliers

Ocasionalmente no resultado da aplicação do algoritmo observou-se que algum objeto que deveria pertencer ao plano de fundo pode ser detectado como movimento e ser agrupado junto às partes da cena que realmente se movem. Observou-se que quando isso ocorre trata-se de objetos que existem em poucos quadros, geralmente dois ou menos, e que podem estar em qualquer posição do vídeo.

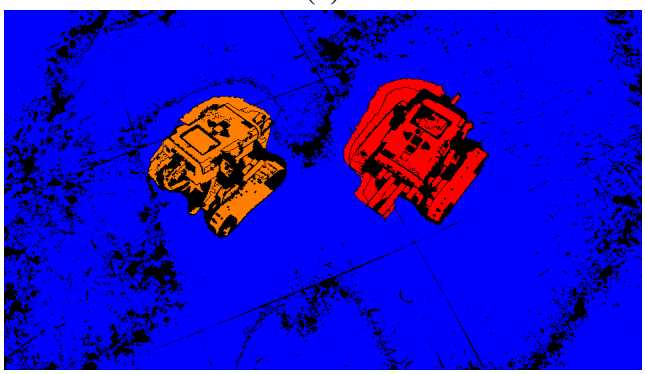
Uma forma simples de remover esses *outliers* é removendo dos grupos todos os objetos que existam em menos de três quadros. Isso traz pouco prejuízo ao resultado da segmentação devido ao fato de que a maioria dos objetos que compõem as partes móveis da cena existe por uma quantidade maior de quadros. A Figura 31 mostra um exemplo de *outlier* existente em uma sequencia de quadros e o resultado da remoção do mesmo.

3.3.1.3 Aplicação de restrição temporal

Em vídeo onde os movimentos são realmente esporádicos (apenas uma parte móvel aparece por vez) a aplicação de restrição temporal resolve o refinamento dos grupos. Essa condição se trata do fato que, se um grupo desaparece e reaparece no vídeo, ele deve ser dividido em dois grupos diferentes. Um efeito colateral desta restrição é que partes móveis que se tornem totalmente oclusas são detectadas como dois grupos distintos.



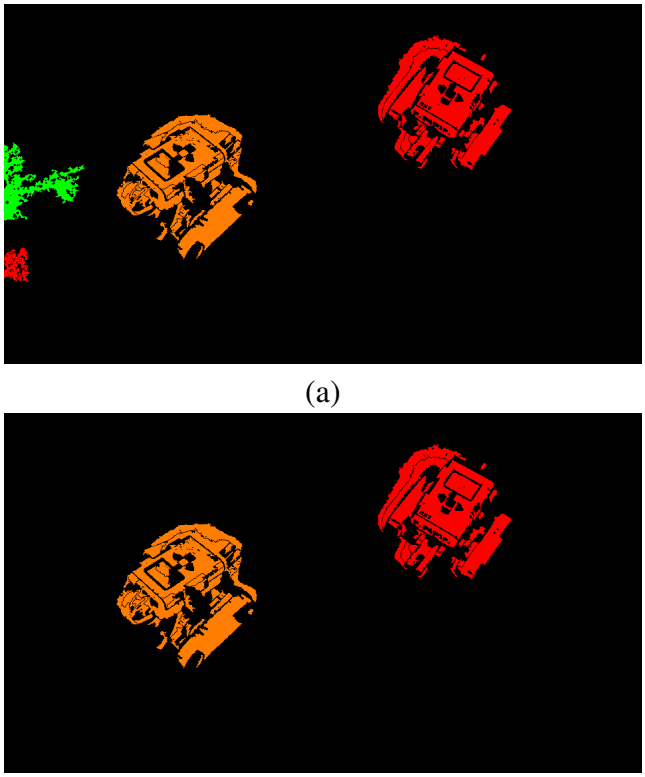
(a)



(b)

(c)

Figura 30: Exemplo de remoção do grupo que contém o plano de fundo. (a) Quadro original; (b) Grupos detectados; (c) Grupos sem o fundo;



(a)

(b)

Figura 31: Exemplo de remoção de *outlier* em sequencia de vídeo. (a) Quadro apresentando *outliers*; (b) mesmo quadro após a remoção.

3.3.1.4 Aplicação de restrição espacial

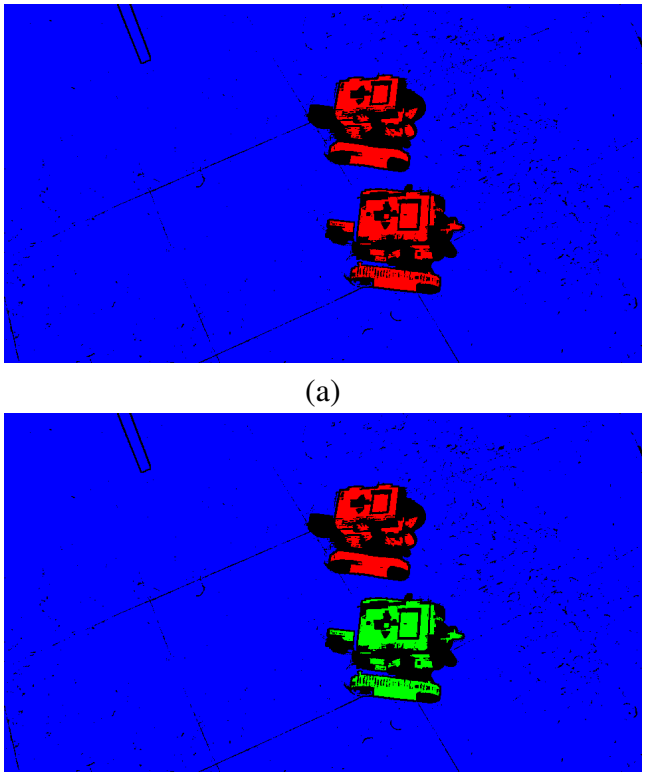
Em vídeos com movimento esporádico é possível que várias partes móveis diferentes sejam detectados como pertencendo ao mesmo objeto. Caso apenas uma parte móvel esteja presente na cena por vez, a aplicação de restrição temporal descrita na seção anterior é suficiente para que os grupos sejam detectados de forma correta. Contudo, se mais de uma parte móvel está presente é necessário aplicar uma restrição espacial.

Para tanto define-se um limiar de distância entre os centróides de objetos que pertencem ao mesmo grupo. Quando dois objetos apresentam uma distância maior que o limiar estabelecido, um deles recebe um rótulo de novo grupo. Em seguida os objetos próximos ao novo grupo são adicionados ao mesmo.

A Figura 32 mostra o resultado da aplicação de restrição temporal em um vídeo.

3.3.2 Conversão em imagens convexas

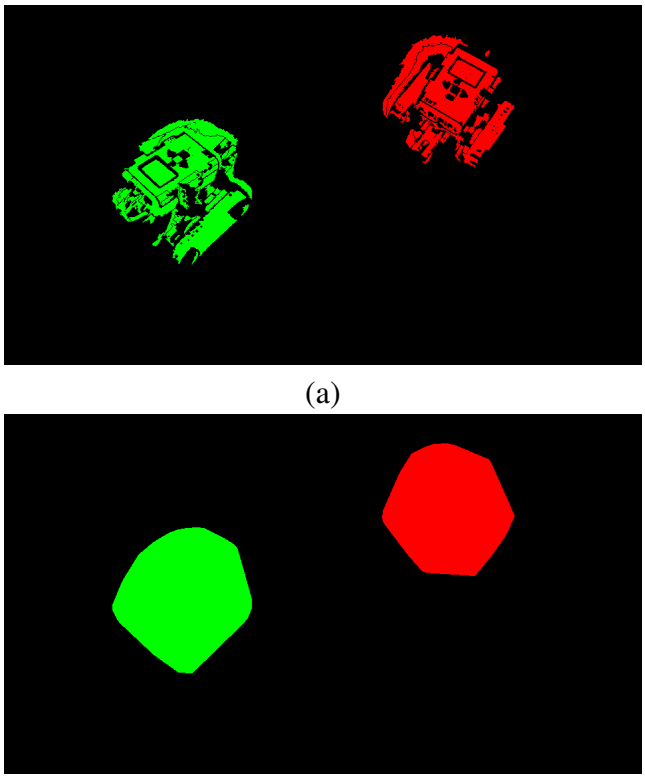
Uma vez que o método apresentado é baseado em túneis espaço-temporais, os grupos detectados são compostos de objetos desconexos entre si. Para que seja criado um túnel único para cada grupo, é calculada para cada grupo, em cada quadro, a imagem convexa do conjunto dos seus objetos. Um exemplo de conversão em imagens convexas é apresentado na Figura 33.



(a)

(b)

Figura 32: Exemplo de aplicação de restrição espacial para refinamento dos grupos. (a) Resultado antes da aplicação da restrição; (b) resultado após a aplicação da restrição.



(a)

(b)

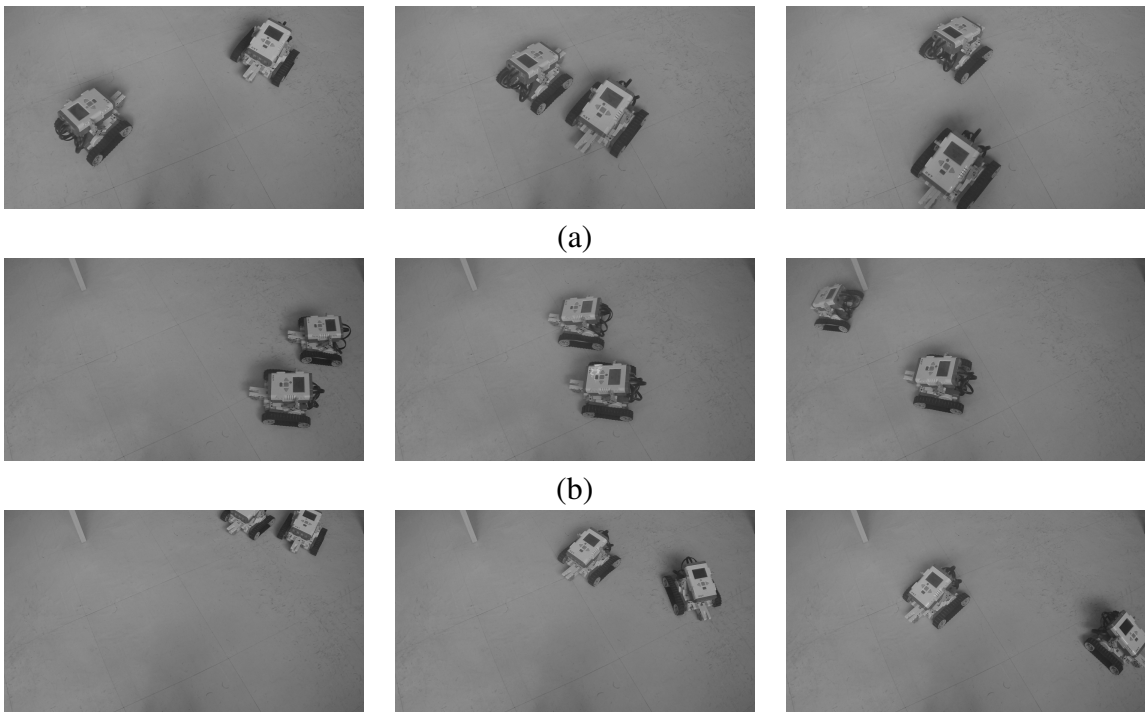
Figura 33: Exemplo de conversão dos objetos do grupo em uma única imagem convexa. (a) Grupos detectados; (b) Grupos convertidos em imagens convexas.

4 RESULTADOS EXPERIMENTAIS

Neste capítulo são apresentados os resultados da aplicação do algoritmo com vídeos gravados em ambiente controlado, uma comparação dos mesmos com o trabalho de (SILVA; SCHARCANSKI, 2010), e o resultado da aplicação do algoritmo em vídeos de dois conjuntos públicos de dados.

4.1 Resultados obtidos em ambiente controlado

Para avaliação e comparação do desempenho do algoritmo, diversos vídeos foram gravados em ambiente controlado. Neste cenário foi utilizada câmera e plano de fundo fixos e foram filmados dois carros robóticos programados para realizar movimentos pré-determinados. Alguns quadros de cada vídeo são mostrados na Figura 34. A Tabela 4 apresenta uma descrição dos vídeos utilizados.



(a) (b) (c)

Figura 34: Quadros dos vídeos utilizados nos testes. (a) Vídeo 1. (b) Vídeo 2. (c) Vídeo 3.

Tabela 4: Descrição dos vídeos de teste

Vídeo	Descrição	Número de quadros
1	Dois carrinhos se movendo na mesma direção, em sentidos opostos	136
2	Dois carrinhos se movendo na mesma direção e sentido, com diferente aceleração	363
3	Dois carrinhos que iniciam se movendo na mesma direção e sentido e a seguir adotam trajetórias diferentes	336

Tabela 5: Comparação de taxa de acerto entre (SILVA; SCHARCANSKI, 2010) e o método proposto

Índice do vídeo	(SILVA; SCHARCANSKI, 2010)	Método proposto
1	96.79%	96.21%
2	86.23%	98.71%
3	97.64%	97.70%

Em todos os vídeos obtidos em ambiente controlado o algoritmo proposto foi capaz de segmentar os carros presentes. A Figura 35 mostra exemplos de resultado da segmentação.

4.2 Comparação

Para realizar a comparação entre o método proposto e o de (SILVA; SCHARCANSKI, 2010), foi obtido o *ground truth* denso dos três dos vídeos apresentados na seção anterior. A Figura 36 apresenta quadros do *ground truth* e dos resultados dos métodos comparados.

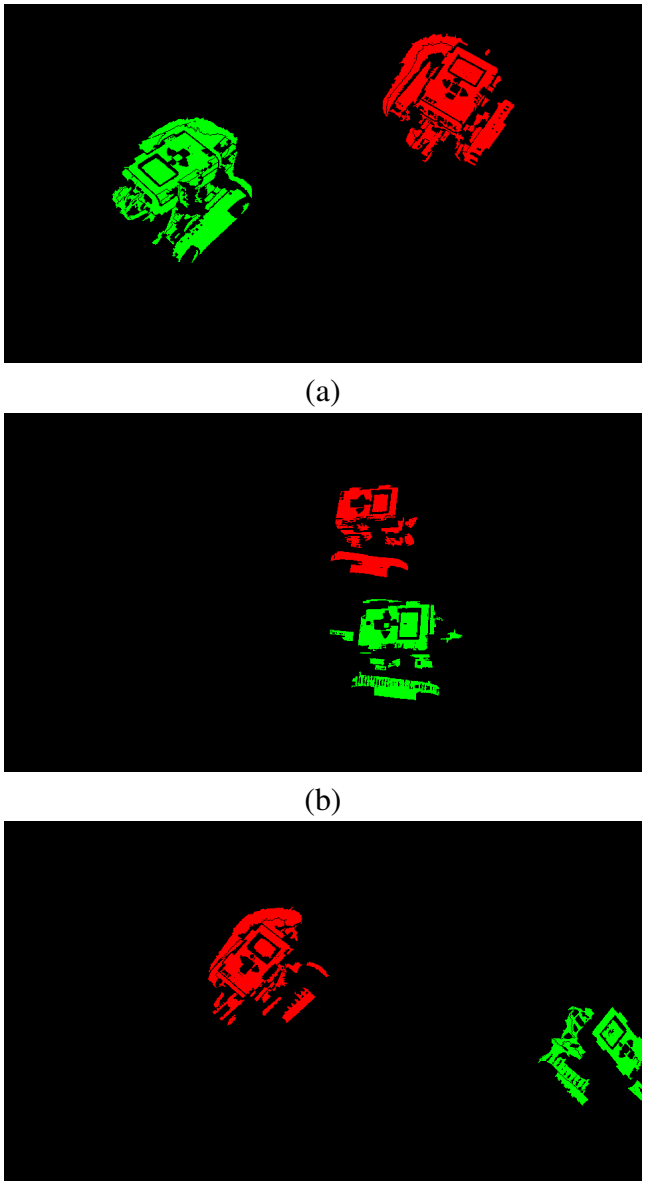
Para comparação de acurácia foi tomada a taxa de acerto como a divisão do número de pixels identificados corretamente pelo total de pixels do vídeo. Os resultados são apresentados na Tabela 6. O método proposto obteve uma taxa de acerto equivalente à de (SILVA; SCHARCANSKI, 2010).

O equipamento de teste utilizado para a execução do método proposto foi um PC com processador Intel Core i7 e 8GB de memória RAM. O código do método foi implementado com código do MATLAB não otimizado. A Tabela 7 apresenta informações e o tempo de execução de alguns dos vídeos testados.

O equipamento de teste utilizado para execução do método de (SILVA; SCHARCANSKI, 2010) foi um PC com processador Intel Core i5 e 8GB de memória RAM. Os tempos de execução são apresentados na Tabela 8. Em média, o tempo de execução do método proposto foi 1% do tempo de (SILVA; SCHARCANSKI, 2010).

	(SILVA; SCHARCANSKI, 2010)		Método proposto	
Vídeo	Tempo	Taxa de acerto	Tempo	Taxa de acerto
1	15h 36min	96.79%	865s	96.21%
2	39h 12min	86.23%	1440s	98.71%
3	38h 30min	97.64%	1566s	97.70%

Tabela 6: Comparação entre o método de (SILVA; SCHARCANSKI, 2010) e o método proposto (tempo de processamento e taxa de acerto).



(a)

(b)

(c)

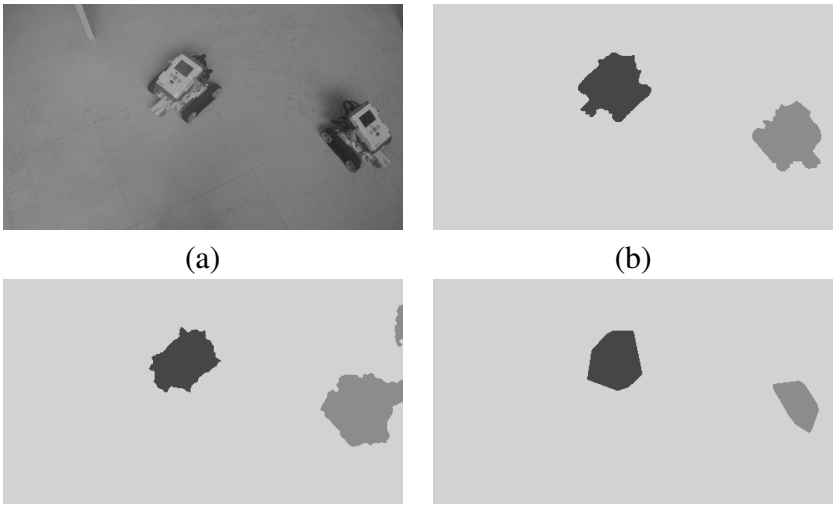
Figura 35: Exemplos de resultados de segmentação. (a) Vídeo 1. (b) Vídeo 2. (c) Vídeo 3.

Tabela 7: Tempo de execução das amostras

Video	Tamanho (pixels)	Quadros	Tempo de execução
1	540x960	136	865s
2	540x960	363	1440s
3	540x960	336	1566s

Tabela 8: Comparação de tempo de execução entre (SILVA; SCHARCANSKI, 2010) e o método proposto.

Video	(SILVA; SCHARCANSKI, 2010)	Método proposto
1	15h 36min	865s
2	39h 12min	1440s
3	38h 30min	1566s



(a) (b)

(c) (d)

Figura 36: Comparação entre resultados de segmentação. (a) Quadro do vídeo original; (b) *Ground truth*; (c) (SILVA; SCHARCANSKI, 2010); (d) Método proposto

Tabela 9: Comparação do tempo de processamento do vídeo 1 utilizando diferentes tamanhos de janelas temporais.

Número de quadros da janela temporal	10	11	12
Tempo de pré-processamento	612s	612s	612s
Tempo de agrupamento e pós-processamento	252s	359s	672s
Tempo total	864s	971s	1284s

4.3 Variação de parâmetros

A fim de observar o impacto que os parâmetros exercem sobre o tempo de processamento foram realizados testes com vídeos obtidos em ambiente controlado variando o divisor da seleção de quadros e o tamanho da janela temporal.

A Tabela 9 apresenta o resultado do tempo de execução para o vídeo 1 utilizando diferentes janelas temporais. Já a tabela 10 apresenta o resultado do tempo de execução para o vídeo 3 utilizando diferentes divisores para a seleção de quadros. Em ambos os casos é apresentado o tempo de pré-processamento separado do tempo de agrupamento e pós-processamento, devido ao fato do primeiro se manter constante e representar uma porção significativa do tempo total. Em ambos os casos o primeiro valor apresentado é o menor valor de parâmetro para o qual foi possível obter um resultado de segmentação satisfatório.

Tabela 10: Comparação do tempo de processamento do vídeo 3 utilizando diferentes divisores na seleção de quadros.

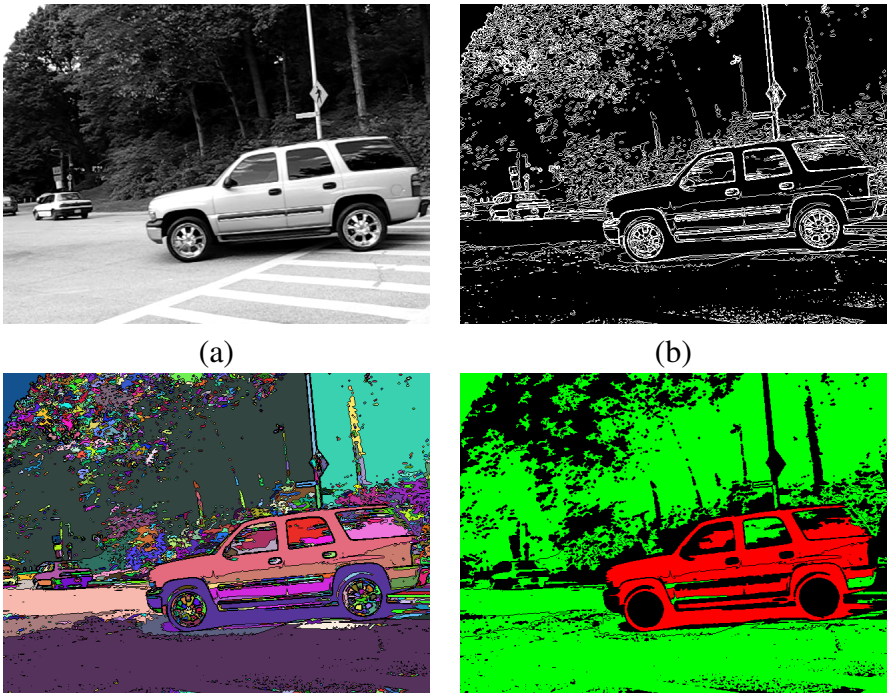
Fator de seleção de quadros	4	3	2	1
Tempo de pré-processamento	1398	1398	1398	1398
Tempo de agrupamento e pós-processamento	123	213	621	2300
Tempo total	1521	1611	2019	3698

4.4 Vídeos da base pública de dados Hopkins155

O algoritmo proposto também foi testado com dois vídeos da base pública de dados *Hopkins155* (TRON; VIDAL, 2007), amplamente utilizada em publicações para comparação de resultados de segmentação.

O vídeo chamado *cars1* apresenta uma cena dinâmica, com câmera móvel e movimento de um carro.

A Figura 37 mostra um quadro do vídeo original, o resultado da extração de bordas com linhas de nível, os objetos detectados e o resultado da segmentação antes da conversão em imagens convexas:



(a) (b)

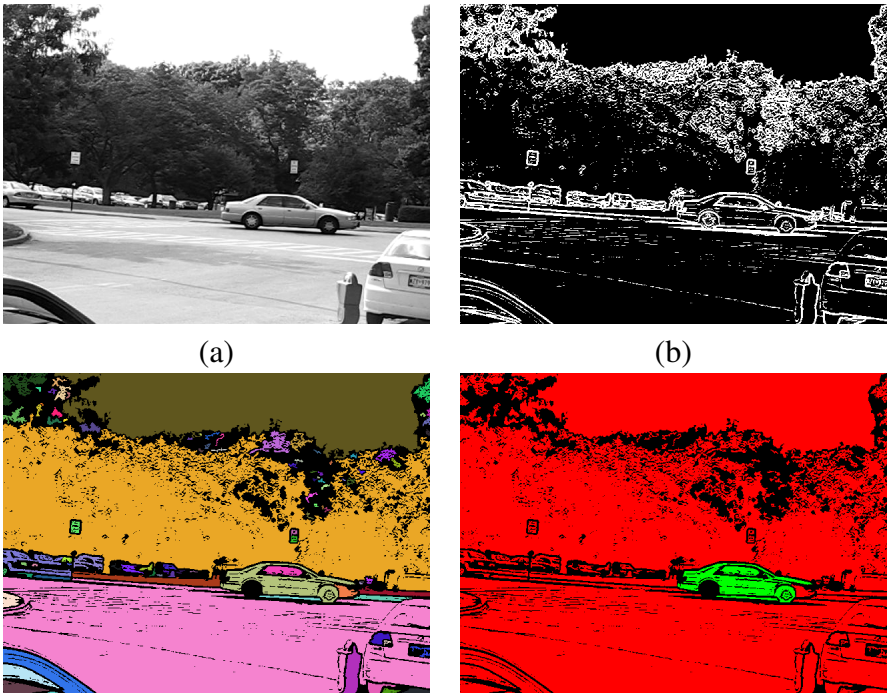
(c) (d)

Figura 37: Resultado da aplicação do algoritmo proposto no vídeo *cars1* da base pública de dados *Hopkins155*. (a) Quadro do vídeo original; (b) resultado da extração de bordas; (c) objetos detectados; (d) resultado da segmentação

Já o vídeo entitulado *cars6* apresenta uma cena semelhante ao *cars1*, porém a câmera se move pouco em comparação com o movimento realizado pelo objeto.

A Figura 38 mostra um quadro do vídeo original, o resultado da extração de bordas com linhas de nível, os objetos detectados e o resultado da segmentação antes da conversão em imagens convexas:

O vídeo possui 20 quadros, tamanho 640x480 pixels. O tempo de pré-processamento foi de 63 segundos e o tempo total de execução do método foi de 73 segundos. Embora a base pública *Hopkins155* forneça *ground truth* de segmentação para seus vídeos, o resultado fornecida é uma segmentação esparsa utilizando partículas, não sendo comparável com o resultado de segmentação densa obtido por este trabalho.



(a) (b)

(c) (d)

Figura 38: Resultado da aplicação do algoritmo proposto no vídeo *cars6* da base pública de dados *Hopkins155*. (a) Quadro do vídeo original; (b) resultado da extração de bordas; (c) objetos detectados; (d) resultado da segmentação

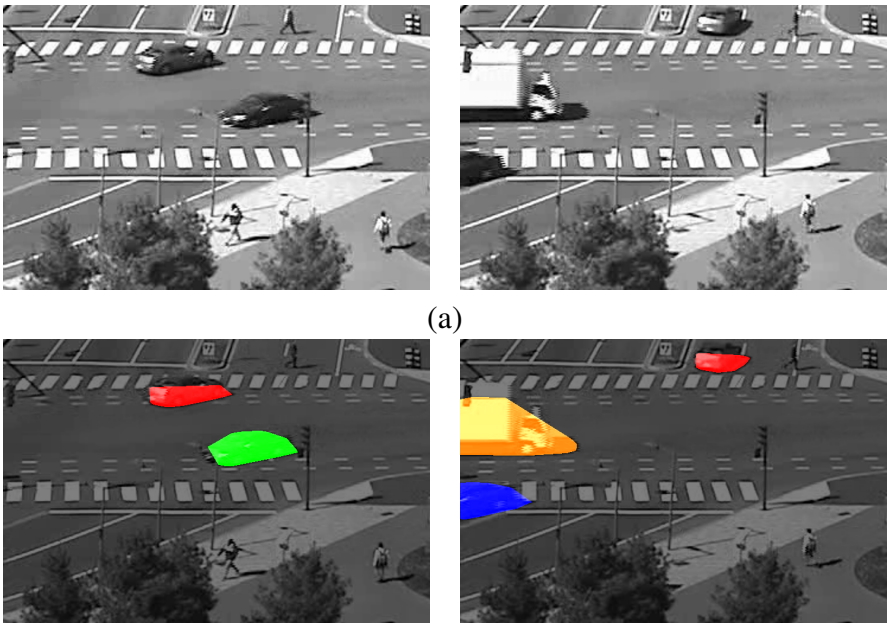
4.5 Vídeo da base pública de dados MIT traffic data set

Para avaliar o desempenho do algoritmo para segmentar diferentes objetos em um cenário de câmera fixa, foi utilizado um vídeo da base pública de dados *MIT traffic data set* (WANG; MA; GRIMSON, 2009).

O vídeo *mv2_001* mostra o tráfego de carros nos arredores de um cruzamento, gravado de uma câmera fixa posicionada em um ponto alto. Para o experimento foi isolada uma área menor do vídeo, centralizada no cruzamento.

A Figura 39 mostra dois quadros do vídeo original e os mesmos no resultado da segmentação após a conversão em imagens convexas:

O vídeo possui 230 quadros, tamanho 720x480 pixels. O tempo de pré-processamento foi de 681 segundos e o tempo total de execução do método foi de 896 segundos.



(a)

(b)

Figura 39: Resultado da aplicação do algoritmo proposto no vídeo *mv2_001* da base pública de dados *MIT traffic data set*. (a) Quadros do vídeo original; (b) resultado da segmentação

5 CONCLUSÕES

Este trabalho apresentou um algoritmo para segmentação de vídeo baseado na análise do movimento de partes móveis de um vídeo. O objetivo era aprimorar o método de segmentação com base em movimento apresentado por (SILVA; SCHARCANSKI, 2010), obtendo resultados de qualidade semelhante porém exigindo um tempo de processamento reduzido. A principal diferença é o uso da detecção de túneis tridimensionais detectados no domínio espaço-temporal ao invés de partículas. O uso de túneis traz duas vantagens: menor quantidade de itens a serem agrupados por vídeo e possibilidade de selecionar os dados de poucos quadros sem prejuízo à segmentação, uma vez que o algoritmo utilizado para detecção dos mesmos faz com que eles sejam consistentes ao longo de muitos quadros.

O novo algoritmo foi divido em três etapas: pré-processamento, agrupamento e pós-processamento. A etapa de agrupamento é derivada diretamente de (SILVA; SCHARCANSKI, 2010), enquanto o pré-processamento e pós-processamento são contribuições originais. Para a etapa de pré-processamento foi implementado um algoritmo de linhas de nível descrito em (NEGRI; LOTITO, 2012) e foi criado um esquema de detecção de objetos que evita a propagação de erros através da restrição do número de objetos com o mesmo rótulo no mesmo quadro. Já para a etapa de pós-processamento foram aplicadas restrições temporais e espaciais que permitem que o resultado seja refinado caso objetos pertencentes à partes em movimento diferentes sejam agrupados juntos.

O método proposto foi testado em vídeos gravados em ambiente controlado, com fundo e câmera fixos. Para esses vídeos foi obtido *Ground Truth*, possibilitando uma análise quantitativa. Também foram feitos testes em vídeos das bases públicas de dados (vídeos *cars1* e *cars6* da base *hopkins155* (TRON; VIDAL, 2007) e vídeo *mv2_001* da base *MIT traffic data set* (WANG; MA; GRIMSON, 2009)).

A contribuição do trabalho proposto sobre o algoritmo de (SILVA; SCHARCANSKI, 2010) foi medida quantitativamente nos vídeos obtidos em ambiente controlado. Neles o método proposto apresentou acurácia equivalente ao método de (SILVA; SCHARCANSKI, 2010), enquanto o tempo de processamento foi reduzido a 1% do tempo original. Essa melhora foi obtida devido ao fato que a detecção de objetos através de túneis espaço-temporais permite que poucos objetos e quadros sejam selecionados, ainda garantindo dados significativos para o agrupamento. Embora tenham sido utilizados processados diferentes (Intel Core i5 e i7) para os testes entre o método proposto e o de (SILVA; SCHARCANSKI, 2010), considera-se que isso não traz grande impacto em comparação com a grande diferença no tempo de processamento observado.

Em uma análise visual, também foram obtidos resultados satisfatórios nos vídeos de bases públicas de dados. A base pública *hopkins155* fornece *ground truth* baseado em partículas, enquanto a base *MIT traffic data set* não fornece, de forma que não foi pos-

sível fazer uma comparação quantitativa. A comparação do tempo de execução do método proposto com o tempo indicado no texto de outros métodos de segmentação baseada em movimento ((MANSOURI; KONRAD, 2003), (SUNDARAM; KEUTZER, 2011), (DIMITRIOU; DELOPOULOS, 2013), (PALOU; SALEMBIER, 2013)) sugere que o primeiro é capaz de realizar a tarefa de segmentação de forma mais eficiente. A comparação com o tempo sugerido em (BELAGIANNIS et al., 2012) não é realizada pois o mesmo processa os quadros do vídeo somente no entorno do objeto monitorado (e suporta segmentação de somente um objeto por vídeo), enquanto o método proposto e os outros trabalhos segmentam os quadros inteiros.

Os principais parâmetros para execução do método são N (número de níveis para detecção de borda), R (raio do círculo utilizado na dilatação para extração de bordas), δ (limiar para extração de bordas), N_{obj} (número de objetos), $N_{quadros}$ (número de quadros), b (largura de banda do *mean shift clustering*) e T (tamanho da janela temporal para meta-agrupamento). Uma limitação do método se trata do fato de que a obtenção de um bom resultado é dependente da escolha dos parâmetros corretos para cada vídeo. Além disso, a escolha de valores altos para os parâmetros N_{obj}, $N_{quadros}$, $N_{feicoes}$, b e T prejudica enormemente o tempo de processamento.

Neste trabalho, a seleção dos parâmetros ideais para cada vídeo foi realizada manualmente, através de repetidas execuções. Uma forma de selecionar automaticamente um conjunto de parâmetros que funcione para cada vídeo pode ser objeto de pesquisa futura.

REFERÊNCIAS

ALLEN, G. et al. Computational Science - ICCS 2009. **Lecture Notes in Computer Science**, Baton Rouge, Estados Unidos da América, v.5545, p.945, 2009.

BELAGIANNIS, V. et al. Segmentation based particle filtering for real-time 2d object tracking. **In: Computer Vision - ECCV 2012**, Florência, Itália, p.842–855, 2012.

BROX, T.; MALIK, J. Object segmentation by long term analysis of point trajectories. **In: Computer Vision - ECCV 2010**, Creta, Grécia, p.282–295, 2010.

BURGER, W. et al. **Principles of Digital Image Processing**. Londres, Reuno Unido: Springer, 2009.

CHIN, T.; WANG, H.; SUTER, D. The ordered residual kernel for robust motion subspace clustering. **In: Advances in Neural Information Processing Systems**, Vancouver, Canadá, p.333–341, 2009.

DIMITRIOU, N.; DELOPOULOS, A. Motion-based segmentation of objects using overlapping temporal windows. **Image and Vision Computing**, Amsterdã, Holanda, v.31, n.9, p.593–602, 2013.

FUKUNAGA, K.; HOSTETLER, L. The estimation of the gradient of a density function, with applications in pattern recognition. **IEEE Transactions on Information Theory**, Nova Iórque, Estados Unidos da América, v.vol. 21, p.32–40, 1975.

JAHNE, B. **Digital Image Processing, 6th ed.** Nova Iórque, Estados Unidos da América: Springer, 2005. 449p.

KONRAD, J. Videopsy: dissecting visual data in space-time. **IEEE Communications Magazine**, Nova Iórque, Estados Unidos da América, Jan. 2007.

MAIRE, M. et al. Using contours to detect and localize junctions in natural images. **In: IEEE Conference on Computer Vision and Pattern Recognition - CVPR 2008**, Anchorage, Estados Unidos da América, p.1–8, Jun. 2008.

MANSOURI, A.-R.; KONRAD, J. Multiple motion segmentation with level sets. **IEEE Transactions on Image Processing**, Nova Iórque, Estados Unidos da América, v.12, n.2, p.201–220, Fev. 2003.

MOESLUND, T. B. **Introduction to video and image processing**: building real systems and applications. Nova Iórque, Estados Unidos da América: Springer, 2012.

NEGRI, P.; LOTITO, P. Pedestrian Detection Using a Feature Space Based on Colored Level Lines. **Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, Lecture Notes in Computer Science**, Buenos Aires, Argentina, v.7441/2012, p.885–892, 2012.

PALOU, G.; SALEMBIER, P. Hierarchical Video Representation with Trajectory Binary Partition Tree. **In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, Portland, Estados Unidos da América, p.2099–2106, Jun. 2013.

RISTIVOJEVIC, M.; KONRAD, J. Space-Time Image Sequence Analysis: object tunnels and occlusion volumes. **IEEE Transactions on Image Processing**, Nova Iórque, Estados Unidos da América, v.15, n.2, p.364–376, Fev. 2006.

ROTHER, C.; KOLMOGOROV, V.; BLAKE, A. Grabcut: interactive foreground extraction using iterated graph cuts. **ACM Transactions on Graphics (TOG)**, Nova Iórque, Estados Unidos da América, v.23, n.3, p.309–314, 2004.

SAND, P.; TELLER, S. Particle video: long-range motion estimation using point trajectories. **International Journal of Computer Vision**, Nova Iórque, Estados Unidos da América, v.80, n.1, p.72–91, 2008.

SILVA, L.; SCHARCANSKI, J. Video Segmentation based on Motion Coherence of Particles in Video Sequence. **IEEE Transactions on Image Processing**, Nova Iórque, Estados Unidos da América, v.19, n.4, p.1036–1049, Abr. 2010.

STREHL, A.; GHOSH, J. Cluster ensembles - A knowledge reuse framework for combining multiple partitions. **Journal of Machine Learning Research**, Brookline, Estados Unidos da América, v.3, p.583–617, 2003.

SUNDARAM, N.; BROX, T.; KEUTZER, K. Dense point trajectories by GPU-accelerated large displacement optical flow. **In: Computer Vision - ECCV 2010**, Creta, Grécia, p.438–451, 2010.

SUNDARAM, N.; KEUTZER, K. Long term video segmentation through pixel level spectral clustering on GPUs. **In: IEEE International Conference on Computer Vision Workshops (ICCV Workshops)**, Barcelona, Espanha, p.475–482, Nov. 2011.

TRON, R.; VIDAL, R. A Benchmark for the Comparison of 3-D Motion Segmentation Algorithms. **In: IEEE Conference on Computer Vision and Pattern Recognition**, Minneapolis, Estados Unidos da América, p.1–8, Jun. 2007.

WANG, X.; MA, X.; GRIMSON, E. Unsupervised Activity Perception in Crowded and Complicated scenes Using Hierarchical Bayesian Models. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, Nova Iórque, Estados Unidos da América, v.31, p.539–555, 2009.

APÊNDICE A PSEUDO-ALGORITMO DE AGRUPAMENTO

Este apêndice apresenta um pseudo-algoritmo demonstrando a implementação utilizada para o agrupamento, derivada de (SILVA; SCHARCANSKI, 2010). Os dados de entrada são a matriz de dados de tamanho $[N_{obj}, N_{quadros}, N_{feicoes}]$ e os parâmetros T (janela temporal), b (largura de banda) e um peso associado para cada feição. O dado de saída é um vetor contendo o índice de grupo para cada objeto.

1. É alocada uma matriz vazia C de tamanho N_{obj} por $N_{quadros}.T$-$\Sigma_{n=1}^{T} n$=$N_{quadros}.T$-$T.(T+1)/2$, sendo N_{obj} o número de objetos, $N_{quadros}$ o número de quadros e T o tamanho da janela temporal.

2. É criado o contador k com valor inicial zero. Ao final do processo k terá o valor $N_{quadros}.T$-$\Sigma_{n=1}^{T} n$=$N_{quadros}.T$-$T.(T+1)/2$, ou seja, a segunda dimensão da matriz C

3. Para $i = 1$ até T

   (a) Para $j = 1$ até $N_{quadros} - i$

      i. No quadro j, são localizados na matriz de dados todos os objetos válidos tanto no quadro atual quanto no quadro de índice $j + i$. É retornada uma matriz chamada ind de tamanho $N_{ObjetosEncontrados}$ x 1

      ii. Para cada objeto encontrado no passo anterior, é tomada a diferença entre o valor das feições do objeto no quadro $j + i$ em comparação ao quadro j. É retornada uma matriz chamada $data$ de tamanho $N_{ObjetosEncontrados}$ x $N_{feicoes}$

      iii. É criada a matriz $repweights$, com o número de objetos encontradas e o valor dos pesos associados a cada feição. Essa matriz possui o mesmo tamanho de $data$

      iv. As matrizes $data$ e $repweights$, que possuem o mesmo tamanho, são multiplicadas item a item. O resultado é armazenado na matriz $weighteddata$.

      v. É feito o cálculo do *mean shift clustering* na matriz $weighteddata$ transposta. Esse passe é detalhado no Apêndice B.

      vi. Os dados de retorno do *mean shift clustering* são a matriz $clustCent$, contendo o centroide dos grupos, e a matriz $data2cluster$, contendo o índice de grupo para cada objeto de entrada.

      vii. Os índices de grupo para cada objeto são armazenados na matriz C na posição $(objeto, k)$. O contador k é incrementado.

4. Após a obtenção dos resultados do *mean shift clustering* para todos os pares de quadros, é realizado o meta-agrupamento para gerar um consenso entre os mesmos, contendo o agrupamento final. O detalhamento do meta-agrupamento é apresentado no Apêndice C.

APÊNDICE B PSEUDO ALGORITMO DE MEAN SHIFT CLUSTERING

Este apêndice apresenta um pseudo-algoritmo demonstrando a implementação utilizada na etapa de *mean shift clustering* (FUKUNAGA; HOSTETLER, 1975). O dado de entrada é a matriz de dados $weighteddata$ transposta e a largura de banda b. Os dados de saída são uma matriz $clustCent$ contendo os centroides dos grupos no espaço de feições e o vetor $data2cluster$ contendo o índice de grupo para cada objeto.

1. É recebido o dado de entrada, uma matriz de tamanho $N_{feicoes}$ x $N_{ObjetosEncontrados}$
2. É criado um vetor $initPtInds$ contendo índices de 1 a $N_{ObjetosEncontrados}$
3. A variável que indica o número inicial de pontos no espaço de feições, $numInitPoints$ é criada e inicializada com o valor $N_{ObjetosEncontrados}$
4. É criado o vetor booleano $beenVisitedFlag$ que vai indicar os pontos que não foram visitado. O vetor possui tamanho $N_{ObjetosEncontrados}$
5. É inicializada a matriz $ClusterVotes$ de tamanho $N_{ObjetosEncontrados}$. A matriz irá conter quantos votos cada objeto recebe para pertencer a um determinado grupo
6. Enquando $numInitPoints$ for maior que zero:
   (a) Dentre os objetos marcados como não-visitados no vetor $beenVisitedFlag$, é escolhido um ponto de entrada aleatório $numInitPoint$, que será o primeiro a constar em um novo grupo.
   (b) É inicializada a lista $myMembers$ com os pontos pertencentes ao grupo. Inicialmente ela possui somente $numInitPoint$.
   (c) É inicializada a matriz $thisClusterVotes$, de tamanho $N_{ObjetosEncontrados}$ x 1, que vai conter quantos votos cada objeto vai possuir para pertencer a este grupo.
   (d) É feito o cálculo da média dos pontos em $myMembers$ no espaço de feições, que é armazenada na variável $myMean$
   (e) Os seguintes passos são repetidos até que haja convergência:
      i. É calculada a matriz $sqDistToAll$ contendo a distância euclidiana entre a média $myMean$ e todos os outros pontos no espaço de feições.
      ii. É criada a lista $inInds$, contendo os índices dos objetos cuja distância da média $myMean$, indicada em $sqDistToAll$, seja menor que o valor da largura de banda b

iii. São somados na matriz $thisClusterVotes$, um voto para cada objeto presente na lista $inInds$

iv. O valor atual de $myMean$ é salvo na variável $myOldMean$ para comparação posterior

v. Um novo valor de média no espaço de feições $myMean$ é calculado com todos os objetos contidos em $inInds$

vi. Os novos membros são adicionados a $myMembers$

vii. Os novos membros são marcados como visitados no vetor $beenVisitedFlag$

viii. Considera-se que houve convergência quando ao final da iteração o movimento da média (distância entre $myOldMean$ e $myMean$) for menor que um limiar (no caso 0.1% da largura de banda b)

(f) Após haver convergência, é feita uma checagem se o novo grupo deve ser fundido a algum grupo previamente detectado. O critério para que isso aconteça é que distância entre a média no espaço de feições entre os grupos deve ser menor que 50% da largura de banda. Se isso não for verdadeiro, o grupo detectado permanece como está.

(g) Se o grupo não for fundido, o contador de grupos $numClust$ é incrementado e a média do grupo no espaço de feições é armazenada na matriz $clustCent$. A matriz $thisClusterVotes$ é armazenada como uma coluna em $ClusterVotes$, para possível resolução de conflitos

(h) No caso do grupo ser fundido com um anterior, a média no espaço de feições do anterior é atualizada considerando a inclusão dos novos membros. Os votos em $thisClusterVotes$ são adicionados aos votos do grupo anterior em $ClusterVotes$.

(i) É recalculada a variável que indica quantos pontos não foram visitados, $numInitPts$

7. Após todos os pontos serem visitados, é designado para cada objeto o grupo que contiver mais votos em $clusterVotes$

8. Os dados de saída do algoritmo são a matriz $clustCent$ com os centroides dos grupos e o vetor $data2cluster$ indicando a que grupo pertence cada objeto.

APÊNDICE C META-AGRUPAMENTO

Este apêndice apresenta um pseudo-algoritmo demonstrando a implementação utilizada na etapa de meta-agrupamento (STREHL; GHOSH, 2003). O dado de entrada é a matriz C contendo os resultados do *mean shift clustering* e o dado de saída é o vetor contendo o índice de grupo de cada objeto.

1. É alocada a matriz H de tamanho N_{obj}
2. Para $i = 1$ até $N_{quadros}.T\text{-}\Sigma_{n=1}^{T} n = N_{quadros}.T\text{-}T.(T+1)/2$:
   (a) É tomada a coluna i da matriz C, contendo índices de grupos para cada objeto
   (b) O valor do índice de grupo mais alto presente na coluna é armazenado em N
   (c) Para $j = 1$ até N, são criadas colunas indicando para cada grupo j quais objetos pertencem ao mesmo. Todas as colunas que não forem vazias são adicionadas a H
3. É calculada a distância de Jaccard (similaridade) entre cada par de colunas (hiper-arestas) em H (hiper-grafo). As similaridades são armazenadas na matriz w. A fórmula utilizada é mostrada na Equação 2.
4. São calculadas as distâncias entre as similaridades na matriz w, que são armazenadas na matriz D
5. A partir dos dados de D, é criada a árvore hierárquica dos grupos Z
6. É calculado o nível na árvore Z onde a distância entre os nós combinados foi máxima. É considerado que este foi o ponto onde meta-grupos diferentes foram unidos. Todas as uniões deste ponto em diante se deram entre meta-grupos diferentes. Tendo isso em vista, o número final de grupos $ngrupos$ é o número de níveis entre o atual (incluso) e o último da árvore.
7. A árvore binária é dividida em $ngrupos$ partes seguindo o critério de menos distância. Os resultados são armazenados na estrutura T
8. É criada uma matriz vazia $prob$ de tamanho N_{obj} x $ngrupos$, que vai conter a probabilidade de cada objeto pertencer a um dos grupos finais
9. A probabilidade de cada objeto pertencer a um grupo em qualquer subdivisão utilizada no *mean shift clustering* é preenchida tendo como base sua ocorrência em T

10. É criado o vetor ind_max, que vai conter o índice de grupo de cada objeto
11. Cada objeto recebe o índice de grupo com a maior probabilidade associada, que é armazenado em ind_max