
Universidade Federal de Pernambuco
Centro de Informática

Graduação em Ciência da Computação

# Aprimoramento da etapa de casamento de uma técnica de rastreamento baseado em arestas

Mailson Daniel Lira Menezes

Trabalho de Graduação

Recife
12 de maio de 2013

Universidade Federal de Pernambuco
Centro de Informática

Mailson Daniel Lira Menezes

# Aprimoramento da etapa de casamento de uma técnica de rastreamento baseado em arestas

*Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.*

Orientadora: *Veronica Teichrieb*
Co-orientador: *Francisco Paulo Magalhães Simões*

Recife
12 de maio de 2013

# Agradecimentos

Nesse momento de grande alegria eu agradeço aos meus pais e familiares, que sempre estiveram ao meu lado, desde o começo da minha vida. Agradeço também à minha adorável namorada Zélia, que me apoiou bastante para que eu saísse da zona de conforto e buscasse ingressar no curso que sempre quis, na universidade que sempre desejei estudar. Agradeço aos meus grandes amigos, Caio e Tomás. E por falar em amigos não há como deixar de mencionar as amizades que fiz no CIn e que levarei para o resto da minha vida. John, Galega, Igor e Felipe, obrigado por tornar o CIn um lugar divertido.

Nesses pouco mais de quatro anos de curso tive o contato com pessoas extraordinárias que deixaram uma marca na minha vida acadêmica e pessoal. Não haveria aqui espaço para citar o nome de cada um, mas posso dizer que fico muito grato de ter participado do PET, que foi um lugar que eu tive contato com os mais diferente tipos de pessoas e foi peça fundamental para o meu desenvolvimento acadêmico. Também fiquei feliz por ter, por um breve período, participado de pesquisas no DEN. Por último gostaria a agradecer às pessoas do VOXAR labs por terem me ajudado nesta etapa final do meu curso.

# Resumo

Este trabalho de graduação apresenta uma discussão sobre uma melhoria na etapa de casamento de múltiplas hipóteses de uma técnica de rastreamento 3D baseado em arestas. Essa melhoria visa facilitar a escolha das poses da câmera ao longo da sequência de vídeo para que se possa trabalhar com múltiplas hipóteses de pose no rastreamento. Neste trabalho são discutidas técnicas de rastreamento 3D e conceitos básicos necessários para o entendimentos do texto. Também é feita uma análise da técnica proposta em [1] e uma discussão dos resultados após a implementação da técnica. Após os experimentos conclui-se que a técnica tem uma certa instabilidade em manter uma pose correta, porém ela apresenta facilidade em encontrar uma pose mesmo quando o modelo projetado está bem distante do objeto rastreado.

**Palavras-chave:** arestas, rastreamento, realidade aumentada

# Sumário

# Lista de Figuras

CAPÍTULO 1

# Introdução

O rastreamento 3D é um passo importante das técnicas de realidade aumentada. A partir de uma sequência de imagens 2D deseja-se saber em que posição do espaço 3D está o objeto rastreado, o que é uma tarefa importante e desafiadora. Várias técnicas de rastreamento 3D podem ser encontradas na literatura [2], porém ainda existe muito a ser pesquisado na área. As técnicas existentes, apesar de apresentar uma certa estabilidade no rastreamento, ainda apresentam dificuldades em lidar com elementos externos no ambiente ou movimentação imprevisível da câmera utilizada no rastreamento.

Como será mostrado no capítulo seguinte, o objetivo das técnicas de rastreamento 3D é encontrar a configuração da câmera virtual que representa, da melhor forma, o posicionamento relativo entre a câmera utilizada para capturar os *frames* e o objeto a ser rastreado. Neste trabalho de graduação será apresentada uma técnica que ao invés de calcular apenas uma pose a partir dos dados recuperados da imagem, calcula várias poses para um único *frame*, e ao final somente a melhor delas é selecionada como a pose calculada para aquele *frame*.

Este trabalho está dividido em cinco capítulos. No capítulo 2 são apresentados os conceitos básicos e técnicas já difundidas na literatura sobre rastreamento 3D. No capítulo 3 é feita uma descrição da abordagem utilizada neste trabalho, bem como alguns detalhes de implementação são explicados. No capítulo 4 são feitas análises de qualidade e desempenho da abordagem, e o capítulo 5 finaliza o trabalho com uma discussão final dos resultados dos experimentos.

CAPÍTULO 2

# Conceitos básicos

Segundo Lepetit e Fua [2], rastrear um objeto é identificar sua posição e orientação na cena quando o objeto e/ou a câmera estão em movimento. Em outras palavras, é saber a posição e orientação da câmera virtual (ou pose da câmera) em relação ao objeto presente na cena, dada uma entrada de imagem ou vídeo. O rastreamento de objetos é o principal alvo de pesquisa deste trabalho de graduação, e na sequência são explicados os conceitos fundamentais relacionados ao tema, com foco em rastreamento de arestas.

## 2.1 Representação da câmera

Neste trabalho de graduação será usado um modelo tradicional de câmera com orifício para representar a câmera virtual [6]. A projeção 2D da cena é formada a partir de dados 3D seguindo um modelo de projeção em perspectiva. A formação da imagem pode então ser definida como a projeção de pontos do espaço 3D para um plano 2D. Seja $M = [X,Y,Z]^T$ um ponto 3D em coordenadas de mundo e $m = [u,v]^T$ um ponto 2D em coordenadas de tela, eles se relacionam de acordo com a equação

$$s\tilde{m} = P\tilde{M} \tag{2.1}$$

em que $s$ é um fator de escala que indica a resolução em que o modelo será projetado na tela, $\tilde{m} = [u,v,1]^T$ e $\tilde{M} = [X,Y,Z,1]^T$ são os pontos $m$ e $M$ em coordenadas homogêneas, e $P$ é uma matriz de projeção em perspectiva $3 \times 4$. A matriz $P$ contém informações da câmera como parâmetros de calibração, posição e orientação da câmera no espaço 3D. Ela pode ser decomposta da seguinte forma:

$$P = K[R|t] \tag{2.2}$$

Em (2.2), $K$ representa os parâmetros intrínsecos da câmera. Esses parâmetros, na maioria dos experimentos, não mudam no decorrer do rastreamento e são conhecidos previamente através de uma calibração da câmera. O parâmetro $K$ também é conhecido como matriz de calibração da câmera [2] e contém informações como fator de escala, distância focal e o ponto de interseção do eixo óptico da câmera e o plano de projeção. Estes conceitos podem ser observados na Figura 2.1.

Ainda na equação (2.2), $[R|t]$ é uma matriz $3 \times 4$ e representa os parâmetros extrínsecos da câmera. Diferente dos parâmetros intrínsecos, os extrínsecos mudam no decorrer do rastreamento.

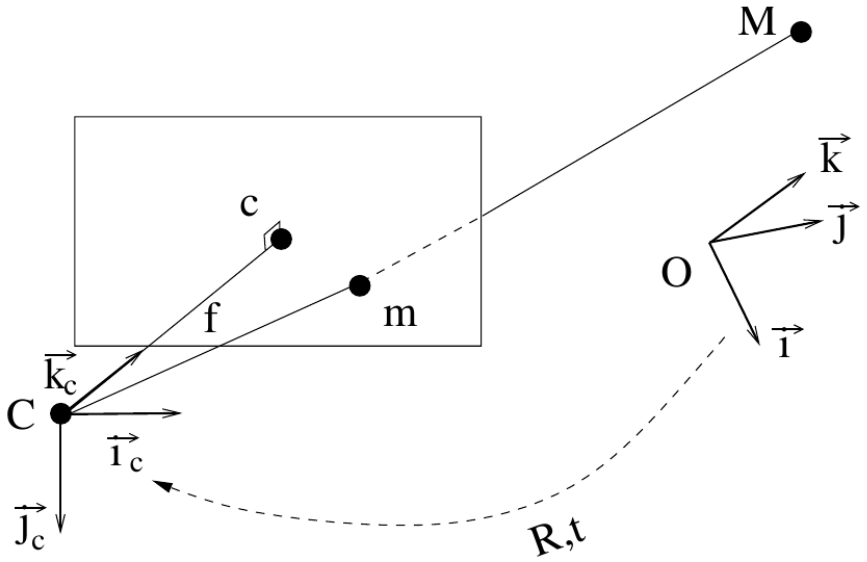



Figura 2.1: Esquema de projeção em perspectiva. Observe na imagem que $(\mathbf{O}, \vec{i}, \vec{j}, \vec{k})$ é o sistema de coordenadas do mundo, $(\mathbf{C}, \vec{i_c}, \vec{j_c}, \vec{k_c})$ é o sistema de coordenadas da câmera, **M** é um ponto 3D e **m** é sua projeção 2D no plano. Imagem retirada de [2].

Ao observar a equação (2.1) pode-se dizer que tendo um ponto 2D $\tilde{m}$ da imagem, para que o ponto $\tilde{M}$ seja projetado em $\tilde{m}$ é necessário descobrir a matriz $P$ da câmera que realize essa operação, ou seja, um $P$ que torne $P\tilde{M}_i \equiv \tilde{m}_i$ válido para todo $i$. Como foi visto em (2.2), $K$ é um parâmetro conhecido da câmera, então o que queremos descobrir, na verdade, é a matriz $[R|t]$, que a partir deste momento do texto iremos chamar de pose da câmera.

### 2.1.1 Cálculo da pose

O cálculo da pose é uma estimativa para que o ponto $\tilde{M}_i$, ao ser projetado em coordenadas de tela ($P\tilde{M}_i$), seja o mais próximo possível do ponto $\tilde{m}_i$, que é um ponto 2D extraído da imagem. A distância do ponto projetado $P\tilde{M}_i$ para seu correspondente extraído da imagem ($\tilde{m}_i$) é chamada de erro de reprojeção. O objetivo das técnicas de rastreamento que usam essa representação de câmera é encontrar uma pose ($[R|t]$) cuja soma dos erros de reprojeção seja a menor possível. Em outras palavras, deseja-se resolver a equação

$$[R|t] = \arg\min_{[R|t]} \sum_i \text{dist}^2(P\tilde{M}_i, \tilde{m}_i) \tag{2.3}$$

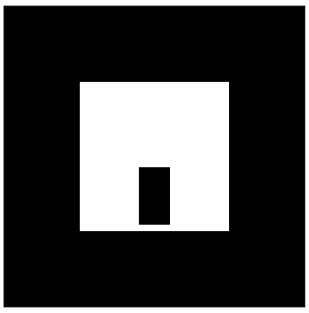
Essa equação não fornece uma solução polinomial de forma fechada, por isso métodos de otimização como o Gauss-Newton [7] ou o Levenberg–Marquardt [8] devem ser utilizados.

Estimadores robustos como um *M-estimator* [2] também podem ser utilizados, e são bastante úteis para auxiliar em uma solução adequada. O papel principal destes estimadores é eliminar a influência de *outliers*, que são algumas poucas correspondências 2D-3D grosseiramente erradas, mas que podem influenciar bastante negativamente no resultado final.

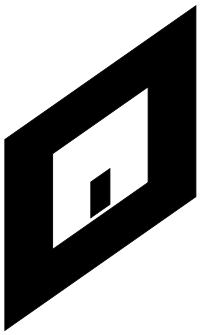
## 2.2 Rastreamento de objetos

Para auxiliar no cálculo da pose é preciso extrair *features*, que são componentes mais simples da cena como marcadores [9], arestas do objeto [10], textura do objeto [11], e pontos de interesse [11], que auxiliem na recuperação da pose (3D) da câmera virtual a partir de uma sequência de imagens 2D. Conhecendo a posição 3D dessas *features* no quadro anterior e extraindo sua posição 2D no quadro atual já é um grande passo para realizar o rastreamento. A intenção dos algoritmos de rastreamento é descobrir uma pose da câmera que, ao usá-la para fazer a projeção 2D destas *features* 3D, essas *features* tenham as mesmas posições 2D das que foram extraídas do quadro atual.

Como foi dito anteriormente, existem na literatura vários tipos de informações que podem ser usadas como *features*. Uma das formas mais conhecidas de rastreamento é a que insere marcadores fiduciais na cena, como o da Figura 2.2a. Nesta técnica deve-se conhecer *a priori* o marcador a ser utilizado. O trabalho consiste então em encontrar o marcador em cada quadro e saber que operação feita na câmera (rotação, translação, cisalhamento ou deformações projetivas) deixaria o marcador tomado como modelo (Figura 2.2a) com a mesma configuração que o marcador extraído da imagem (Figura 2.2b).



(a) Exemplo de marcador artificial.



(b) Exemplo de marcador extraído de uma imagem.

Figura 2.2: Exemplo de marcador e sua projeção.

O uso de marcadores, no entanto, requer que uma interferência no ambiente seja feita e, apesar de simplificar o rastreamento 3D, nem sempre pode ser usado. Isto acontece porque podem existir limitações técnicas que impeçam a introdução prévia de elementos auxiliares no ambiente real ou podem até existir restrições baseadas na decisão do usuário final. Neste caso deve-se buscar *features* que estejam naturalmente presentes na cena, o que muitas vezes ocorre quando elas são parte do próprio objeto a ser rastreado. Abrir mão de marcadores artificiais torna o rastreamento muito mais desafiador, e a discussão dessas técnicas será o foco do resto deste trabalho de graduação.

Existem várias formas de rastrear um objeto na cena sem a utilização de marcadores externos [12] e entre elas existe o rastreamento baseado em modelo e o *Structure from Motion* (*SfM*).

O rastreamento baseado em modelo é caracterizado pelo uso de um modelo 3D pré-definido do objeto a ser rastreado. O modelo serve para saber as características do objeto que serão

usadas para rastrear. Já na técnica baseada em *SfM* [12] não é necessário um conhecimento prévio da cena e, por isso, é bastante útil quando o ambiente a ser rastreado é desconhecido.

Ao se comparar as duas abordagens pode-se dizer que a baseada em modelos é computacionalmente mais eficiente que a *SfM* [3]. A *SfM* baseia-se na análise de todo o *frame* da sequência de vídeo para poder identificar um movimento na câmera, e sabendo-se que poucos segundos de um vídeo podem conter bastante informação, essa análise pode se mostrar lenta computacionalmente. A técnica baseada em modelos, por sua vez, pode tirar proveito de um modelo já conhecido do objeto a ser rastreado. Com isso apenas características-chave precisam ser analisadas para o rastreamento. Neste trabalho de graduação a técnica baseada em modelos é explorada.

### 2.2.1 Rastreamento sem marcadores baseado em modelo

Algumas das técnicas de rastreamento sem marcadores, como foi dito anteriormente, baseiam-se no uso de um modelo 3D (construído em uma ferramenta CAD como o 3D Studio Max [13] ou Maya [14]) do objeto a ser rastreado [2]. Ter esse modelo é importante, já que a maioria das técnicas de rastreamento sem marcadores utiliza *features* do próprio objeto. Considerando uma análise superficial, a ideia é equivalente ao método que utiliza marcadores: extrair as *features* da imagem 2D capturada e tentar chegar a uma pose da câmera virtual que, ao ser usada para fazer a projeção 2D do modelo 3D do objeto, chegamos a uma imagem equivalente àquela capturada do objeto.

As técnicas baseadas em modelo podem usar uma gama de informações do objeto para dar suporte ao processo de rastreamento, sendo uma delas a textura do objeto [12]. O rastreamento baseado em textura pode ser feito de diferentes formas: uma das técnicas aplica um modelo de distorção a uma imagem de referência [15]; outra utiliza pontos de interesse, e se baseia no casamento de *features* das imagens recuperadas com *features* de uma imagem de template pré-definida [2].

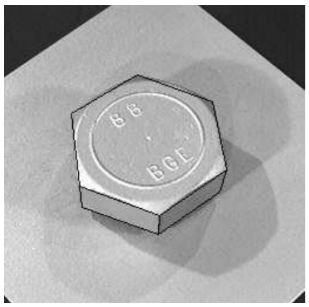
Além da textura do objeto, um outro tipo de informação que pode ser usada são as suas arestas [2]. O processo de casamento de aresta pode ser descrito da seguinte forma: tendo as arestas do modelo CAD pré-definido, o objetivo é encontrar os pontos de forte gradiente da imagem para formar linhas (que serão as arestas do objeto) e então encontrar a pose que projeta as arestas do modelo 3D o mais próximo possível das arestas extraídas da imagem.

A escolha entre textura ou arestas do modelo depende do objeto a ser rastreado. A Figura 2.3a ilustra um objeto com textura cujas arestas não são bem definidas. Se for feita uma rotação do objeto no eixo $Z$, conforme mostrado na imagem, dificilmente um algoritmo baseado em arestas irá percebê-la corretamente. Por outro lado, o objeto mostrado na Figura 2.3b, apesar de não ter textura, tem arestas bem definidas. No presente trabalho as técnicas de rastreamento baseado em arestas são investigadas.

### 2.2.2 Técnicas de rastreamento baseado em arestas

Independente das técnicas que serão apresentadas abaixo, o rastreamento baseado em arestas segue praticamente o mesmo *pipeline*. Primeiro é preciso saber a pose do *frame* anterior. Se for o primeiro *frame* este, ainda assim, terá que ser informado, e é por isso que nos experimentos


(a) Este objeto, por ser cilíndrico, não possui arestas bem definidas. No entanto seu desenho é um convite ao uso de técnicas de rastreamento baseadas em textura.



(b) Este objeto não possui textura muito complexa, porém suas arestas são bem definidas. Imagem retirada de [16].

Figura 2.3: Dois tipos de objetos: um com textura e outro com arestas bem definidas.

é preciso definir uma pose inicial. Depois de obtida a pose do *frame* anterior, o *frame* atual é capturado a partir de uma sequência de vídeo. Essa imagem capturada será então analisada em busca de pontos de forte gradiente para a extração de bordas. Obtém-se então as arestas do modelo que estão visíveis após serem projetadas utilizando a pose do quadro anterior. As arestas visíveis são aquelas que ao serem projetadas na tela não são oclusas por outras partes do objeto (considerando um objeto opaco). Em seguida, a projeção das arestas visíveis é comparada com as bordas extraídas da imagem. Nessa comparação é feito um casamento entre a aresta 3D projetada do modelo e a aresta extraída da imagem (veja Figura 2.4). Esse casamento dá à aresta extraída da imagem uma estimativa inicial de sua posição 3D. Após isso é feito o cálculo da pose para que as arestas do modelo 3D projetado na tela, utilizando a pose calculada, fiquem o mais próximo possível das arestas extraídas da imagem. Esta pose será então a pose do *frame* atual.

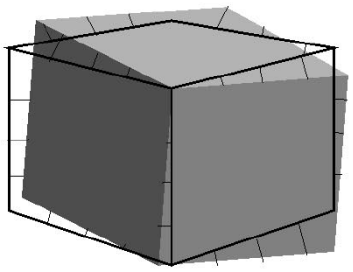


Figura 2.4: Casamento de arestas do modelo (*wireframe*) com arestas do objeto na imagem (cubo cinza). Imagem retirada de [3].

Como descrito no *pipeline*, uma das etapas do rastreamento baseado em arestas é a extração das arestas do quadro atual, que é uma imagem 2D. Duas técnicas importantes a serem

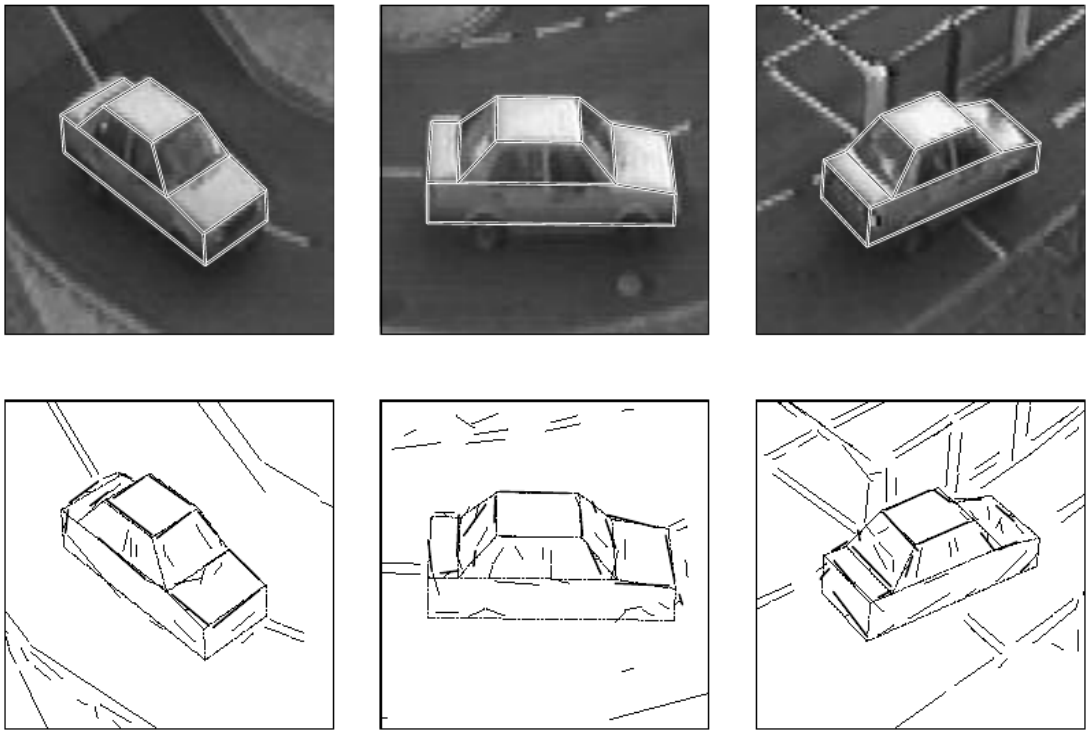


Figura 2.5: Processo de extração explícita de arestas. Imagem retirada de [4].

mencionadas são a extração explícita de aresta [4] e a técnica da amostragem de pontos[3].

Na primeira técnica são extraídos segmentos de reta da imagem utilizando algum algoritmo de detecção de linha, como a transformada de Hough, ao mesmo tempo em que o modelo é renderizado utilizando a pose do *frame* anterior, como mostra a Figura 2.5. Um procedimento recursivo é então utilizado com a finalidade de encontrar as melhores correspondências entre as arestas do modelo 3D e os segmentos de reta 2D extraídos da imagem. Neste procedimento as arestas projetadas do modelo são agrupadas com os segmentos de reta extraídos de acordo com a menor distância de Mahalanobis. Na sequência a técnica da amostragem de pontos é explicado, tendo sido o alvo principal de pesquisa deste trabalho de graduação.

### 2.2.3 Amostragem de pontos

A técnica de amostragem de pontos [3] apresenta outra maneira de fazer o casamento entre as arestas do modelo e as arestas do objeto. Neste caso as arestas da imagem não são extraídas explicitamente para depois serem comparadas com as arestas projetadas no modelo, como na abordagem anterior.

Cada aresta do modelo, que será chamada de $E_i$, é dividida de modo que se obtenham $n_i$ pontos amostrados, sendo $\{e_{i,j}\}$ o conjunto dessas amostras de $E_i$. A técnica de amostragem de pontos associa pontos do modelo 3D com pontos 2D extraídos da imagem. A extração desses pontos é feita a partir de uma busca de pontos de forte gradiente, chamados de $e'_{i,j}$, na normal de cada amostra $e_{i,j}$, como pode ser visto na Figura 2.6.

Tendo em vista esse casamento de amostras do modelo com amostras do objeto, o objetivo

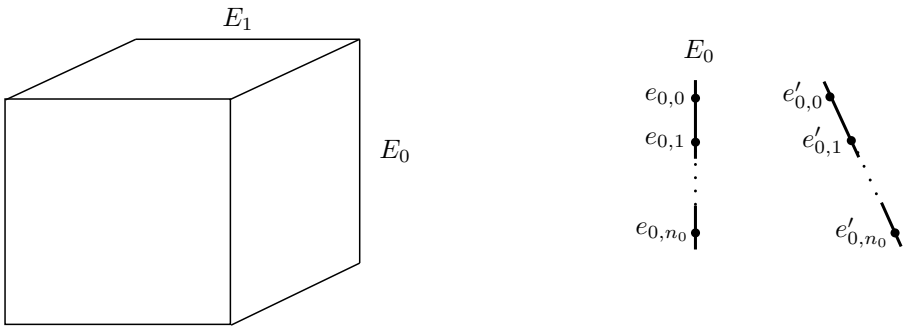



Figura 2.6: Diagrama de pontos amostrados. Pode-se observar que a aresta $E_0$ da imagem possui $n_0$ amostras e cada uma delas está associada a uma hipótese $e'_{i,j}$, resultado da busca na normal da amostra.

do algoritmo é encontrar uma pose que minimize a diferença entre a projeção da amostra, $P \cdot e_{i,j}$, e seu correspondente na imagem capturada. Utilizando a equação (2.3), o problema pode ser descrito como

$$[R|t] = \arg\min_{[R|t]} \sum_i \text{dist}^2(P \cdot e_{i,j}, e'_{i,j}) \tag{2.4}$$

É importante lembrar que somente as arestas visíveis do modelo são usadas, porém pode existir o caso em que a aresta esteja parcialmente visível, e uma das vantagens desta abordagem é poder lidar com esse caso, como mostra a Figura 2.7.

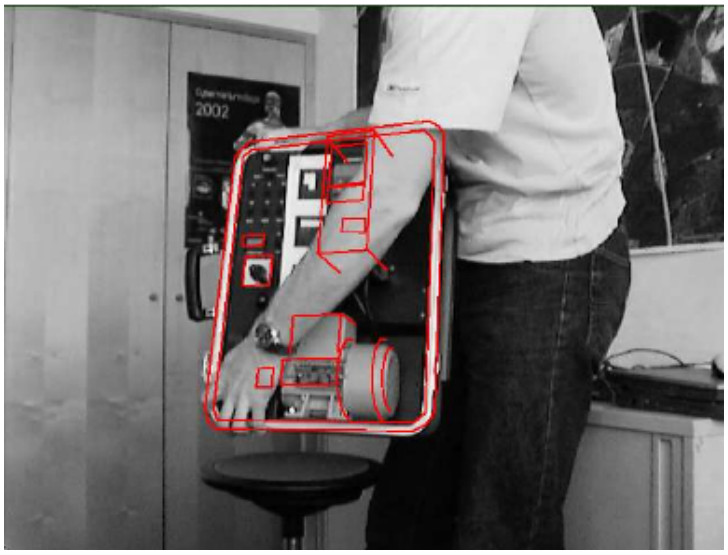


Figura 2.7: Objeto com oclusão parcial das arestas. Imagem retirada de [5].

Outra vantagem é que a busca por pontos de forte gradiente na imagem é bastante simplificada. Ela acontece apenas em uma dimensão e somente nas normais das amostra do modelo o que torna a execução do algoritmo bastante rápida.

### 2.2.4 Múltiplas hipóteses

Um dos problemas que ocorrem nas técnicas baseadas em arestas é que nem sempre se consegue extrair com precisão as arestas da imagem. Na busca por pontos na normal da amostra pode existir mais de um ponto de forte gradiente devido ao ruído da câmera, objetos externos na cena, sombra do objeto ou até arestas do próprio objeto, dependendo da pose. A Figura 2.8 ilustra um exemplo em que mais de um ponto de forte gradiente é encontrado.

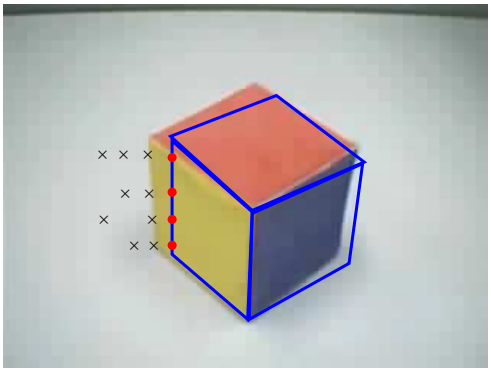
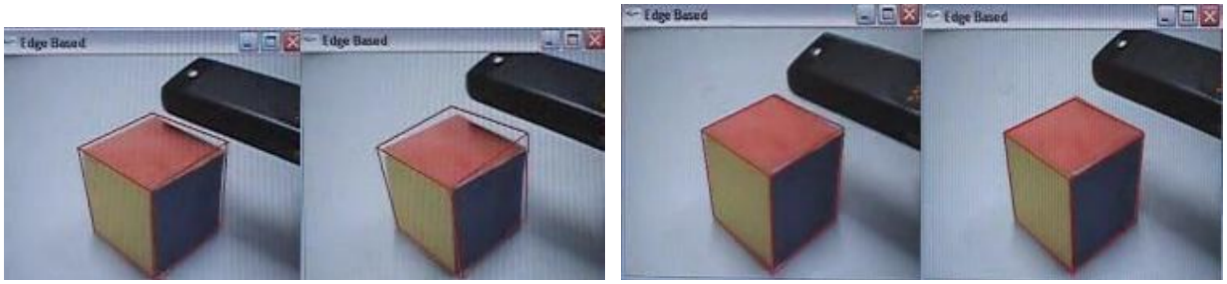


Figura 2.8: O aramado azul representa o modelo da pose anterior; a imagem do cubo é a cena atual da qual pretende-se extrair a pose; os pontos vermelhos são as amostras do modelo; e os pontos em formato de X são as correspondências encontradas na cena atual. Nota-se que devido a ruídos na imagem algumas amostras possuem mais de uma correspondente.

Neste caso é preciso escolher com qual dos pontos achados será feito o casamento da amostra. Uma das alternativas para a escolha do ponto extraído seria escolher a opção de maior gradiente. No entanto essa escolha nem sempre é uma boa opção, pois um alto contraste nem sempre significa que o ponto tem maiores chances de pertencer ao modelo. Na Figura 2.9a observa-se que objetos externos podem confundir o casamento das amostras.




(a) Experimento com hipótese única. Observe que os pontos do controle remoto (de maior contraste) confundem o rastreamento.

(b) Experimento com múltiplas hipóteses.

Figura 2.9: Resultado de um rastreamento com múltiplas hipóteses e com hipótese única.

Para este tipo de problema uma boa solução é o uso de múltiplas hipóteses da amostra [17]. Ou seja, cada amostra $e_{i,j}$ está associada a um conjunto de hipóteses $\{e'_{i,j,l}\}$, e somente as

correspondências mais prováveis seriam usadas no processo de minimização. Na prática essa técnica é menos vulnerável a mudanças no ambiente. Na Figura 2.9b é mostrado o resultado de uma técnica de rastreamento com múltiplas hipóteses de amostra. Observe que mesmo aproximando um objeto com forte gradiente, o rastreamento sofre pouca distorção.

Para fins práticos as hipóteses mais prováveis são aquelas que mais se aproximam da amostra projetada. Utilizando estas informações e a equação (2.4) pode-se chegar à seguinte fórmula de cálculo de pose:

$$[R|t] = \arg\min_{[R|t]} \sum_{i} \text{dist}^2(P \cdot e_{i,j}, \arg\min_{e'_{i,j,l}}(\text{dist}(e'_{i,j,l}, P \cdot e_{i,j}))) \tag{2.5}$$

em que $\arg\min_{e'_{i,j,l}}(\text{dist}(e'_{i,j,l}, P \cdot e_{i,j}))$ resulta na hipótese com a menor distância da amostra projetada.

CAPÍTULO 3

# Rastreamento com múltiplas hipóteses de pose

A técnica estudada e desenvolvida neste trabalho é baseada em arestas e usa um modelo pré-definido para a pose inicial. A extração das arestas é feita por amostragem de pontos, técnica descrita no capítulo anterior.

Na técnica de múltiplas hipóteses um dos candidatos é escolhido para ser associado com a amostra do modelo, no caso, o candidato mais próximo da aresta projetada. Entretanto isso nem sempre resulta na melhor pose final, pois outros pontos de forte gradiente, como objetos externos ou até uma outra aresta do próprio objeto, podem confundir todo o cálculo da pose só porque estão mais próximos de uma determinada aresta do modelo. Isso pode ser visualizado na Figura 3.1, em que além da aresta do objeto, o lápis também é identificado pelo algoritmo de busca. O fato de uma das arestas do modelo projetado estar mais perto do lápis do que do próprio objeto faz com que o algoritmo de rastreamento a utilize como aresta do objeto e faça o cálculo de pose a partir disso, o que vai fazer uma grande diferença no processo de rastreamento a partir de então. Uma forma de contornar essa situação seria também escolher outras hipóteses (não somente a mais próxima) e calcular novas poses a partir de diferentes agrupamentos amostra-hipótese. O trabalho [1] propõe exatamente isso: que sejam calculadas diversas poses, utilizando hipóteses diferentes, para que no final as poses calculadas sejam comparadas e a melhor dentre elas seja escolhida.

Um dos principais passos do algoritmo visa fazer combinações de agrupamento amostra-hipótese. Isso é importante, pois combinar todos os possíveis casamentos amostra-hipótese não é uma estratégia válida, já que isso pode significar o cálculo de milhares de poses por cada *frame*, mesmo que o objeto rastreado seja simples, como um cubo. Um outro problema de se fazer todas as possíveis combinações é que nem sempre um agrupamento de hipóteses revela uma possível aresta do objeto. Na Figura 3.2 temos as amostras (em vermelho) e suas respectivas hipóteses (quadrados). A Figura 3.2 exemplifica um caso de um casamento qualquer de amostras e hipóteses em que as hipóteses escolhidas estão em verde. A escolha dessas hipóteses claramente não resultaria em uma pose válida, pois o sistema iria tentar convergir uma de suas arestas (a com amostras vermelhas) para os pontos marcados em verde, e isso estaria longe de uma pose válida. Como o algoritmo trabalha com múltiplas hipóteses de pose, a pose gerada pela Figura 3.2 provavelmente seria descartada em um passo posterior, mas é melhor evitar esse tipo de agrupamento para que sejam feitos menos cálculos de pose desnecessários. O trabalho proposto em [1] sugere que se façam agrupamentos que sejam o mais próximo possível de uma reta, como exemplificado na Figura 3.3. Seguindo essa estratégia pode-se dizer que a abordagem trabalha não com múltiplas hipóteses de pontos do objeto, mas com múltiplas hipóteses de aresta. Veja a Figura 3.4.

O uso de múltiplas hipóteses de aresta simplifica a escolha de hipóteses para se calcular

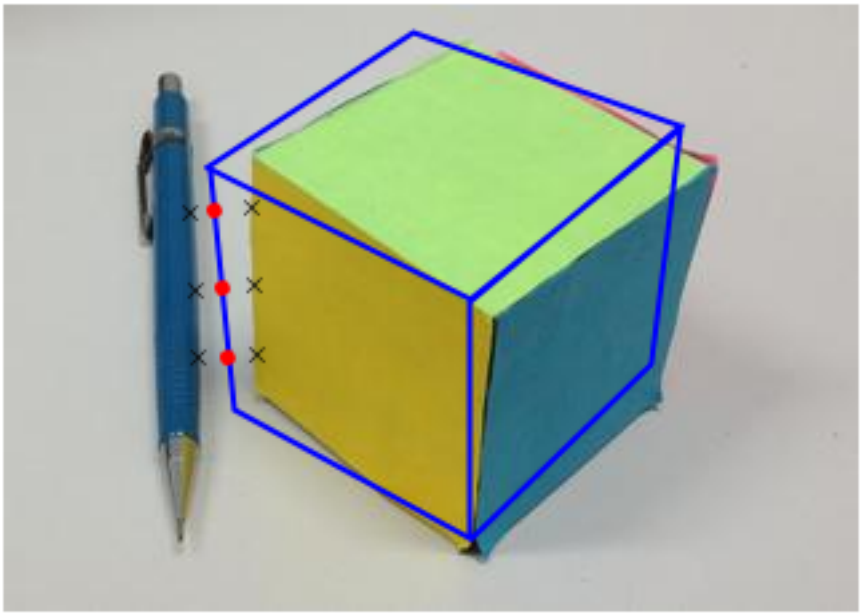


Figura 3.1: Casamento de hipóteses errado. O algoritmo encontrou múltiplas hipóteses para as amostras, mas optou por casar com as hipóteses erradas porque estavam mais próximas.

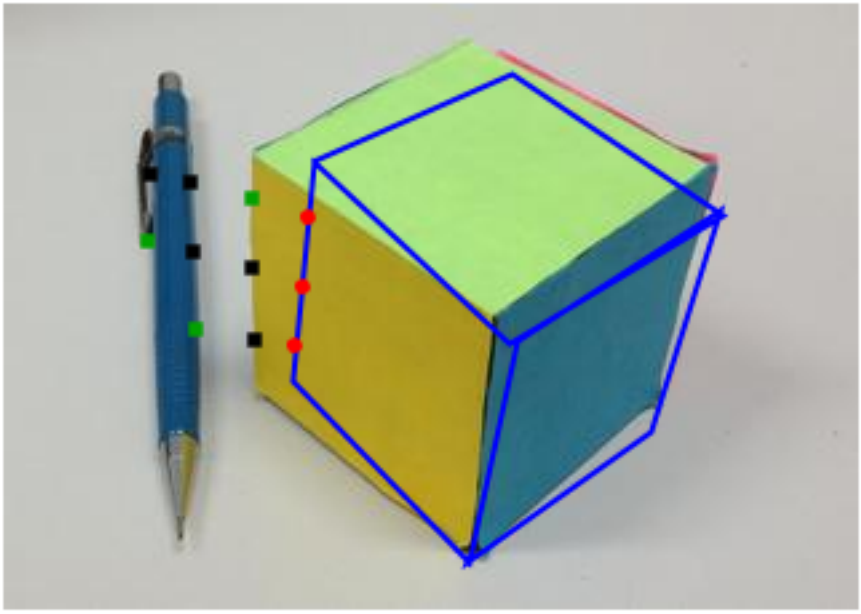


Figura 3.2: Hipóteses agrupadas sem formar uma linha. Os pontos vermelhos são as amostras do modelo; os quadrados são as hipóteses de cada amostra; os quadrados de cor verde são as hipóteses escolhidas.

novas poses. No caso, basta selecionar uma hipótese de aresta para cada aresta visível do modelo, que já é possível calcular uma nova pose. O critério para escolha da hipótese de aresta

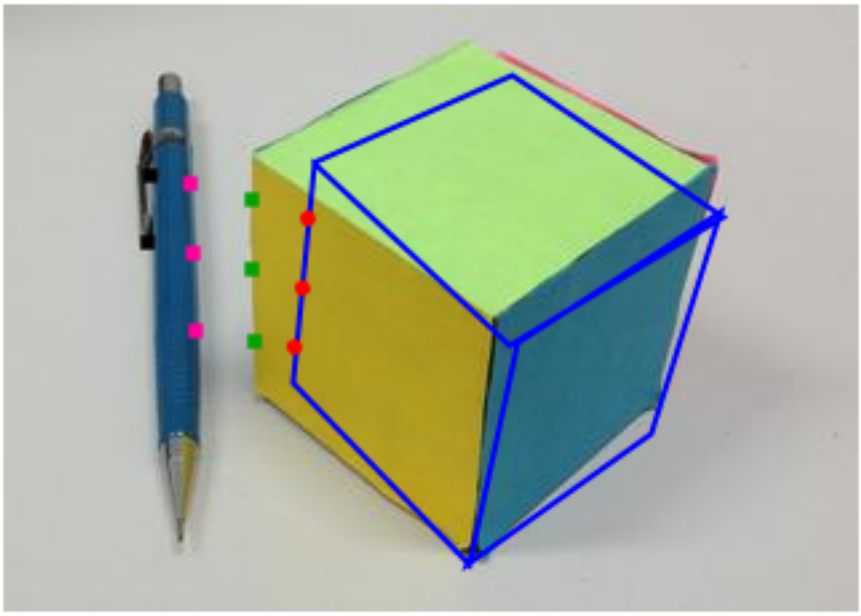


Figura 3.3: Hipóteses agrupadas de modo que se aproximem de uma linha (agrupamentos rosa e verde). Esse tipo de agrupamento é mais próximo de uma aresta que o mostrado na Figura 3.2.

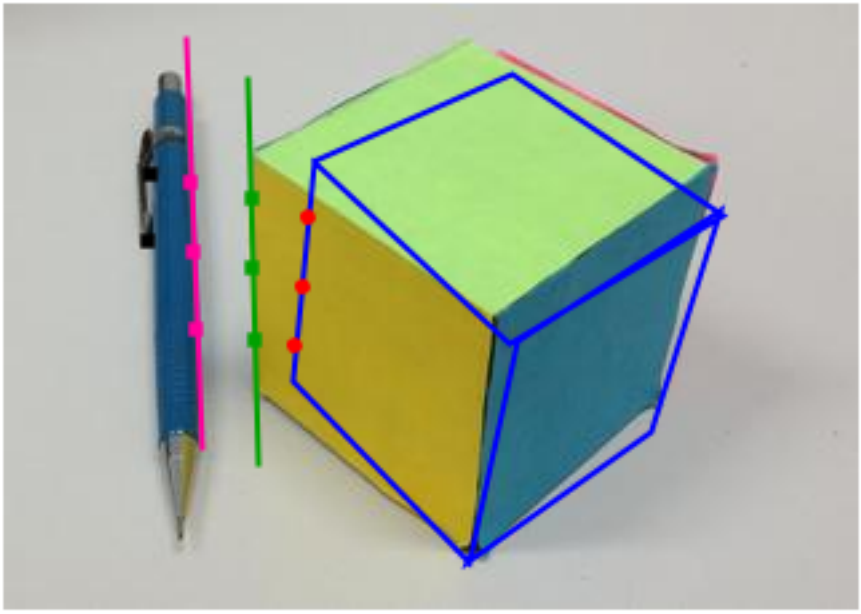


Figura 3.4: Dois tipos de agrupamento, um rosa e outro verde, formando duas hipóteses de aresta de mesma cor.

será discutido em breve, mas é importante ressaltar que apesar de todos os passos feitos até então, não se sabe com clareza quais das múltiplas hipóteses de aresta são arestas do objeto. É por isso que são feitas diferentes combinações dessas hipóteses para que diferentes poses

sejam calculadas. O número de vezes em que hipóteses de arestas são escolhidas e novas poses são calculadas depende da quantidade de poses que se deseja obter. Depois de um número arbitrário de poses terem sido calculadas, o último passo é escolher uma pose dentre todas as poses calculadas. Cada pose possui um erro de reprojeção e o valor desse erro é o que vai ser usado para escolher a melhor pose.

Em poucas palavras o algoritmo se resume a:

- Projetar o modelo usando a pose do quadro anterior;
- Amostrar as arestas do modelo projetado;
- Fazer uma busca por pontos de forte gradiente na normal de cada amostra. Esses pontos serão as hipóteses da amostra;
- Em cada aresta, criar grupos de hipóteses de modo que não existam duas hipóteses em um mesmo grupo que pertençam à mesma amostra. Esses grupos devem aproximar, da melhor forma possível, seus elementos de uma reta;
- Para cada aresta visível do modelo, escolher uma de suas hipóteses de aresta e calcular a pose que projeta o modelo o mais próximo possível das arestas selecionadas. Observe que a escolha das hipóteses de aresta seguido do cálculo da pose se assemelha bastante à técnica de extração explícita de aresta, vista na seção 2.2.2;
- Repetir o passo anterior tantas vezes quanto forem a quantidade de poses necessárias;
- Avaliar o erro de cada pose calculada e escolher a que possuir o menor erro de reprojeção.

## 3.1 Extraindo a hipótese de aresta

Uma das primeiras etapas do algoritmo é o agrupamento das hipóteses de cada amostra em conjuntos cujos elementos formam uma reta (ou o mais próximo disso). Vale ressaltar que o processo de extração dessas hipóteses é pela amostragem de pontos, descrita na seção 2.2.3.

As hipóteses de aresta são formadas agrupando $k_i$ conjuntos de $e'_{i,j,l}$. Cada aresta $E_i$ do modelo possui um conjunto $\{e_{i,j}\}$ de amostras (veja a seção 2.2.3) e cada uma dessas amostras está associada a um conjunto de hipóteses $\{e'_{i,j,l}\}$, como mostra a Figura 3.5. O número $k_i$ de conjuntos é dado pelo maior número de hipóteses detectadas por uma amostra da aresta $E_i$, ou seja $k_i = max_j\{n_{i,j}\}$, sendo $n_{i,j}$ o número de candidatos associados à amostra $e_{i,j}$.

O agrupamento dessas hipóteses é feito utilizando o algoritmo de classificação *k-means* [18]. A classificação tem a tarefa de decidir a que grupo cada hipótese vai pertencer. Como o objetivo é que o conjunto de pontos de cada grupo (ou classe) sejam o mais próximo possível de serem colineares, esta etapa deve alocar as hipóteses para cada grupo de forma que cada novo elemento adicionado perturbe o menos possível a colinearidade do grupo. Seja $\{e'_{i,j,l}\}$ o conjunto de elementos da aresta $E_i$, o objetivo é distribui-los entre as $k_i$ classes de $E_i$, definidas por $(c^i_1, \ldots, c^i_{k_i})$.

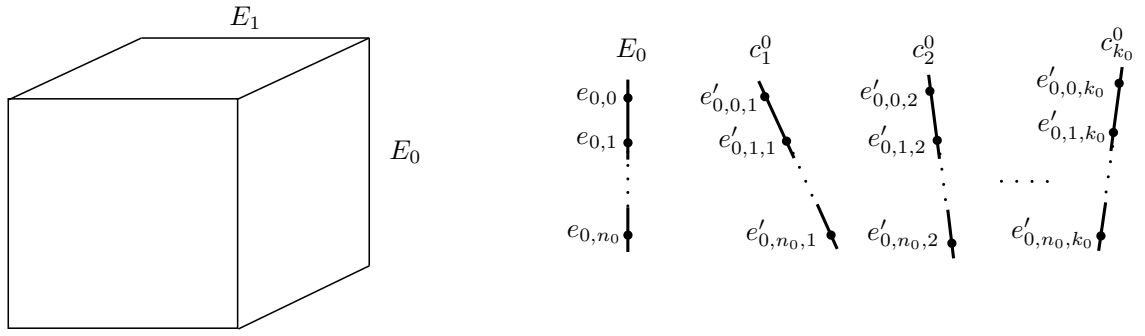



Figura 3.5: Diagrama de múltiplas hipóteses. A aresta $E_0$ possui $n_0$ amostras e contém $k_0$ *clusters* $c_i^0$.

Inicialmente as hipóteses $e'_{i,j,l}$ da aresta $E_i$ são agrupadas nas classes $(c_1^i, \ldots, c_{k_i}^i)$ na ordem em que elas foram encontradas na busca pela normal. Dessa forma a classe $c_m^i$ é formada inicialmente pelo conjunto $\{e'_{i,j,m}\}$, em que $0 < j \leq n_i$ e $n_i$ é o número de amostras da aresta $E_i$. Em outras palavras, as hipóteses mais próximas das amostras serão colocadas nas primeiras classes. Esse é um bom agrupamento inicial, já que boa parte das amostras já estão localizadas no seu *cluster* final.

Em seguida ocorre o procedimento iterativo de reagrupar as hipóteses. Nessa etapa, o primeiro passo é computar o centróide de cada classe. O centróide é uma reta do espaço cuja soma das distâncias de cada elemento da classe para a reta é a menor possível. A estimativa dessa reta é calculada através de um procedimento denominado *fitline* [19]. O que ele faz basicamente é encontrar uma reta que minimiza o erro (definido pela soma das distâncias de cada elemento da classe à reta) de cada *cluster*.

Para cada amostra $e_{i,j}$ é executado outro procedimento iterativo que reagrupa as hipóteses nos *clusters*. Nessa etapa é calculada a distância das hipóteses $\{e'_{i,j,l}\}$ da amostra $e_{i,j}$ para todos os centróides. Isso é feito para que a hipótese seja movida para o *cluster* cujo centróide seja o mais próximo. Porém pode acontecer de mais de uma hipótese da mesma amostra ter o mesmo *cluster* como o mais próximo, mas infelizmente somente uma delas pode ir para o *cluster*, pois como foi visto, não podem existir duas hipóteses de uma mesma amostra em uma mesma classe. A solução para isso foi definir que a primeira hipótese teria prioridade sobre as hipóteses seguintes, ou seja, quando uma hipótese tiver sido movida para uma classe, nenhuma das outras hipóteses seguintes que serão avaliadas poderão optar pela mesma classe. Quando ocorrer da hipótese optar por uma classe que já tiver sido ocupada, ela será movida para o segundo *cluster* mais próximo. Devido a isso, ao invés do cálculo das distâncias guardar apenas o centróide mais próximo da hipótese, nessa etapa cada hipótese mantém uma lista com os centróides ordenados por ordem de distância. Assim fica mais fácil de saber qual o segundo (ou terceiro) centróide mais próximo quando se tenta alocar a hipótese a um *cluster* já ocupado.

No trabalho original [1] é dito que esse último passo seja repetido até que não haja mais trocas entre as classes, mas na prática precisou-se estabelecer um número máximo de iterações, pois muitas vezes o algoritmo repetia indefinidamente.

Ao final das iterações é computado o erro residual $r_m^i$, de cada classe $c_m^i$. O erro residual é

calculado pela média das distâncias de cada elemento da classe para o seu centróide. Ele indica o quão próximo de uma reta o conjunto de elementos da classe está, como mostra a equação

$$r_m^i = \frac{1}{N}\sum_{j=0}^{N} \text{dist}(e'_{i,j,m}, c_m^i) \tag{3.1}$$

em que $N$ é o número de elementos do *cluster* $c_m^i$, e $\text{dist}(e', c)$ é a função que calcula a distância do ponto $e'$ à reta $c$. O resíduo $r_m^i$ será usado na próxima seção e influenciará na escolha das hipóteses de aresta.

## 3.2 Obtendo as hipóteses de pose

As hipóteses de pose são calculadas a partir de um conjunto de hipóteses de aresta selecionadas. Nesse ponto não é fácil avaliar quais das hipóteses de aresta são as mais corretas, por isso que elas são combinadas de formas diferentes de modo que sejam calculadas várias poses. Portanto é razoável que em cada combinação as hipóteses de aresta sejam escolhidas aleatoriamente. Entretanto cada hipótese tem um erro residual $r_m^i$ associado, e apesar desse erro não indicar necessariamente que uma determinada hipótese é uma boa escolha, ele pode pelo menos ajudar na escolha da aresta. É por isso que apesar da escolha ser aleatória, cada hipótese de aresta $c_m^i$ tem um peso $w_m^i$ que o ajuda a ser sorteado. O peso $w_m^i$ é dado pela equação

$$w_m^i = \begin{cases} e^{-\lambda\left(\frac{r_m^i - r_{min}^i}{r_{max}^i - r_{min}^i}\right)^2} & \text{se } r_{max}^i \neq r_{min}^i \\ 1 & \text{senão.} \end{cases} \tag{3.2}$$

em que $\lambda$ é um parâmetro que pode ser ajustado e indica a importância do residual no cálculo do peso [1]. Nos experimentos do Capítulo 4 o valor utilizado para o parâmetro $\lambda$ foi de 0.001.

Mesmo tendo calculado os pesos, é importante fazer uma escolha aleatória porque pode ser que alguma hipótese, mesmo tendo o menor peso, seja a melhor escolha para a pose do quadro atual. Por outro lado vale a pena se utilizar dos pesos (e não somente escolher uma hipótese ao acaso) porque o fato de um *cluster* ser formado por pontos que se aproximam bastante de uma reta já dá um indício (mesmo não sendo tão forte) que determinada hipótese pode ser uma boa escolha.

Após isso é calculado o erro de reprojeção de cada pose formada. A que possuir menor erro é considerada a pose da cena atual e será usada para as próximas iterações do algoritmo.

## 3.3 Uso da GPU

Neste trabalho de graduação a técnica foi implementada tanto em C++ quanto em CUDA [20], plataforma de computação paralela criada pela NVIDIA que compila código para ser executado na GPU. A decisão de se implementar para essa plataforma foi feita após avaliar que o algoritmo era computacionalmente complexo, mas com cálculos bastante independentes, o que é um excelente caso para implementar um algoritmo que seja executado em paralelo. As placas

atuais da NVIDIA possuem várias unidades de processamento que, apesar de serem individualmente mais lentas que uma CPU atual, tem a vantagem de se comunicar rapidamente entre seus outros *cores*.

As placas da NVIDIA que suportam CUDA agrupam suas unidades de processamento em blocos e cada bloco, por sua vez, possui várias *threads* (veja a Figura 3.6). Diferentes *threads* podem trocar mensagens entre si através da memória compartilhada, contanto que elas estejam no mesmo bloco. Se estiverem em blocos diferentes elas podem se comunicar através da memória global, porém esse tipo de comunicação é mais lento. Veja o diagrama da Figura 3.7.

### 3.3.1 Implementação em GPU

Na etapa de classificação do algoritmo apresentado, pode-se observar que cada aresta $E_i$ trabalha de forma independente das demais. Nenhum tipo de cálculo feito em uma aresta interfere no resultado da outra. Isso já é um bom indício que a clusterização utilizando o *k-means* pode ser executada em diferentes *threads*.

Cada aresta executa de forma independente a clusterização *k-means*, que ocorre em duas etapas. A primeira é o cálculo do centróide, em que um elemento de cada amostra é utilizado para encontrar a reta que mais se aproxima delas. Depois é feita a realocação das hipóteses de acordo com sua distância para os centróides. Nessa segunda etapa de ordenação cada amostra pode trabalhar de forma independente, bastando que sejam conhecidos todos os centróides calculados. Por isso é possível separar cada cálculo de realocação das amostras (que envolve cálculo de distâncias e ordenação) em *threads* separadas, assim como foi feito com as arestas. Para isto, existe uma única condição que a informação dos centróides fiquem em algum tipo de memória compartilhada entre as amostras da arestas.

Devido a essa restrição de toda amostra utilizar informações em comum com as outras amostras da aresta, é importante que as *threads* que computam as amostras de uma mesma aresta sejam do mesmo bloco da GPU. Dessa forma, pode-se utilizar a memória compartilhada, a qual compartilha acesso com todas as *threads* e possui um tempo de acesso mais rápido que a memória global, que compartilha acesso com as *threads* de qualquer bloco. O cálculo dos centróides também pode ser paralelizado na GPU, já que cada centróide utiliza informações de apenas um *cluster* para minimizar a reta.

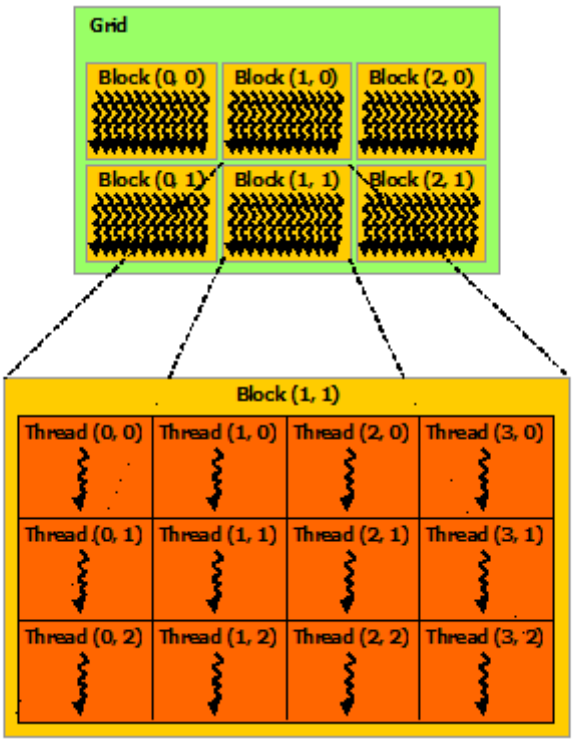



Figura 3.6: Esquema de separação de blocos e *threads* das placas da NVIDIA.

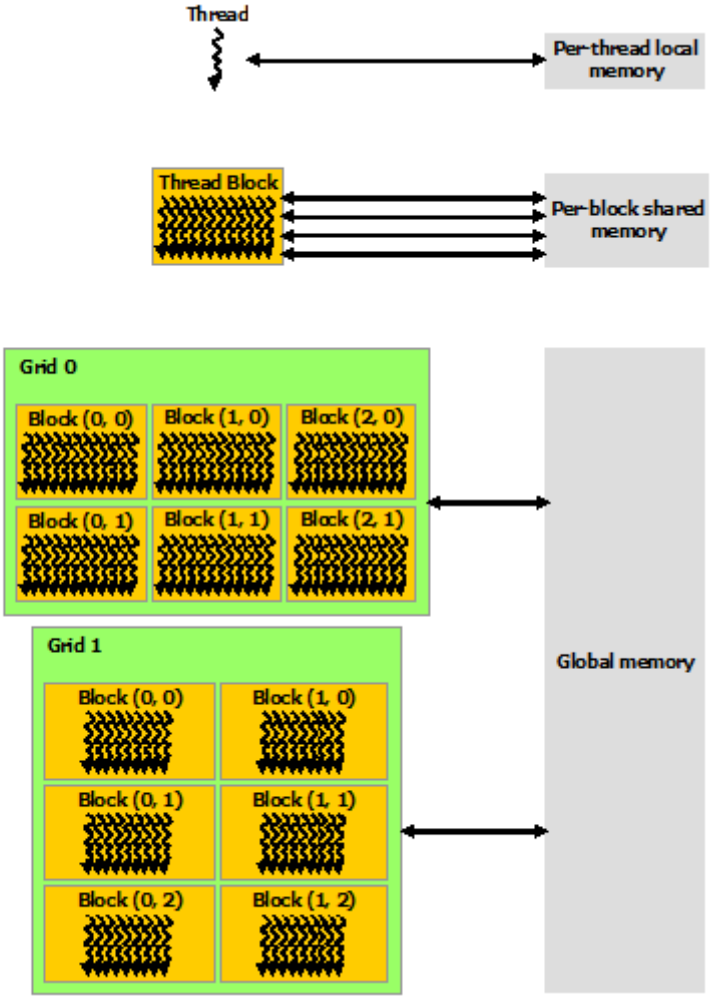



Figura 3.7: Organização de memória das placas da NVIDIA.

CAPÍTULO 4

# Resultados

A técnica desenvolvida neste trabalho de graduação foi avaliada a fim de verificar a qualidade do rastreamento obtido e o tempo de execução da técnica, visando o seu uso em aplicações de realidade aumentada. Os experimentos foram executados em um computador equipado com um processador Core i7-3770 3.4 GHz da Intel com 4GB de memória RAM e placa de vídeo GeForce GTS 450 com suporte a CUDA (versão 5.0). O teste em vídeo foi feito utilizando diferentes filmagens, com diferentes quantidades de quadros. Todas as filmagens tem resolução de 320 por 240 *pixels* e tiveram como objetivo rastrear um cubo parado enquanto se fazia movimentos com a câmera utilizada na filmagem.

Foram feitos dois tipos de teste: um de desempenho, que compara o tempo médio para a execução do algoritmo em cada quadro, e o teste de qualidade do rastreamento, que indica, através de um resultado numérico, o quão próxima a pose encontrada está dos pontos extraídos da imagem.

No teste de desempenho foi medido o tempo, em milésimos de segundos, que o algoritmo leva para executar as etapas que permitem o rastreamento (obtenção dos pontos de forte gradiente, escolha das hipóteses, cálculo da pose). Não foi considerado o tempo de obtenção dos quadros a partir da câmera e de renderização do modelo na tela do computador, pois isso é bastante dependente dos respectivos *drivers* de entrada e saída. Neste teste também foi feita uma comparação da velocidade do algoritmo utilizado nesse trabalho (utilizando tanto a implementação em CPU como a implementação em GPU) com o algoritmo de escolha de hipóteses baseado na distância até a amostra do modelo [11]. O ideal é que cada quadro seja computado em pelo menos 33 milésimos de segundo, o que em termos de quantidade de *frames* por segundo é algo em torno de 30 FPS, que é aproximadamente o limiar humano de percepção de movimento quadro a quadro.

No teste de qualidade foi utilizado como métrica o erro de reprojeção da pose encontrada após a minimização utilizando o algoritmo de Levenberg-Marquardt. Como foi dito anteriormente, o erro de reprojeção é a média das distâncias de cada hipótese de ponto com sua amostra correspondente do modelo utilizando a pose obtida após a minimização. Os resultados do teste de qualidade são independentes da capacidade de processamento da máquina utilizada, portanto também é indiferente se o algoritmo foi executado em GPU ou CPU. No teste aqui apresentado a qualidade do algoritmo será comparada com uma técnica baseada em arestas com múltiplas hipóteses na qual a hipótese mais próxima da amostra sempre é escolhida [11].

## 4.1 Qualidade do rastreamento

Para testar a qualidade do rastreamento o principal ponto a se considerar é o erro de reprojeção da pose. Este tipo de métrica é utilizada em diversos trabalhos relacionados na literatura.

Os dados da Figura 4.1 mostram a variação do erro de reprojeção (eixo vertical) no decorrer do rastreamento de um cubo. A linha vermelha indica o rastreamento utilizando a técnica descrita neste trabalho, enquanto que a linha verde indica um rastreamento baseado em aresta sem a utilização da clusterização *k-means*. A partir do *frame* 350 há um crescimento no erro de reprojeção em ambos os rastreamentos. Isso foi devido a uma movimentação rápida da câmera durante a gravação do experimento. Após esse movimento na câmera o modelo renderizado passou a não corresponder com o objeto rastreado (veja Figura 4.2). O experimento que utilizou a clusterização, no entanto, conseguiu rapidamente chegar a uma pose mais próxima do objeto enquanto que a abordagem tradicional permaneceu distante até o fim do experimento. Isso pode ser evidenciado ao analisar o gráfico no *frame* 600, onde há um pico no erro de reprojeção, e comparar com a Figura 4.3, em que claramente pode-se perceber que que o modelo renderizado pelo *edge-based* está muito distante do objeto rastreado.

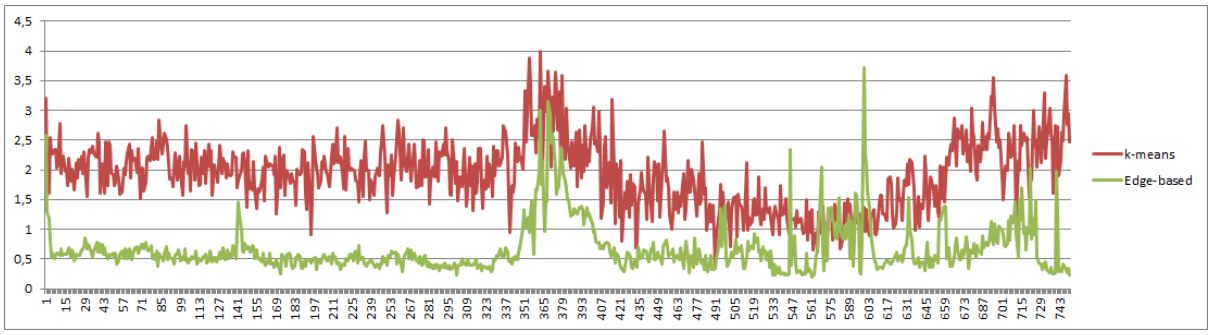



Figura 4.1: Gráfico de qualidade do rastreamento de um cubo.

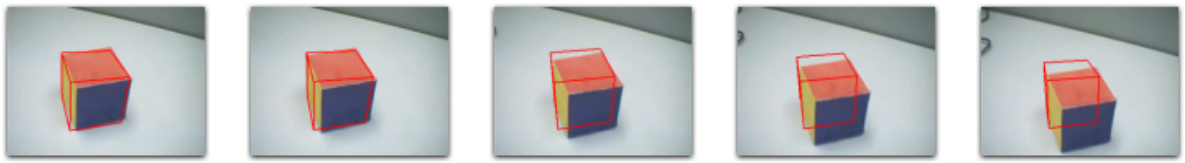



Figura 4.2: Sequência de *frames* mostrando como estava o modelo antes e como ficou depois do *frame* 350.

Foi observado nos testes que apesar das vantagens teóricas para a escolha da melhor hipótese, a técnica discutida apresenta bastante instabilidade na escolha da pose. Mesmo quando a câmera está em repouso, as poses calculadas em dois *frames* consecutivos podem mudar bruscamente. Isso também pode ser visto no gráfico da Figura 4.1 no que diz respeito à instabilidade no erro de reprojeção da linha vermelha. Observe que enquanto a linha verde tem

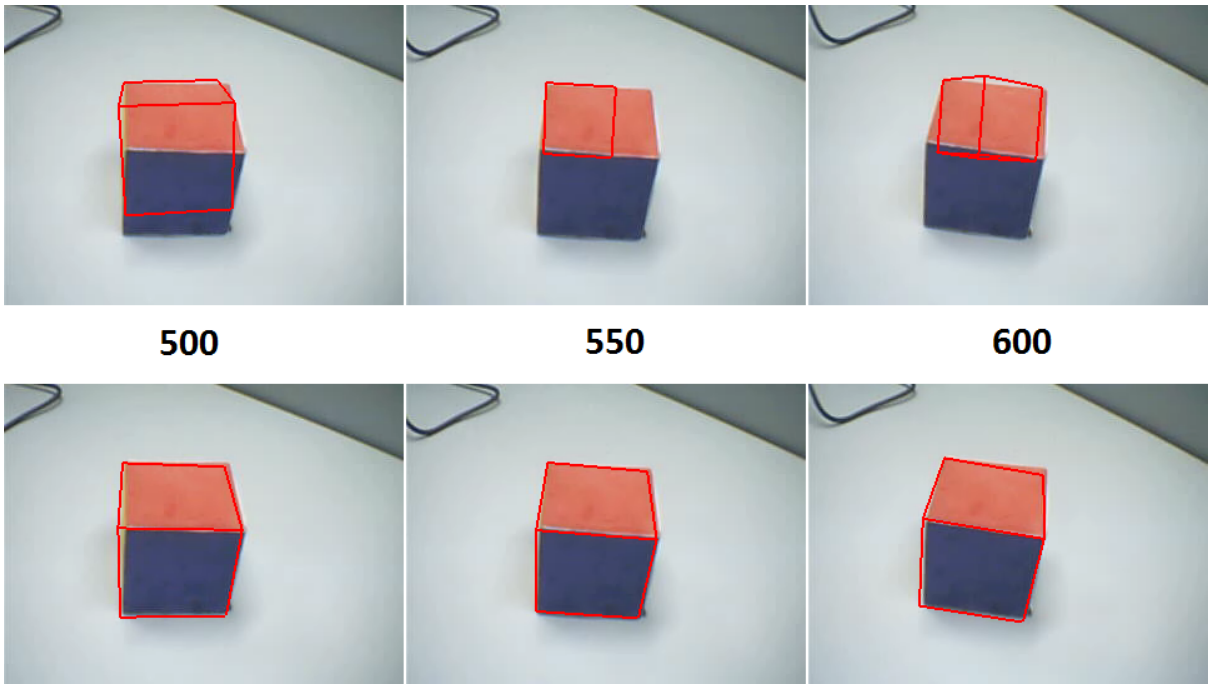



Figura 4.3: Sequência de imagens comparando as duas técnicas de rastreamento. A imagem mostra tanto o cubo a ser rastreado como também o modelo sendo renderizado. O sequência de baixo é resultado do algoritmo que utiliza a clusterização *k-means* e a de cima é resultado do rastreamento baseado em arestas sem a utilização da técnica.

poucas variações no decorrer do tempo, a linha vermelha tem muitas variações, no decorrer de todo o experimento.

Apesar do que foi mostrado no gráfico acima, ao se analisar o rastreamento quadro a quadro, pode ser visto que em determinado momento (a partir do *frame* 500) o rastreamento utilizando a técnica baseada em arestas tradicional calcula uma pose bastante diferente do objeto rastreado, mesmo que o erro de reprojeção continue pequeno. Por outro lado, a técnica desenvolvida neste trabalho apresenta erros de reprojeção maiores, porém a pose se aproxima bastante do objeto. Na Figura 4.3 é possível observar a sequência de *frames* nos dois casos.

Foi notado nos experimentos que o algoritmo não apresenta melhora significativa na qualidade do rastreamento comparado com outras técnicas já existentes. O fato das amostras escolhidas serem as mais colineares possíveis torna o rastreamento bastante instável, pois não muito raro as amostras que formam o conjunto mais colinear se distanciam bastante da aresta do modelo, chegando em alguns casos a uma posição quase que perpendicular a que deveria realmente ser escolhida. O algoritmo encontra muita dificuldade em manter a pose escolhida e é possível perceber isso ao analisar o vídeo. Por outro lado, notou-se que um ponto forte da técnica é a facilidade em encontrar a pose correta quando o modelo está bastante distante do real. Nesses casos ela obteve os melhores resultados. Foi realizado outro experimento em que a câmera era movimentada constantemente com o objetivo de testar se o rastreamento consegue se recuperar com facilidade. Assim como na Figura 4.1, a Figura 4.4 mostra a variação do erro de reprojeção (eixo vertical) no decorrer do tempo (eixo horizontal) e compara a técnica imple-

mentada neste trabalho (linha vermelha) com uma técnica de rastreamento baseado em arestas sem utilizar o *k-means* (linha verde). Com o algoritmo deste trabalho a pose sempre volta a rastrear o objeto, apesar de todas as movimentações; o mesmo não acontece com o algoritmo tradicional. No final do experimento percebe-se que o erro de reprojeção da técnica que utiliza o *k-means* já é menor que o erro do rastreamento tradicional. Neste momento a técnica *edge-based* não conseguiu mais rastrear o objeto enquanto que a abordagem utilizada neste trabalho conseguiu encontrar novamente uma pose próxima ao objeto rastreado.

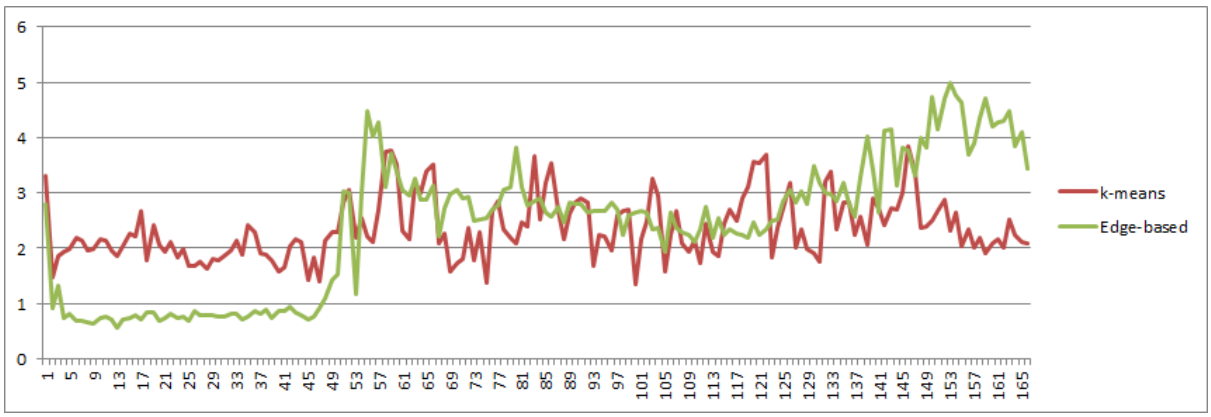



Figura 4.4: Gráfico comparativo de qualidade em um teste em que são feitos vários movimentos com a câmera

## 4.2 Desempenho

No teste de desempenho é avaliado o tempo, em milésimos de segundo, no qual o algoritmo foi executado. Esse tipo de teste depende bastante da máquina utilizada, mas é importante discutilo para se fazer uma comparação relativa entre as técnicas apresentadas. O vídeo de entrada utilizado nos três testes é uma sequência de 751 *frames* de um cubo parado enquanto que a câmera faz alguns movimentos. Foi feito o teste utilizando os algoritmos de múltiplas poses de câmera rodando tanto em GPU quanto em CPU, como também foi utilizado um algoritmo de hipótese única de câmera para efeitos de comparação. A Figura 4.5 mostra um gráfico comparativo de desempenho das três simulações. O eixo horizontal mostra os *frames* do vídeo e o eixo vertical mostra o tempo de execução de cada *frame* em milissegundos. Como é um teste de tempo de execução, quanto menor o valor, melhor o resultado. Deseja-se executar o algoritmo em um tempo menor que 33 milésimos de segundo, por *frame*, para que a execução seja considerada em tempo real.

A Figura 4.5 mostra um gráfico comparativo do desempenho do algoritmo executado tanto em CPU (linha azul) quanto em GPU (linha vermelha). Para efeitos de comparação com técnicas já existentes, também foi feito o mesmo teste com uma abordagem baseada em aresta tradicional (linha verde). O eixo vertical indica o tempo de execução do rastreamento em cada *frame* (eixo horizontal) e a linha preta pontilhada é o limiar para que a execução seja considerada em tempo real. Como essa é uma medida de tempo, quanto menor o valor do eixo vertical,

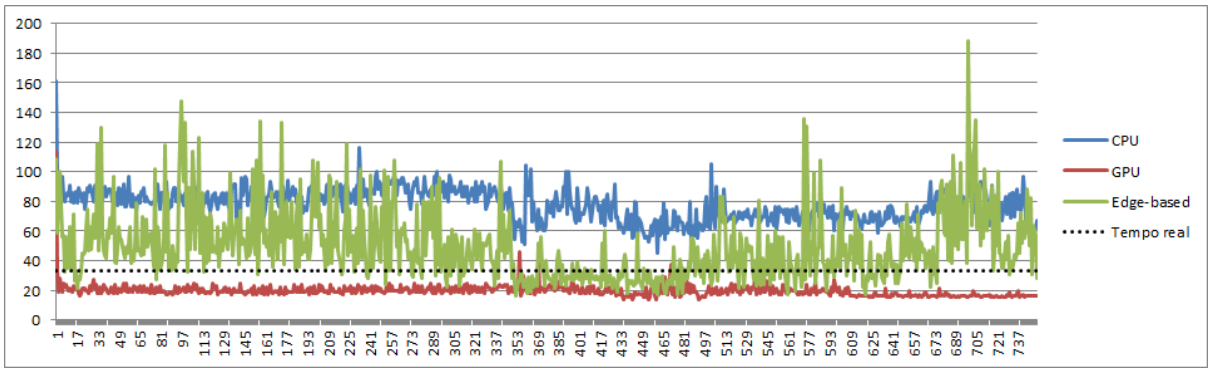



Figura 4.5: Gráfico comparativo da performance do algoritmo em CPU e em GPU. Também é mostrado um rastreamento sem utilizar o algoritmo para efeitos de comparação. A linha pontilhada mostra o limite para que a execução seja considerada em tempo real.

melhor o desempenho. No gráfico pode-se observar que o uso de múltiplas hipóteses de pose juntamente com a clusterização *k-means* aumenta bastante o tempo de execução do algoritmo. Apesar disso, o mesmo algoritmo implementado em CUDA apresenta resultados melhores do que a implementação em CPU da técnica de pose única, sendo inclusive o único cuja execução do experimento pode ser considerada em tempo real, pois praticamente todos os *frames* são executado em menos de 33 milissegundos (linha preta do gráfico).

CAPÍTULO 5

# Conclusão

A técnica de rastreamento 3D utilizando múltiplas hipóteses de pose, apesar de apresentar facilidade ao encontrar a melhor pose quando o modelo projetado se distancia muito do objeto rastreado, mostrou-se bastante instável na maioria dos experimentos, tanto no erro de reprojeção médio quanto na qualidade visual do rastreamento. A proposta de se calcular múltiplas poses para então escolher a melhor é interessante, entretanto a forma como essas hipóteses de pose são obtidas acaba por atrapalhar no resultado final, pois na maioria das vezes nenhuma das poses calculadas é boa se comparada à abordagem tradicional de rastreamento baseado em aresta.

Resultados melhores talvez possam ser encontrados se formas diferentes de se obter as hipóteses de pose forem usadas em conjunto com a baseada em clusterização *k-means*. O trabalho apresentado por Teuliere *et al.* [1] mostra que esta técnica em conjunto com um filtro de partículas apresenta resultados melhores que as técnicas baseadas em aresta tradicionais. Outra abordagem que talvez resulte em resultados melhores é utilizar como uma das hipóteses a pose calculada pela técnica baseada em arestas tradicional, em que as hipóteses de pontos associados à amostra são escolhidos seguindo um critério de distância ao ponto amostrado. Estas direções serão investigadas como trabalhos futuros desta pesquisa.

# Referências Bibliográficas


[1] C. Teuliere, E. Marchand, and L. Eck, "Using multiple hypothesis in model-based tracking," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 4559–4565.

[2] V. Lepetit and P. Fua, "Monocular model-based 3d tracking of rigid objects: A survey," in *Foundations and Trends in Computer Graphics and Vision*, 2005, pp. 1–89.

[3] T. Drummond and R. Cipolla, "Real-time visual tracking of complex structures," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 7, pp. 932 –946, jul 2002.

[4] D. Koller, K. Daniilidis, and H. h. Nagel, "Model-based object tracking in monocular image sequences of road traffic scenes," 1993.

[5] H. Wuest, F. Vial, and D. Strieker, "Adaptive line tracking with multiple hypotheses for augmented reality," in *Mixed and Augmented Reality, 2005. Proceedings. Fourth IEEE and ACM International Symposium on*, 2005, pp. 62–69.

[6] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.

[7] B. Triggs, P. Mclauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment – a modern synthesis," in *Vision Algorithms: Theory and Practice, LNCS*. Springer Verlag, 2000, pp. 298–375.

[8] K. Levenberg, "A method for the solution of certain nonlinear problems in least squares," 1944.

[9] R. Roberto, "Desenvolvimento de sistema de realidade aumentada projetiva com aplicação em educação," Dissertação de Mestrado, Universidade Federal de Pernambuco, 2012.

[10] V. Teichrieb, J. Lima, E. Apolinário, T. Farias, M. Bueno, J. Kelner, and I. Santos, "A survey of online monocular markerless augmented reality," *International Journal of Modeling and Simulation for the Petroleum Industry*, vol. 1, no. 1, pp. 1–7, 2007.

[11] F. Simões, "Realidade aumentada sem marcadores a partir de rastreamento baseado em textura - uma abordagem baseada em pontos de interesse e filtro de partículas," Dissertação de Mestrado, Universidade Federal de Pernambuco, 2011.

[12] J. P. S. d. M. Lima, F. P. M. Simoes, L. S. Figueiredo, and J. Kelner, "Model based markerless 3d tracking applied to augmented reality," *SBC Journal on 3D Interactive Systems*, vol. 1, no. 1, pp. 2–15, 2010.

[13] Autodesk, "3ds official site," http://www.autodesk.com/3dsmax, 2013, acessado em 25 de Abril de 2013.

[14] ——, "Maya official site," http://www.autodesk.com/maya, 2013, acessado em 25 de Abril de 2013.

[15] F. Jurie and M. Dhome, "A simple and efficient template matching algorithm." in *Proc. ICCV*, 2001, pp. 200–1.

[16] E. Marchand, P. Bouthemy, F. Chaumette, and V. Moreau, "Robust real-time visual tracking using a 2d-3d model-based approach," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 1, 1999, pp. 262–268 vol.1.

[17] L. Vacchetti, V. Lepetit, and P. Fua, "Combining edge and texture information for real-time accurate 3d camera tracking," in *Mixed and Augmented Reality, 2004. ISMAR 2004. Third IEEE and ACM International Symposium on*, 2004, pp. 48–56.

[18] J. Hartigan, *Clustering Algorithms*, ser. Wiley Series in Probability and Mathematical Statistics. Books on Demand, 1975.

[19] OpenCV, "Opencv documentation," http://docs.opencv.org/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html#fitline, 2013, acessado em 19 de Abril de 2013.

[20] NVIDIA, "Nvidia cuda site," http://www.nvidia.com/object/cuda_home_new.html, 2013, acessado em 25 de Janeiro de 2013.