

*OPENMOCAP*: UMA APLICAÇÃO DE CÓDIGO  
LIVRE PARA A CAPTURA ÓPTICA DE  
MOVIMENTO

DAVID LUNARDI FLAM

*OPENMOCAP*: UMA APLICAÇÃO DE CÓDIGO  
LIVRE PARA A CAPTURA ÓPTICA DE  
MOVIMENTO

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

ORIENTADOR: ARNALDO DE ALBUQUERQUE ARAÚJO  
CO-ORIENTADOR: JOÃO VICTOR BOECHAT GOMIDE

Belo Horizonte

Julho de 2009

© 2009, David Lunardi Flam.  
Todos os direitos reservados.

Flam, David Lunardi  
F577o *OpenMoCap*: uma aplicação de código livre para a  
captura óptica de movimento / David Lunardi Flam.  
— Belo Horizonte, 2009.  
xiv, 78 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de  
Minas Gerais. Departamento de Ciência da  
Computação.

Orientador: Arnaldo de Albuquerque Araújo.  
Co-orientador: João Victor Boechat Gomide.

1. Captura de Movimento. 2. Rastreamento.  
3. Reconstrução Tridimensional. I. Orientador.  
II. Co-orientador. III. Título.

CDU 519.6\*83.7



UNIVERSIDADE FEDERAL DE MINAS GERAIS  
INSTITUTO DE CIÊNCIAS EXATAS  
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

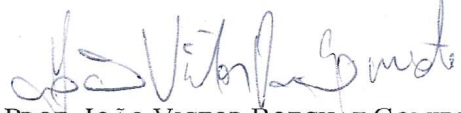
## FOLHA DE APROVAÇÃO

OpenMoCap: Uma Aplicação de Código Livre para a Captura Óptica de Movimento

**DAVID LUNARDI FLAM**

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

  
PROF. ARNALDO DE ALBUQUERQUE ARAÚJO - Orientador  
Departamento de Ciência da Computação - UFMG

  
PROF. JOÃO VICTOR BOECHAT GOMIDE - Co-orientador  
Faculdade de Ciências Empresariais - FUMEC

  
PROF. MARCELO BERNARDES VIEIRA  
Departamento de Ciência da Computação - UFJF

  
PROF. LUIZ CHAIMOWICZ  
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 31 de julho de 2009.

*Aos meus pais, Samuel e Patrícia, ao meu irmão, Efraim, as minhas avós, Chana e Veramuza e minha tia, Andrea.*

# Agradecimentos

A D’us, por ter me dado a oportunidade de concluir mais um ciclo de conhecimento formal em minha vida.

Ao meu orientador, Prof. Arnaldo de Albuquerque Araújo, e ao meu co-orientador, Prof. João Victor Boechat Gomide, por ajudarem a definir foco e direções deste trabalho.

Aos “revisores”, Fernando, Joelma e Monica, pela leitura e sugestões de correção de uma prévia deste texto.

Aos membros da banca, Marcelo Bernardes Vieira e Luiz Chaimowicz, pela revisão técnica minuciosa, sugestões e recomendações.

Ao meu pai, Samuel, pelos conselhos e por ser um exemplo para mim de determinação e conquista de objetivos.

A minha mãe, Patrícia, pelo apoio logístico, incentivos e carinho nos momentos mais críticos do mestrado.

Ao meu irmão, Efraim, pelas infinitas discussões filosóficas e amizade incondicional.

Aos meus familiares, pelas orações.

Aos amigos da “turminha legal”, da Vetta, da grad031, do CSA e da ETH, pelos momentos de diversão e lazer.

Aos amigos do NPDI, pelas ideias, apoio e confraternização.

Aos professores e funcionários do DCC, pela infraestrutura fornecida para completar o mestrado.

Ao CNPq, Capes e FAPEMIG, pelo suporte financeiro.

Finalmente, a todos que contribuíram direta ou indiretamente para a conclusão deste trabalho!

# Resumo

A captura de movimento é, nos dias de hoje, uma técnica valiosa para a animação de personagens virtuais em filmes e jogos digitais graças ao alto grau de realismo que pode ser alcançado. Infelizmente, grande parte dos sistemas disponíveis atualmente para realizar a tarefa têm alto custo e são proprietários. Neste trabalho, uma aplicação de código livre para a captura óptica de movimento é desenvolvida, baseada em técnicas de análise de imagens digitais. As etapas de inicialização, rastreamento, reconstrução e saída são todas realizadas pelo *software* construído, o *OpenMoCap*. A arquitetura definida é apropriada para gravação em tempo real de movimento e é flexível, permitindo a adição de novos módulos otimizados para partes específicas do fluxo de captura, aproveitando as outras partes já existentes. Experimentos com duas câmeras com LEDs infravermelhos e marcadores reflexivos foram realizados e a metodologia criada foi avaliada. Apesar de não possuir a mesma robustez e precisão da solução comercial comparada, este trabalho funciona para animações simples e serve como incentivo para a pesquisa na área.

# Abstract

Nowadays motion capture is a valuable technique for virtual character animation in digital movies and games due to the high degree of realism that can be achieved. Unfortunately, most of the systems currently available to perform that task are expensive and proprietary. In this work, an open source application for optical motion capture is developed based on digital image analysis techniques. The steps of initialization, tracking, reconstruction and output are all accomplished by the built OpenMoCap software. The defined architecture is designed for real time motion recording and it is flexible, allowing the addition of new optimized modules for specific parts of the capture pipeline, taking advantage of the existing ones. Experiments with two cameras with infrared LEDs and reflexive markers were carried out and the created methodology was assessed. Although not having the same robustness and precision of the compared commercial solution, this work can do simple animations and it serves as an incentive for research in the area.



# Lista de Figuras

1.1	Modelo de Câmera <i>Pinhole</i> . . . . .	5
1.2	Sistema Óptico — <i>Impulse</i> (PhaseSpace Inc., 2008). . . . .	8
1.3	Sistema Magnético — <i>Flock of Birds</i> (Ascension Technology Corporation, 2009). . . . .	9
1.4	Sistema Protético — <i>Gypsy 6</i> (Animazoo, 2008). . . . .	10
1.5	Sistema Inercial — <i>IGS 190</i> (Animazoo, 2008). . . . .	11
1.6	Transmissor de Sistema Acústico — <i>Hx11</i> (Hexamite, 2009). . . . .	12
2.1	Sequência de Fotos Obtidas por Muybridge. . . . .	15
2.2	Trabalho de Marey. . . . .	15
3.1	Geometria Câmera Projetiva. . . . .	23
3.2	Semelhança de Triângulos no Plano YZ. . . . .	23
3.3	Sistemas de Coordenadas da Câmera e da Imagem. . . . .	24
3.4	Mudança do Sistemas de Coordenadas da Câmera. . . . .	25
3.5	Exemplo de Correspondência de Pontos em Imagens Estéreo. . . . .	26
3.6	Triangulação e Solução Algébrica para Reconstrução. . . . .	27
3.7	Esquema Geral de Algoritmos Evolutivos. . . . .	29
3.8	Exemplo de Limiarização Binária. . . . .	31
3.9	Conectividade de <i>Pixels</i> . . . . .	32
3.10	Dilema da Conectividade. . . . .	33
3.11	Fluxograma do Algoritmo de Rotulação com Adjacência de Seis <i>Pixels</i> (Umbaugh, 2005). . . . .	33
3.12	Tipos de Dados Gerados por Sistemas de Captura de Movimento. . . . .	36
4.1	Fluxograma da Captura de Movimento no <i>OpenMoCap</i> . . . . .	38
4.2	<i>Hardware</i> Utilizado da <i>NaturalPoint</i> . . . . .	39
4.3	Comparação Fotografia e Imagem Obtida pela Câmera <i>OptiTrack FLEX:V100</i> . . . . .	39

4.4	Detecção de POIs. . . . .	41
4.5	Sugestão de Posicionamento de Marcadores e Representação Hierárquica em BVH. . . . .	42
4.6	Seleção de Semântica de um POI. . . . .	43
4.7	Interface Gráfica para Determinação de Parâmetros das Câmeras. . . . .	44
4.8	Exemplo do Rastreamento. . . . .	46
4.9	Exemplo de Arquivo Gerado no Formato TRC. . . . .	48
4.10	Diagrama da Arquitetura do <i>OpenMoCap</i> . . . . .	49
4.11	Interface Gráfica Principal do <i>OpenMoCap</i> . . . . .	50
5.1	Gráficos das Coordenada X dos Centróides Obtidos pelo <i>OpenMoCap</i> e pelo <i>Tracking Tools</i> . . . . .	59
5.2	Comparação de Trajetórias do <i>Tracking Tools</i> e <i>OpenMoCap</i> . . . . .	61
5.3	Cena Capturada com Estrutura de Marcadores. . . . .	63
5.4	Estrutura Tridimensional Gerada pelo <i>Tracking Tools</i> e <i>OpenMoCap</i> . . . . .	64
5.5	Primeiro Quadro Exemplo de Captura de Movimento pelo <i>OpenMoCap</i> . . . . .	66
5.6	Segundo Quadro Exemplo de Captura de Movimento pelo <i>OpenMoCap</i> . . . . .	66
5.7	Terceiro Quadro Exemplo de Captura de Movimento pelo <i>OpenMoCap</i> . . . . .	67
5.8	Gráfico Área do Tempo de Processamento Total do <i>OpenMoCap</i> . . . . .	68
5.9	Gráfico Caixa do Tempo de Processamento de Detecção de POIs pelo Número de Câmeras <i>OpenMoCap</i> . . . . .	69

# Lista de Tabelas

5.1	Configuração do Computador Utilizado nos Experimentos. . . . .	57
5.2	Configuração para Detecção de POIs. . . . .	58
5.3	Análise Estatística Comparativa de Detecção de Centróides 2D. . . . .	59
5.4	Parâmetros Encontrados no Ajuste de Curva da Trajetória do Pêndulo no <i>Tracking Tools</i> . . . . .	60
5.5	Deslocamento Entre os Centróides Obtidos pelo <i>OpenMoCap</i> e a Trajetória Ajustada do Pêndulo pela Solução Comercial. . . . .	62
5.6	Distâncias entre Pontos da Estrutura Obtida pelo <i>Tracking Tools</i> em metros. . . . .	64
5.7	Distâncias entre Pontos da Estrutura Obtida pelo <i>OpenMoCap</i> . . . . .	65
5.8	Fator Multiplicativo entre Distâncias <i>OpenMoCap</i> e <i>Tracking Tools</i> . . . . .	65

# Sumário

Agradecimentos	vi
Resumo	vii
Abstract	viii
Lista de Figuras	ix
Lista de Tabelas	xi
<b>1 Introdução</b>	<b>1</b>
1.1 Motivação . . . . .	2
1.2 Objetivos . . . . .	3
1.2.1 Objetivo Geral . . . . .	3
1.2.2 Objetivos Específicos . . . . .	3
1.3 Conceitos Preliminares . . . . .	4
1.3.1 Câmera Digital . . . . .	4
1.3.2 Marcadores . . . . .	5
1.3.3 Aquisição de Dados de Captura . . . . .	6
1.3.4 Sistemas de Captura de Movimento . . . . .	7
1.3.5 Sensores e Fontes . . . . .	12
1.4 Estrutura da Dissertação . . . . .	13
<b>2 Trabalhos Relacionados</b>	<b>14</b>
2.1 Início da Captura de Movimento . . . . .	14
2.2 Era Digital . . . . .	16
2.3 Estado da Arte . . . . .	17
2.3.1 Sistemas Comerciais . . . . .	17
2.3.2 Trabalhos Recentes . . . . .	18

2.4	Considerações . . . . .	20
<b>3</b>	<b>Fundamentos Teóricos</b>	<b>22</b>
3.1	Modelo de Câmera . . . . .	22
3.2	Reconstrução . . . . .	26
3.2.1	Decomposição em Valores Singulares . . . . .	28
3.3	Evolução Diferencial . . . . .	29
3.4	Limiarização . . . . .	30
3.5	Componentes Conectados . . . . .	32
3.5.1	Propriedades de um Componente Conectado . . . . .	34
3.6	Estimador Alfa-Beta . . . . .	34
3.7	Dados de Captura de Movimento . . . . .	35
3.8	Considerações . . . . .	36
<b>4</b>	<b>Metodologia</b>	<b>37</b>
4.1	Fluxo de Captura de Movimento . . . . .	37
4.2	Extração de POIs . . . . .	39
4.3	Seleção de Semântica dos POIs . . . . .	41
4.4	Determinação dos Parâmetros das Câmeras . . . . .	42
4.5	Rastreamento . . . . .	45
4.6	Triangulação . . . . .	46
4.7	Geração de Arquivo de Saída . . . . .	47
4.8	Parâmetros do <i>OpenMoCap</i> . . . . .	48
4.9	Arquitetura da Aplicação <i>OpenMoCap</i> . . . . .	49
4.9.1	Pacotes . . . . .	52
4.10	Considerações . . . . .	54
<b>5</b>	<b>Resultados Experimentais</b>	<b>55</b>
5.1	<i>NaturalPoint</i> . . . . .	55
5.1.1	<i>NaturalPoint Tracking Tools</i> . . . . .	56
5.2	<i>Hardware</i> Comum . . . . .	57
5.2.1	Câmeras <i>NaturalPoint OptiTrack FLEX:V100</i> . . . . .	57
5.2.2	Computador . . . . .	57
5.3	Experimentos e Resultados . . . . .	58
5.3.1	Estabilidade e Precisão dos Centróides 2D . . . . .	58
5.3.2	Rastreamento 2D . . . . .	60
5.3.3	Estimação de Parâmetros de Câmeras . . . . .	62
5.3.4	Estrutura 3D . . . . .	62

5.3.5	Saída e Movimento 3D . . . . .	66
5.3.6	Tempo de Processamento . . . . .	67
5.4	Considerações . . . . .	70
<b>6</b>	<b>Conclusões</b>	<b>71</b>
6.1	Objetivos Alcançados . . . . .	71
6.2	Trabalhos Futuros . . . . .	72
	<b>Referências Bibliográficas</b>	<b>74</b>

# Capítulo 1

## Introdução

A análise e captura de movimento humano vêm crescendo constantemente desde a última década. Pesquisadores e profissionais têm encontrado diversas aplicações na indústria de entretenimento, na medicina, nos esportes e em segurança. Uma classificação apresentada em Moeslund & Granum (2001) e Moeslund et al. (2006) as agrupa em três grandes categorias: *Controle*, *Análise* e *Vigilância*.

Aplicações de *Controle* interpretam o movimento de um ator e o transformam em uma sequência de operações. Filmes como a trilogia *Lord of the Rings* (2001 - 2003), *Polar Express* (2004), *King Kong* (2005) e o mais recente *Beowulf* (2007) usam essas operações geradas para animar personagens. Jogos de computador, como *FX Fighter* e as séries *Fifa Soccer* e *NBA Live* também fazem o mesmo uso da técnica.

Aplicações de *Análise* geralmente se dedicam ao estudo de uma pessoa como um conjunto de objetos. Alguns exemplos incluem aprimoramento de técnicas de atletas, estudo de casos clínicos e definição do comportamento do corpo de um idoso.

Aplicações de *Vigilância* focam em examinar movimento de uma pessoa como um objeto único ou um grupo de objetos no caso de uma aglomeração de pessoas. Determinar o fluxo de pessoas em uma galeria para descobrir um padrão a fim de otimizar a localização das lojas é um exemplo. Outro é criar um modelo que descreve o comportamento de pessoas em uma prisão. Caso uma anomalia seja detectada, ações preventivas podem ser tomadas.

De acordo com Menache (2000), a captura de movimento (do inglês *Motion Capture* — MoCap) é o processo que permite traduzir uma atuação ao vivo em uma atuação digital. Ou, como definido (tradução livre) por Sturman (1994):

Captura de movimento é a gravação de movimento do corpo humano (ou outro movimento) para análise imediata ou postergada e reprodução. A in-

formação capturada pode ser geral, como uma simples posição do corpo em um espaço, ou complexa, como deformações da face e massas musculares.

Em geral, capturam-se separadamente os movimentos corporais dos movimentos faciais, devido às características e sutilezas desses últimos.

A captura de movimento pode ser feita por meio de sensores e a seleção de alguns pontos de interesse (do inglês *Points of Interest* — POIs). Um sistema que realiza tal tarefa é complexo, mas sua essência pode ser descrita por quatro processos, baseado em Moeslund & Granum (2001).

O primeiro é chamado *Inicialização*. Nele, deve-se garantir que o sistema de captura comece em um estado correto de interpretação da cena. Os sensores precisam estar calibrados e suas informações de posicionamento e correção de distorções devem ser conhecidas. Além disso, os POIs necessitam ter suas coordenadas bidimensionais determinadas e suas semânticas atribuídas. Essas semânticas são relações entre os pontos da cena imageada e partes de estrutura previamente definida, como as juntas do corpo humano.

O *Rastreamento* é o segundo processo e envolve a segmentação e o monitoramento no tempo dos POIs selecionados no primeiro processo. A cada quadro, a relação semântica entre os pontos selecionados e o modelo escolhido não deve ser perdida.

O terceiro processo é a *Estimação de Pose*. Seu objetivo é agregar os dados (coordenadas e semânticas) colhidos pelos sensores dos POIs e transformá-los em uma representação no tempo do modelo escolhido. Caso ele seja tridimensional, algoritmos de triangulação são usados para determinar a profundidade de cada ponto.

O último é o *Reconhecimento*, que consiste em definir um formato de saída para os dados capturados e opcionalmente a tomada de algum tipo de ação a partir deles. Portanto, as informações obtidas em uma sessão de captura podem alimentar um *software* de animação ou um aplicativo específico que reconheça gestos ou movimentos e que execute, a partir deles, comandos previamente definidos.

## 1.1 Motivação

A captura de movimento é, nos dias de hoje, uma ferramenta valiosa para a animação de personagens virtuais em filmes e jogos digitais graças ao alto grau de realismo e suavidade que podem ser alcançados. Atualmente, no Brasil, o *know-how* da técnica é restrito e apenas a Rede Globo de Televisão, no Rio de Janeiro, e a RPM Produtora (2008), de São Paulo, possuem equipamentos robustos de captura de movimento voltados para a animação de personagens, ambos importados e de custo elevado.



Esta dissertação é o primeiro passo de um projeto em desenvolvimento no NPDI (Núcleo de Processamento Digital de Imagens), com apoio da FAPEMIG (Fundação de Amparo à Pesquisa do Estado de Minas Gerais) e do CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), de construção de um sistema de captura de movimento robusto, que seja capaz de atender às demandas de geração de bancos de dados de movimento para o audiovisual e os jogos digitais. Um sistema nacional de custo reduzido pode aumentar a utilização da técnica em diversos projetos e áreas, beneficiando inclusive novas pesquisas.

## 1.2 Objetivos

### 1.2.1 Objetivo Geral

Desenvolver uma aplicação de código livre e arquitetura flexível para a implementação de soluções ópticas de aquisição de dados em tempo real para captura de movimento. Seus componentes devem ser modulares e intercambiáveis, realizando cada etapa existente no fluxo de obtenção de dados independentemente. Finalmente, uma comparação com um sistema comercial típico deve ser realizada com a finalidade de determinar a qualidade do sistema construído. Espera-se, desse modo, reduzir o custo total da utilização da técnica de captura de movimento, assim como facilitar e fomentar pesquisas relacionadas.

### 1.2.2 Objetivos Específicos

- Definir uma arquitetura flexível para uma plataforma de aquisição de dados de captura de movimento em tempo real.
- Desenvolver uma metodologia para captura de movimento adaptando ou criando algoritmos para as etapas de inicialização, detecção de pontos de interesse, rastreamento, reconstrução tridimensional e geração de saída.
- Construir uma interface gráfica de fácil uso.
- Comparar dados de saída obtidos pela aplicação desenvolvida com dados produzidos por uma aplicação comercial típica.

## 1.3 Conceitos Preliminares

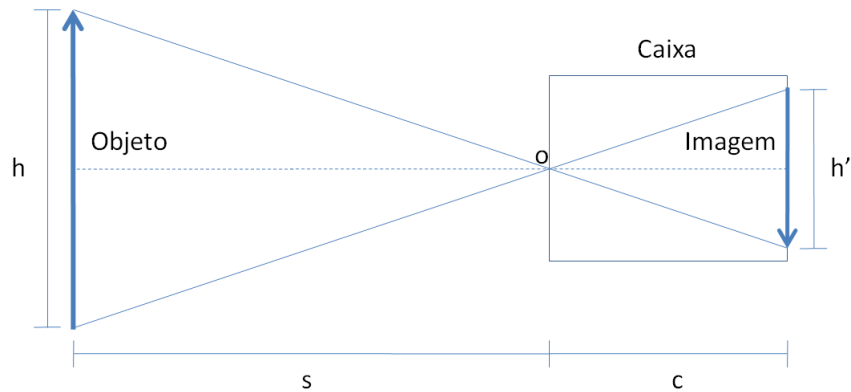
### 1.3.1 Câmera Digital

Uma câmera digital é um aparelho capaz de obter e armazenar uma representação digital de uma cena do mundo real em um determinado instante de tempo. Essa representação é gerada pela interação de energia luminosa com dispositivos capazes de mensurá-la. Esses dispositivos, chamados de sensores, são geralmente formados por um conjunto de células organizadas em formato matricial, que são responsáveis por transformar uma medida de energia em um elemento de uma figura, ou um *pixel* de uma imagem digital. Diversos tipos de imagem podem ser geradas (Umbaugh, 2005) dependendo da sensibilidade dos sensores, os mais comuns são sensíveis ao segmento visível do espectro eletromagnético, mas também podem ser sensíveis a faixas fora desse segmento, como infravermelho, ultravioleta. É possível ainda produzir imagens a partir de ondas sonoras, elétrons e lasers.

De acordo com Steger et al. (2008), dois tipos de sensores são mais comumente encontrados em câmeras: o semicondutor metal-óxido complementar (do inglês *Complementary metal-oxide-semiconductor* — CMOS) e o dispositivo de carga acoplado (do inglês *Charge-coupled device* — CCD). A principal diferença entre os dois está no modo de leitura dos *pixels*. No CMOS, a intensidade de cada *pixel* (carga elétrica) é lida diretamente na célula em que foi captado, enquanto no CCD, a carga contida na célula do sensor precisa ser transportada até uma das laterais do dispositivo para ser lida. Isso faz com que o processo de fabricação do CCD se torne mais elaborado, já que é necessário mover uma carga até a borda do sensor amenizando ao máximo a interferência gerada. Portanto, o custo de produção de um sensor CMOS é mais baixo, porém está mais suscetível a ruído.

O modelo de câmera *pinhole*, exibido pela Figura 1.1, explica de maneira simplificada o processo de obtenção de uma imagem em uma câmera. Na figura, a câmera é uma caixa escura com um pequeno orifício que deixa passar alguns raios de luz. Desconsiderando a natureza dual da luz, tratando-a somente como partícula, os raios de luz propagam em linha reta, passando pelo centro de projeção  $o$  (orifício da caixa) formando a imagem invertida do objeto em um plano do lado oposto. A distância do objeto  $s$  e o comprimento da caixa  $c$  influenciam na altura  $h'$  da imagem formada. Por semelhança de triângulos, a Equação 1.1 quantifica essa influência. Quanto maior a distância do objeto, menor é sua imagem e quanto maior for a distância entre o orifício da caixa e seu fundo, maior será a imagem formada.

$$h' = h \frac{c}{s} \quad (1.1)$$



**Figura 1.1.** Modelo de Câmera *Pinhole*.

Infelizmente, no mundo real uma câmera *pinhole* não é prática. A principal razão é a pequena quantidade de luz que passa pelo seu diminuto orifício, insuficiente para formar uma imagem de luminosidade adequada em um curto espaço de tempo. Logo, o tempo de exposição deve ser extremamente elevado para se obter uma imagem de qualidade. Certas aplicações se tornam impossíveis, como fotos de objetos em movimento ou vídeos. Porém, esse problema pode ser solucionado com o uso de lentes.

Lentes são objetos feitos geralmente de vidro ou plástico, capazes de divergir ou convergir raios de luz. Elas funcionam com base no conceito de refração, em que um raio de luz muda de velocidade ao entrar em um meio diferente do que está. O uso de uma lente convergente torna possível concentrar uma quantidade de luz suficiente para a obtenção de uma imagem em um curto espaço de tempo. Idealmente, os raios de luz de um ponto do objeto devem convergir para um mesmo ponto de foco ao passar pela lente, mas isso não acontece precisamente na realidade. Algumas aberrações e distorções podem surgir por causa desse fato e deterioram a qualidade da imagem final (Mahajan, 1998).

### 1.3.2 Marcadores

A captura de movimento pode ser feita usando ou não marcadores. Marcadores são objetos que auxiliam na determinação ou efetivamente determinam os pontos a serem gravados ou analisados em uma seção de captura. É possível classificá-los em *passivos* ou *ativos*, dependendo de sua reatividade com o mundo externo.

Marcadores passivos são aqueles que apenas refletem algum tipo de onda devido a uma propriedade natural do material que são constituídos. Eles não realizam qualquer tipo de comportamento específico baseado no impulso externo recebido. Ou seja, somente auxiliam na segmentação, na separação de uma região de interesse do fundo. O ponto de interesse em si é obtido por meio do cálculo do centróide da região por um *software*. É necessário ainda definir sua semântica em uma etapa de inicialização e rastreá-lo precisamente no tempo. Um exemplo bastante difundido desta categoria são bolinhas de borracha revestidas de um material reflexivo iluminadas por lâmpadas que emitem luz no comprimento de onda infravermelho (Vicon Motion Systems, 2008).

Marcadores ativos são aqueles que realizam algum tipo de processamento sobre impulsos externos recebidos ou que emitem impulsos. Necessariamente possuem algum tipo de dispositivo como acelerômetro, potenciômetro, giroscópio, sensor acústico, sensor de luz ou diodos emissores de luz (do inglês *Light Emitting Diodes* — LEDs) para poderem realizar cálculos e responder adequadamente ao estímulo recebido. No mínimo, são capazes de se identificar, como o sistema produzido pela PhaseSpace Inc. (2008), que modula singularmente a frequência de luz emitida por cada LED. Portanto, não requerem uma longa etapa de inicialização e o rastreamento é bastante simplificado. Além disso, marcadores ativos podem ser capazes de determinar sua posição tridimensional e rotações sofridas ao longo do tempo (Raskar et al., 2007).

Existem diversos tamanhos, tecnologias e modelos de marcadores. Entretanto, a entidade alvo da captura limita muitas vezes a escolha do tipo de marcador utilizado. Isso ocorre especialmente quando se deseja capturar o movimento detalhado de uma área relativamente pequena, como uma face. Atualmente, somente marcadores passivos podem ser usados nesse caso, tendo em vista seu tamanho e peso reduzidos. Outro fator restritivo é o ambiente em que a captura ocorre. Dependendo das condições existentes, como luz natural, interferências e meio físico, marcadores ativos com giroscópios e acelerômetros podem ser mais adequados. Ainda outra limitação é em relação ao tipo do movimento capturado. Movimentos que envolvam quedas são mais adequados para marcadores passivos, pois não possuem frágeis componentes eletrônicos.

### 1.3.3 Aquisição de Dados de Captura

Segundo da Silva (1998), dois são os modos de se adquirir dados em uma captura de movimento: o *direto* e o *indireto*. Aquisição direta significa não necessitar de uma etapa de pós-processamento para realizar cálculos adicionais sobre os dados gerados. Ou seja, dados adquiridos desse modo podem ser usados exatamente da mesma forma que foram capturados, sem qualquer intervenção. Sistemas desse tipo são próprios para

aplicações de tempo real, apesar de serem mais intrusivos.

Dados adquiridos indiretamente precisam ser tratados por um *software* que calcula informações de rotação e de posição relativa dos pontos de interesse. Somente assim, os dados podem ser usados. As vantagens existentes em sistemas que capturam dados desse modo são, normalmente, a precisão e a velocidade de captura, já que parte das informações necessárias pode ser obtida por meio de processamento posterior.

### 1.3.4 Sistemas de Captura de Movimento

De acordo com Gomide (2006), os sistemas de captura de movimento podem ser divididos em três classes, dependendo do princípio físico empregado, *sistemas ópticos*, *sistemas magnéticos* e *sistemas mecânicos*. Em Furniss (1999), Menache (2000) e Liverman (2004), os autores ainda aumentam a granularidade da classe dos sistemas mecânicos por meio da tecnologia utilizada em *sistemas protéticos*, *sistemas inerciais* e *sistemas acústicos*. Fundamentalmente, todos eles se diferenciam pelo tipo de dado gerado, pelas restrições associadas e pelo custo total. A seguir, uma descrição com exemplos de soluções reais é feita para cada classe.

**Sistemas Ópticos:** Usualmente, consistem em um conjunto de duas ou mais câmeras digitais ligadas a um computador que as sincroniza e processa os quadros de vídeo gerados. Caso marcadores sejam utilizados, eles são presos diretamente na pele ou em uma roupa escura vestida pelo ator alvo da captura. No caso de marcadores passivos, as câmeras são construídas ou anexadas com LEDs, que emitem energia em um comprimento de onda refletido pelos marcadores. O inverso geralmente ocorre quando são utilizados marcadores ativos, os LEDs fazem parte e são controlados pelos próprios marcadores. Finalmente, se marcadores não forem usados, os pontos de interesse podem ser determinados por meio de detecção de silhueta ou componentes estruturais (cabeça, ombro, perna, mãos e etc.). Uma técnica recente simplifica esse processo por meio de uma maquiagem feita por tintas fosforescentes, usada no produto *Mova Contour Reality Capture* (Mova LLC, 2008).

Os dados gerados por sistemas ópticos são normalmente adquiridos de maneira indireta. Isso alivia a pressão computacional no momento da captura, o que permite uma amostragem de gravação suficiente para movimentos extremamente rápidos, como uma tacada de golfe. Outra vantagem é a grande liberdade de movimento provida por esse tipo de sistema, tendo em vista que não existem cabos presos ao corpo do ator e o volume de captura depende diretamente do

número de câmeras utilizadas. Pelas mesmas razões, é possível também capturar simultaneamente o movimento de mais de um indivíduo.

Uma restrição encontrada em sistemas ópticos é a necessidade de se controlar a iluminação ambiente, especialmente quando marcadores passivos são utilizados. Porém, alguns sistemas mais modernos baseados em marcadores ativos já conseguem realizar a captura em luz natural, fora de ambientes especialmente preparados. Ainda assim, existe o problema da oclusão, em que pelo menos duas câmeras devem visualizar um marcador para ser possível definir sua posição tridimensional por meio de uma triangulação (Hartley & Sturm, 1997). Habitualmente, os dados gerados precisam ser limpos por causa de ruído (informação incompleta, imprecisa ou não desejada) gerado pelo processo de rastreamento em situações de oclusão.

Outro ponto fraco em sistemas ópticos é o custo elevado, resultante da alta tecnologia usada na produção das câmeras. As principais variáveis na determinação desse custo são o número de câmeras, a resolução e a quantidade processada de quadros por segundo (do inglês *frames per second* — FPS). Um sistema básico da *PhaseSpace* (ver Figura 1.2) com 10 câmeras, recomendado para captura de uma pessoa, capaz de gravar movimento a 480 FPS e com resolução de 12,4 *megapixels* custa aproximadamente 56 mil dólares<sup>1</sup>. Esse valor sobe para aproximadamente 124 mil dólares<sup>1</sup> quando 24 câmeras são utilizadas.



**Figura 1.2.** Sistema Óptico — *Impulse* (PhaseSpace Inc., 2008).

---

<sup>1</sup>Preço sugerido sem impostos e sem frete, retirado de Inition (2008).

**Sistemas Magnéticos:** Usam marcadores ativos presos nas articulações de uma roupa vestida por um ator. Eles são capazes de medir o fluxo magnético entre bobinas presentes no próprio marcador e bobinas estacionárias em posições conhecidas. Assim, é possível determinar a translação e orientação de cada marcador. É, portanto, um tipo de sistema de aquisição direta.

A liberdade de movimento em sistemas magnéticos é menor se comparada aos sistemas ópticos. Necessariamente, os marcadores utilizados possuem baterias ou fios que limitam o movimento do ator e o volume de captura. Além disso, os dados produzidos podem ser corrompidos e distorcidos devido a interferências elétricas ou magnéticas. Ou seja, existem também restrições em relação ao local de gravação. Preferencialmente, o ambiente escolhido não deve possuir objetos metálicos e aparelhos eletrônicos sem blindagem eletromagnética.

Uma vantagem de sistemas magnéticos é a ausência de oclusão e o custo reduzido em relação aos sistemas ópticos. Um exemplo é o sistema da *Ascension*, o *Flock of Birds* da Figura 1.3, com 12 sensores, capaz de gravar movimento a 144Hz e adequado para a captura de movimento humano. Pode ser comprado por aproximadamente 25 mil dólares<sup>2</sup>.



**Figura 1.3.** Sistema Magnético — *Flock of Birds* (Ascension Technology Corporation, 2009).

---

<sup>2</sup>Preço sugerido sem impostos e sem frete, retirado de Inition (2008).

**Sistemas Protéticos:** Tipicamente, os atores alvo da captura devem usar um exoesqueleto formado por um conjunto de bastões rígidos e potenciômetros em suas extremidades. Conhecendo a configuração inicial dos sensores e usando as informações de ângulos relativos obtidas pelos mesmos, é possível determinar a posição e orientação relativas de cada articulação. Portanto, a aquisição de dados é direta.

Uma desvantagem de sistemas desse tipo é que, sozinhos, não são capazes de produzir informação de translação global dos pontos de interesse. Isso impossibilita capturar movimentos que envolvam saltos ou que necessitem de um referencial externo. Porém, sensores magnéticos são comumente combinados aos potenciômetros para superar esse obstáculo. Outro fator restritivo é o próprio exoesqueleto, que pode se quebrar. Além disso, ele não oferece a mesma liberdade de movimento de outros sistemas de captura. Por exemplo, não é possível gravar movimento de uma pessoa rolando no chão.

Sistemas protéticos não sofrem de problemas como interferência elétrica ou magnética e oclusão. São geralmente portáteis e possuem um custo semelhante ao de sistemas magnéticos, relativamente mais acessíveis quando comparados aos ópticos. O sistema *Animazoo Gypsy 6* mostrado na Figura 1.4 é indicado para captura de um indivíduo, amostra movimento a uma taxa máxima de 120Hz e custa 30 mil dólares<sup>3</sup>.



**Figura 1.4.** Sistema Protético — *Gypsy 6* (Animazoo, 2008).

---

<sup>3</sup>Preço sugerido sem impostos e sem frete, retirado de Inition (2008).



**Sistemas Inerciais:** Marcadores ativos contendo giroscópios e acelerômetros são posicionados estrategicamente nas articulações de uma roupa vestida pela pessoa alvo da captura. Após uma etapa de calibração, é possível usar as informações de rotação dos giroscópios e de posicionamento dos acelerômetros para aquisição direta dos dados.

Sistemas inerciais são portáteis e quase não possuem restrições com relação ao local de captura. Alguns até permitem obter dados em ambientes subaquáticos (Animazoo, 2008). Além disso, não estão sujeitos a efeitos de oclusão e interferência nos dados adquiridos. Outra vantagem é que grande parte deles usa tecnologia de transmissão de dados sem fio. Isso garante boa liberdade de movimento e um volume extenso de gravação. Porém, é preciso usar baterias, o que limita o tempo de captura sem recarga ou implica na necessidade de possuir baterias reservas.

O custo de sistemas inerciais é elevado, comparável aos melhores sistemas ópticos. A principal razão é o nível de tecnologia empregado na miniaturização dos sensores. Portanto, o valor final de um sistema desse tipo depende essencialmente do número de marcadores utilizados. Um exemplo é o sistema *Animazoo IGS 190* (ver Figura 1.5) que possui 18 sensores e captura movimento a uma taxa máxima de 120Hz. É ideal para a captura de corpo inteiro de um indivíduo e seu preço é de 75 mil dólares<sup>4</sup>. É possível também comprar somente a parte superior ou inferior do torso, custando aproximadamente 52 e 43 mil dólares<sup>4</sup>.



**Figura 1.5.** Sistema Inercial — *IGS 190* (Animazoo, 2008).

---

<sup>4</sup>Preço sugerido sem impostos e sem frete, retirado de Inition (2008).

**Sistemas Acústicos:** Um grupo de emissores de som é preso nas articulações de uma roupa vestida pelo ator alvo da captura. Cada emissor faz parte de um marcador ativo que emite um som de alta frequência (ultra-sônico) único. A área de captura pode ser grande e tem seus limites definidos pelas posições de microfones. Pelo menos três deles devem captar um mesmo som para determinar a posição tridimensional do marcador por meio de trilateração. Portanto, o modo de aquisição é indireto, as rotações precisam ser calculadas com base nos dados posicionais.

Sistemas acústicos praticamente não sofrem de oclusão ou interferência elétrica ou magnética, mas são susceptíveis a efeitos de reverberação e reflexão de ondas sonoras. A liberdade de movimento é limitada por cabos ou baterias em sistemas sem fio. Outra desvantagem é o número restrito de marcadores simultâneos suportados, insuficiente para gravação de movimento preciso de um ser humano. Nos dias de hoje, sistemas puramente acústicos são desenvolvidos para a área de vigilância, tendo em vista que podem monitorar grandes áreas e um numeroso conjunto de marcadores não-simultâneos. O produto *Hexamite Hx11* (ver Figura 1.6) é constituído por monitores capazes de determinar até 50 posições por segundo de transmissores em um ambiente fechado. Seu custo estimado para cobrir uma área de 10.000 metros quadrados é de 120 mil dólares<sup>5</sup>, excluindo o preço dos transmissores.



**Figura 1.6.** Transmissor de Sistema Acústico — *Hx11* (Hexamite, 2009).

### 1.3.5 Sensores e Fontes

Uma taxonomia importante proposta por Mulder (1994) e também citada em da Silva (1998) divide os sistemas de captura de movimento em três tipos: *Inside-In*, *Inside-Out* e *Outside-In*. A classificação é baseada na localização dos sensores e fontes, respectiva-

---

<sup>5</sup>Preço sugerido sem impostos e sem frete, retirado de Inition (2008).

mente. A localização somente possui duas variações, definida pelas expressões “dentro” ou “fora”.

Em sistemas mecânicos, os sensores e fontes fazem parte do exoesqueleto preso ao corpo do ator. Os potenciômetros (sensores) determinam o ângulo (fonte) entre duas barras rígidas, representando o grau de inclinação de uma articulação em uma determinada direção. Portanto, inserem-se no grupo *Inside-In*, os sensores e fontes estão “dentro”.

Sistemas magnéticos, inerciais e alguns acústicos são elementos do conjunto *Inside-Out*. Sensores posicionados no alvo da captura recebem estímulos de fontes externas naturais ou artificiais. Em um sistema inercial, um acelerômetro (sensor) movimenta-se no campo gravitacional terrestre (fonte natural) para obter informações de direção. Outro exemplo é a bobina (sensor) presa a uma roupa vestida por um ator. Ela é capaz de medir um campo magnético (fonte artificial) gerado por outras bobinas em posições conhecidas para determinar sua própria posição no espaço. Logo, os sensores estão “dentro” e as fontes, “fora”.

A última categoria, *Outside-In*, é formada em quase sua totalidade por sistemas ópticos. Neles, câmeras (sensores) registram imagens de um ator com marcadores (fontes artificiais) ou sem eles (fontes naturais) para posterior triangulação. Consequentemente, os sensores estão “fora” e as fontes, “dentro”.

## 1.4 Estrutura da Dissertação

No Capítulo 1, foram apresentados alguns conceitos básicos da captura de movimento e os objetivos deste trabalho.

No Capítulo 2, é apresentado um breve histórico da captura de movimento e trabalhos relacionados são comentados.

No Capítulo 3, são apresentados conceitos fundamentais ao desenvolvimento da aplicação *OpenMoCap*.

No Capítulo 4, a metodologia criada e a arquitetura escolhida para a aplicação são descritas, assim como uma visão geral da interface da aplicação é exposta.

No Capítulo 5, experimentos realizados e seus resultados são discutidos e comparados a um sistema comercial.

No Capítulo 6, conclusões e sugestões de trabalhos futuros são apresentadas.

# Capítulo 2

## Trabalhos Relacionados

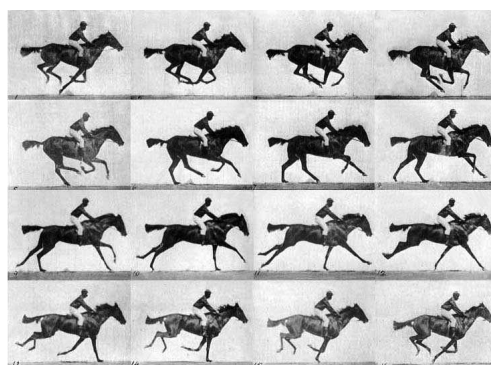
Neste capítulo, é apresentado um breve histórico da captura de movimento, incluindo processos precursores analógicos e os primeiros digitais da técnica. O estado da arte da captura óptica de movimento é descrito por meio de comentários sobre os sistemas comerciais e trabalhos recentes de pesquisa mais relevantes. Finalmente, algumas considerações são feitas relacionando este trabalho com o estado da arte.

### 2.1 Início da Captura de Movimento

A captura de movimento hoje é feita com equipamentos e aplicativos com tecnologia de ponta e alto custo. Os exemplos mais conhecidos da utilização da técnica são produções de cinema milionárias ou jogos de computador com um grande orçamento. Apesar de parecer pouco provável, as primeiras tentativas e processos considerados como captura de movimento não utilizaram computadores, não eram tridimensionais e surgiram no final do século XIX e início do século XX.

Como descrito em Menache (2000) e Kitagawa & Windsor (2008), diversos estudiosos contribuíram para o surgimento da técnica de captura de movimento. Em 1878, Eadweard Muybridge realizou uma das primeiras capturas de movimento ao registrar fotografias de um cavalo galopando usando um conjunto de câmeras acionadas pelas patas do animal (ver Figura 2.1). Dessa maneira, ele conseguiu provar que, em determinado momento do galope, as quatro patas do animal estariam no ar. Muybridge contribuiu também para os primeiros passos do cinema ao inventar o zoopraxiscópio, um dispositivo capaz de exibir um série de imagens em sequência, dando impressão de movimento.

Outro pioneiro foi Etienne-Jules Marey, mais conhecido por ser o inventor do esfigmógrafo (aparelho para medição da pulsação arterial) portátil. Marey também tinha



**Figura 2.1.** Sequência de Fotos Obtidas por Muybridge.

interesse em estudar movimentos de animais e humanos e, inspirado pelos trabalhos de Muybridge, inventou uma câmera cronofotográfica de placa fixa em 1880. Isso permitia a obtenção de uma sequência de movimentos em uma mesma placa, e mais tarde, em um mesmo filme fotográfico (ver Figura 2.2). O grande diferencial entre seu trabalho e o de Muybridge era o uso de uma única câmera em oposição ao uso de múltiplas câmeras.



**Figura 2.2.** Trabalho de Marey.

Max Fleischer e Howard Edgerton foram outros dois colaboradores para a captura de movimento. Fleischer iniciou em 1915 a primeira animação usando um rotoscópio, equipamento responsável por projetar e paralisar cada quadro de um filme. Ele filmou

seu irmão movimentando-se com uma roupa de palhaço e produziu desenhos sobre os quadros individualmente, combinando-os em um animação concluída em 1916. Edgerton, por sua vez, criou um instrumento conhecido como estroboscópio em 1931. O estroboscópio é capaz de capturar fotografias nítidas de objetos em movimentos cíclicos de alta velocidade, “congelando-os”, piscando luzes na mesma frequência em que o movimento ocorre.

Todos esses inventos e criações analógicos impulsionaram o desenvolvimento da captura de movimento em modo bidimensional. Entretanto, somente o surgimento dos computadores e a evolução do cinema possibilitaram a solidificação da técnica e, mais tarde, a gravação de movimentos tridimensionais na era digital.

## 2.2 Era Digital

Como apresentado em Menache (2000) e Kitagawa & Windsor (2008), a pesquisa e o desenvolvimento da captura digital de movimento começaram na década de 70, na busca de aplicações para a área militar e médica. A indústria de imagens geradas por computador (do inglês *computer-generated imagery* — CGI) somente descobriu as potenciais aplicações da técnica nos anos 80. O primeiro *software* de animação tridimensional comercializado foi produzido pela *Wavefront Technologies* nessa década. Na época, existiam apenas cerca de meia dúzia de empresas com foco na área de CGI e produziam, em sua maioria com o *software*, comerciais curtos de logos voadores para seus clientes.

Em 1985, os telespectadores da liga americana profissional de futebol americano se impressionaram ao ver o comercial “*Brilliance*”. Nele, uma personagem digital gerada por computador, uma robô, se mexia com bastante suavidade e realismo, assim como um ser humano. Esse foi o primeiro caso de sucesso, que se tem notícia, do uso da captura de movimento digital. Os criadores desse comercial inventaram sua própria metodologia para animar a personagem. Eles pintaram de preto dezoito juntas de uma atriz que se movia enquanto tiravam fotos de diversos ângulos com várias câmeras. Um fato interessante é que, para renderizar essa animação de trinta segundos, os autores tiveram que pedir emprestado por duas semanas vários computadores *VAX 11* existentes nos Estados Unidos.

O filme de ficção científica *Total Recall* (1990), estrelado por Arnold Schwarzenegger e Sharon Stone, por outro lado, foi o primeiro caso de fracasso da técnica. A *Metrolight*, empresa responsável pelos efeitos especiais do longa, não conseguiu produzir a animação de esqueletos (uma cena em que pessoas passam por uma máquina de raio-x

em um posto de segurança) com dados de captura de movimento. Acredita-se que a principal razão para a falha tenha sido a falta de cuidado na obtenção e na verificação inicial dos resultados das seções de captura. Os dados recebidos pelos animadores da empresa eram inúteis, mesmo com limpeza dos mesmos. Apesar disso, o filme ganhou um Oscar por seus efeitos visuais.

Um outro caso de sucesso, importante para o fortalecimento do uso da captura de movimento digital atualmente, foi o jogo de combate *FX Fighter*, lançado em 1995. Os lutadores são animados em tempo real com dados obtidos em seções de captura digital de movimento de golpes, como chutes e socos.

Hoje em dia, a captura de movimento digital possui aplicações em áreas que vão além da médica, militar e de entretenimento. Muitos esportes a usam para analisar o movimento de atletas a fim de aumentar seu desempenho e prevenir possíveis lesões. *Designers* usam dados da captura digital para entender melhor o movimento humano e suas restrições para desenvolverem produtos mais adaptados e confortáveis. Engenheiros tentam criar robôs que andam de maneira semelhante a seres humanos a partir de marchas gravadas digitalmente.

## 2.3 Estado da Arte

Nesta seção, são apresentadas soluções para a captura óptica de movimento tridimensional. Elas são divididas em sistemas comerciais e soluções baseadas em trabalhos recentes encontrados na literatura de captura de movimento. As primeiras geralmente consistem em um pacote fechado e incluem um *software* proprietário e todo equipamento específico (câmeras e outros sensores ou marcadores) necessário para gravar movimento. Já os trabalhos da área acadêmica são usualmente compostos apenas por um *software* combinado a um *hardware* de propósito geral.

Revisões bastante completas e abrangentes da área de visão computacional, computação gráfica e processamento digital de imagens sobre a captura de movimento podem ser encontradas em Gavrila (1999), Moeslund & Granum (2001) e Moeslund et al. (2006).

### 2.3.1 Sistemas Comerciais

A empresa Vicon Motion Systems (2008) oferece diversos sistemas comerciais de captura óptica de movimento usando marcadores passivos e câmeras inteligentes (a etapa de rastreamento é feita por um *hardware* especial existente no dispositivo). Um exemplo é o sistema composto por oito câmeras *Vicon T160* (capaz de detectar posições de

marcadores a 120Hz, com resolução de 16 *megapixels*) e os *softwares* proprietários para captura e análise do movimento. Porém, o custo de um sistema desses é proibitivo para o uso em pequenos laboratórios, ou mesmo doméstico.

A *NaturalPoint* possui em sua linha OptiTrack (2008) três produtos voltados para a captura óptica de movimento, também usando marcadores passivos e câmeras inteligentes. O *Body Motion Capture* é próprio para a captura de corpo inteiro, com esqueleto. O *Face Motion Capture* é indicado para captura de face e o *Tracking Tools* feito para captura de nuvem de pontos em geral. Todos são vendidos em pacotes contendo um conjunto de marcadores e seis câmeras V100, capazes de capturar pontos a 100Hz com resolução de 1 *megapixel*. É possível ainda comprar os *softwares* e as câmeras separadamente, apesar dos aplicativos somente funcionarem com as câmeras específicas. O custo dessas soluções é reduzido quando comparado aos sistemas da *Vicon*, mas a resolução máxima é bem mais baixa.

Outra companhia que fornece soluções para a captura óptica de movimento é a PhaseSpace Inc. (2008). Seu sistema *IMPULSE* é também composto por câmeras inteligentes, mas usa até 128 marcadores ativos, LEDs que possuem uma identificação única. As câmeras utilizadas no produto possuem resolução de aproximadamente 12 *megapixels* e conseguem capturar dados a uma velocidade de 480Hz. Por usar marcadores ativos, o problema conhecido como troca de marcador (do inglês *marker-swapping*), em que dois marcadores têm suas semânticas trocadas, não ocorre. A faixa de preço dessa solução encontra-se entre o sistema da *Natural Point* e o da *Vicon*.

A primeira solução comercial para a captura óptica de movimento sem marcadores é a Stage (2009) da *Organic Motion*. O princípio básico do produto é a geração de malhas tridimensionais sobre a silhueta de um ator vista por várias câmeras. Depois dessa etapa, estima-se a posição dos ossos na malha gerada, criando aproximadamente 22 marcadores “virtuais”. O produto é vendido com dez câmeras e consegue capturar movimento a 120Hz, com uma resolução equivalente a um sistema com marcadores de 2 *megapixels*. Infelizmente, seu custo é alto, comparável aos sistemas produzidos pela *Vicon*.

### 2.3.2 Trabalhos Recentes

Figuerola et al. (2003) relatam a construção de um aplicativo de captura de movimento voltado para a análise de marcha, esportes e outras aplicações na área biomédica. O foco do trabalho é o rastreamento dos marcadores. Ele é composto por três processos: segmentação, relacionamento e previsão. A segmentação é flexível e pode combinar diversas ferramentas de morfologia matemática para extrair o marcador da cena. O



relacionamento é feito por meio de casamento de padrões (tamanho e intensidade dos marcadores) e a distância mínima. Já a previsão é realizada por um filtro de *Kalman* (Grewal & Andrews, 2008) modelado considerando a posição bidimensional e velocidade constante de um marcador. O trabalho ainda cita a reconstrução tridimensional calibrada, usando o algoritmo de transformação direta linear (do inglês *Direct Linear Transformation* — DLT Hartley & Zisserman (2003)), mas não detalha com experimentos específicos. Por fim, o sistema usa câmeras com resolução de aproximadamente 1 *megapixel* e é pós-processado, sequências de vídeos devem ser gravadas para depois obter resultados da seção de captura.

Uchinoumi et al. (2004) realizam a captura de movimento sem usar marcadores, usando apenas silhuetas de atores criadas por luzes e capturadas por quatro câmeras. As informações são processadas por quatro computadores que enviam os dados por uma rede a um servidor. O servidor faz então a reconstrução tridimensional do objeto que está sendo capturado, baseado nas silhuetas processadas por cada máquina cliente. Os autores concluem o trabalho explicitando que o uso de mais câmeras pode melhorar a precisão da captura realizada, mas isso pode diminuir a velocidade, que com quatro câmeras, atinge quase 11 quadros por segundo.

Sundaresan & Chellappa (2005) também descrevem um sistema de captura de movimento sem marcadores baseados em múltiplas câmeras e representam o corpo humano usando super-quádricas. Os autores desenvolveram algoritmos para capturar o modelo do corpo humano, estimar a posição inicial e rastrear a posição do ator ao longo dos quadros de filmagem. Para analisar quantitativamente a precisão do modelo proposto, os autores propõem comparar os resultados obtidos com os de um *scanner* tridimensional.

Em Tanie et al. (2005), é apresentada uma técnica para capturar movimento usando uma grande densidade de marcadores passivos. Isso é alcançado através do uso de uma camada de fita retro refletora que, quando imageada, pode gerar diversos nodos através do algoritmo desenvolvido pelos autores. A abordagem é interessante, pois pode ser usada em sistemas ópticos de captura de movimento já estabelecidos. Resultados são mostrados em situações em que foram gerados quinhentos nodos e a taxa de captura chega a 15,2 quadros por segundo.

Castro et al. (2006) descrevem o desenvolvimento de uma sistema de captura de movimento baseado em marcadores passivos focado em análise de marcha, chamado *SOMCAD3D*. Ele também é pós-processado, assim como o desenvolvido por Figueroa et al. (2003). A técnica usada tanto para calibração quanto para a reconstrução é a DLT e o rastreamento dos pontos de interesse é feito por interpolação de curvas. Os autores ainda comparam a precisão e a acurácia do sistema construído com

outros existentes na literatura.

Raskar et al. (2007) apresentam um sistema de captura com alto desempenho e poucas restrições em relação ao ambiente de captura. Ele pode capturar movimentação, orientação tridimensional e incidência luminosa de pontos de interesse escolhidos. Não são usadas câmeras, nem *scanners* a laser, mas sim foto sensores, que funcionam como marcadores ativos, decodificando sua posição e orientação através de padrões binários emitidos por LEDs. Por não depender de câmeras, não existe limitação por parte delas na velocidade de captura, atingindo taxas de até 480Hz. Apesar do custo dos componentes utilizados ser reduzido, a sua montagem e configuração são bastante complexas. Por fim, a captura facial não é possível, tendo em vista o tamanho dos marcadores.

## 2.4 Considerações

Nas primeiras duas seções deste capítulo, foi apresentado um breve histórico da captura de movimento, explicando sucintamente o que motivou o desenvolvimento da técnica e suas aplicações. Na terceira seção, uma visão geral do estado da arte da captura óptica de movimento foi dada, incluindo sistemas comerciais e trabalhos recentes da literatura, com marcadores ativos, passivos e sem marcadores.

Todos os sistemas comerciais descritos na Seção 2.3.1 conseguem realizar a captura em tempo real, ou seja, os resultados podem ser vistos simultaneamente com o movimento do objeto alvo. Isso tem grande importância, já que pode minimizar situações em que a captura precisa ser repetida por algum tipo de erro de calibração, troca de marcadores ou interferências externas, limitando o desperdício de tempo e aumentando as chances de sucesso de utilização da técnica. A precisão desses sistemas é geralmente boa, da ordem de milímetros, apesar da necessidade de um passo relativamente elaborado de calibração. Infelizmente, para atingir toda essa robustez, *hardware* especial, bem integrado com o *software* proprietário, é necessário. Portanto, são sistemas fechados (não é possível comprar a câmera de um fabricante e usá-la com a aplicação de outro) e vendidos em pacotes com preço elevado.

Os trabalhos descritos na literatura não apresentam a mesma robustez de sistemas comerciais e geralmente não possuem uma aplicação tão bem definida. Ou seja, geralmente são mais flexíveis, apesar dos trabalhos de Figueroa et al. (2003) e de Castro et al. (2006) serem voltados para a análise de marcha. Outra diferença significativa é em relação a maturidade da tecnologia utilizada. Enquanto trabalhos na literatura focam em propostas de novas técnicas, sistemas comerciais usualmente avan-

çam e aprimoram metodologias já sólidas, como segredos industriais. Isso não significa que a reprodução dos trabalhos descritos na Seção 2.3.2 seja simples, ou até mesmo possível, já que alguns detalhes de implementação são omitidos e nenhum deles disponibiliza o código da aplicação construída. Como consequência, somente um dos trabalhos apresentados (Castro et al., 2006) compara qualitativamente a solução de captura de movimento criada com outras soluções.

A aplicação principal da solução desenvolvida nesta dissertação é a geração de dados realísticos para animar personagens humanoides virtuais. Este é o primeiro trabalho que se tem conhecimento, no Brasil, de um *software* de captura de movimento de código livre, criado para essa finalidade.

# Capítulo 3

## Fundamentos Teóricos

Este é um trabalho multidisciplinar que envolve diversos conceitos e técnicas das áreas de processamento digital de imagens, visão computacional, computação gráfica e outras. Neste capítulo, são apresentadas formalmente as principais ferramentas teóricas utilizadas dessas áreas para o desenvolvimento deste projeto.

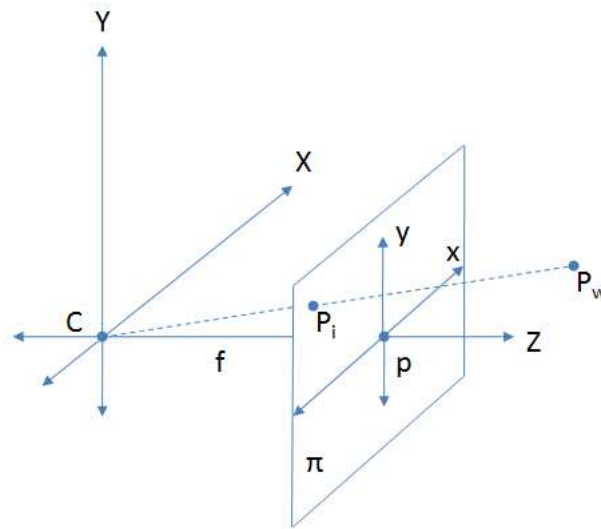
### 3.1 Modelo de Câmera

O modelo de câmera apresentado no Capítulo 1 é apenas um conceito preliminar e é estendido nesta seção. Segundo Hartley & Zisserman (2003), uma câmera é um mapeamento entre um mundo tridimensional (espaço de objetos) e uma imagem bidimensional. Ele é feito a partir de uma matriz com propriedades especiais, que representam uma transformada projetiva produzida pela câmera e seus respectivos parâmetros.

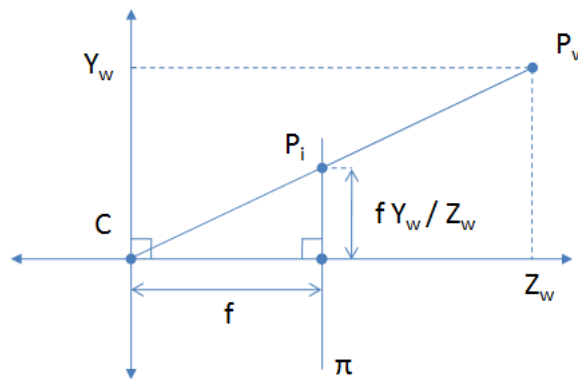
Seja  $C$  o centro de projeção de uma câmera e a origem de um sistema de coordenadas euclideo no mundo  $W$ , como mostrado na Figura 3.1. Seja ainda um plano  $\pi = f$ , chamado de plano de imagem. Um ponto no espaço com coordenadas  $P_w = (X_w, Y_w, Z_w)$  é mapeado em um ponto  $P_i$  no plano de imagem, onde uma reta, que passa pelo centro de projeção  $C$  e pelo ponto  $P_w$ , intercepta esse plano. Por meio de semelhança de triângulos, é fácil perceber que as coordenadas de  $P_i$  são  $(fX_w/Z_w, fY_w/Z_w)$ , como mostra a Figura 3.2.

$C$  também é conhecido como centro da câmera e a linha perpendicular que o liga ao plano de imagem é chamada de raio principal. Já o ponto  $p$  de interseção do raio principal com o plano  $\pi$  recebe o nome de ponto principal. Finalmente, a distância  $f$  entre o centro da câmera e o ponto principal é a distância focal.

Representando os pontos no mundo e na imagem por coordenadas homogêneas (Hartley & Zisserman, 2003), a projeção realizada pela câmera pode ser modelada por



**Figura 3.1.** Geometria Câmera Projetiva.



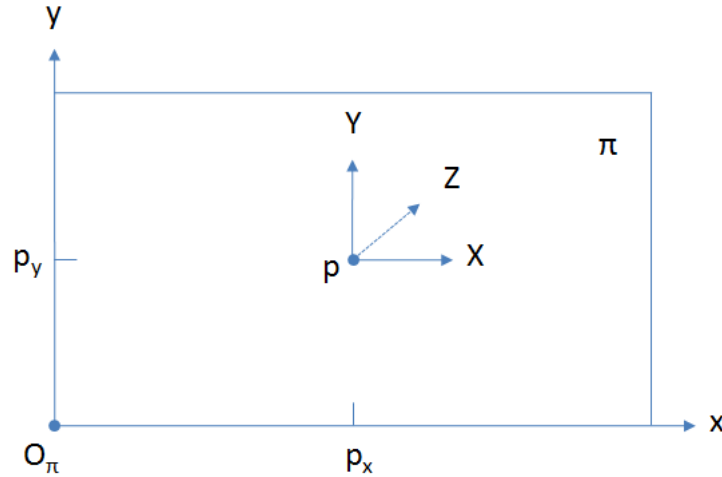
**Figura 3.2.** Semelhança de Triângulos no Plano YZ.

um mapeamento linear. Ele é escrito como uma multiplicação de matrizes, como mostra a Equação 3.1, ou mais simplificada a Equação 3.2.  $M$  é uma matriz  $3 \times 4$  conhecida como matriz de projeção da câmera e, da forma como está escrita na Equação 3.2, descreve uma câmera *pinhole* de projeção central.

$$\begin{bmatrix} fX_w \\ fY_w \\ Z_w \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad (3.1)$$

$$P_i = MP_w \quad (3.2)$$

Em geral, as imagens são representadas por um sistema de coordenadas cartesianas  $(x, y)$ , em que a origem  $O_\pi$  fica localizada na extremidade inferior esquerda da imagem, como mostra a Figura 3.3. Logo, as coordenadas do ponto principal ficam deslocadas de  $(p_x, p_y)$  e o ponto  $P_i$  tem suas coordenadas homogêneas alteradas para  $(fX_w + p_xZ_w, fY_w + p_yZ_w, Z_w)$ . Para refletir esse novo modelo, a matriz  $M$  precisa ser alterada também, como mostra a Equação 3.3.



**Figura 3.3.** Sistemas de Coordenadas da Câmera e da Imagem.

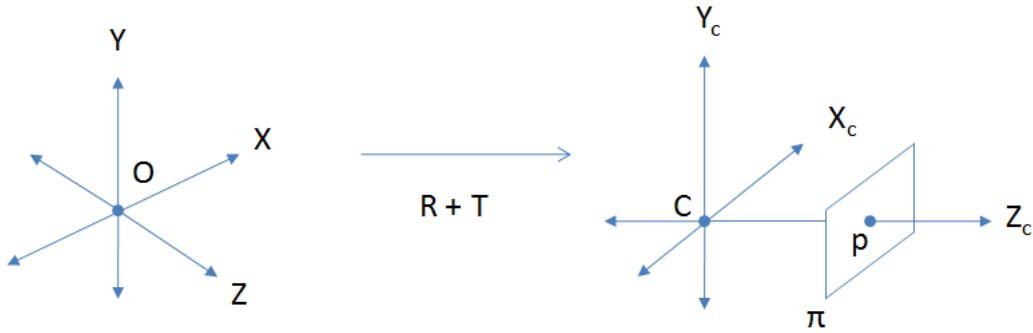
$$\begin{bmatrix} fX_w + p_xZ_w \\ fY_w + p_yZ_w \\ Z_w \end{bmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad (3.3)$$

Em câmeras reais, com sensores CCD ou CMOS, os elementos da imagem (*pixels*) geralmente não ocupam exatamente uma unidade de distância nas coordenadas da imagem. Além disso, eles podem não ser quadrados, ou seja, podem ter uma razão de aspecto diferente de 1 : 1. Particularmente, se o número de *pixels* por unidade de distância nas coordenadas da imagem for  $m_x$  e  $m_y$  ao longo das direções dadas pelos eixos  $X$  e  $Y$ , a transformação das coordenadas do mundo para coordenadas em *pixels* é feita como mostra a Equação 3.4.

$$\begin{bmatrix} fm_x X_w + p_x m_x Z_w \\ fm_y Y_w + p_y m_y Z_w \\ Z_w \end{bmatrix} = \begin{bmatrix} fm_x & 0 & p_x m_x & 0 \\ 0 & fm_y & p_y m_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad (3.4)$$

Segundo Trucco & Verri (1998) e Cyganek (2009), todos os parâmetros apresentados até o momento neste texto ( $f$ ,  $m_x$ ,  $m_y$ ,  $p_x$  e  $p_y$ ) são denominados intrínsecos da câmera, dependem apenas das características internas dela. Outros parâmetros intrínsecos, já citados no Capítulo 1, descrevem as distorções e as aberrações cromáticas produzidas nas lentes, que não são abordados nesta seção.

Existe ainda um outro conjunto de parâmetros, os extrínsecos, que trata da mudança dos sistemas de coordenadas da câmera para um sistema de coordenadas de um ponto arbitrário no mundo. A Figura 3.4 ilustra essa alteração. Transformações de sistemas de coordenadas podem ser descritas pela multiplicação de uma matriz  $3 \times 3$  de rotação  $R$  mais um vetor de translação  $T$ . Finalmente, combinando todos os parâmetros da câmera descrita nesta seção, a Equação 3.5 mostra a matriz projetiva resultante em coordenadas homogêneas.



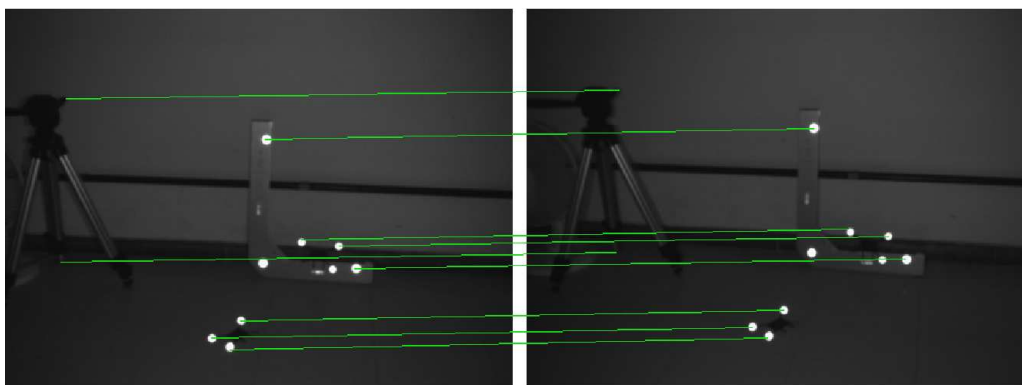
**Figura 3.4.** Mudança do Sistemas de Coordenadas da Câmera.

$$M = \begin{bmatrix} fm_x & 0 & p_x m_x \\ 0 & fm_y & p_y m_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_x \\ r_{2,1} & r_{2,2} & r_{2,3} & t_y \\ r_{3,1} & r_{3,2} & r_{3,3} & t_z \end{bmatrix} \quad (3.5)$$

## 3.2 Reconstrução

Segundo Forsyth & Ponce (2002), a disparidade entre duas imagens produzidas pelos olhos de um homem permite um ganho significativo na noção de profundidade de objetos em uma cena. Esse processo visual é conhecido como *stereopsis* e é um importante artifício binocular usado pelo cérebro para ajudar na percepção da profundidade, além de vários outros existentes, como o contraste, as sombras, o conhecimento prévio dos objetos e paralaxe. Visão estéreo, do inglês *stereo vision*, é a técnica capaz de reproduzir artificialmente esse processo utilizando duas ou mais câmeras. Ela é imprescindível para aplicações como a captura de movimento, a navegação de robôs, a cartografia e o reconhecimento aéreo.

A fusão de características e a reconstrução são os dois passos que devem ser realizados na visão estéreo. A fusão de características consiste em determinar a correspondência entre pontos de imagens produzidas com diferentes origens, mas de uma mesma cena, como mostra a Figura 3.5. Já a reconstrução é responsável por obter as coordenadas tridimensionais de pontos na cena por meio das correspondências adquiridas no primeiro passo e algumas propriedades matemáticas, formalizadas nos próximos parágrafos.

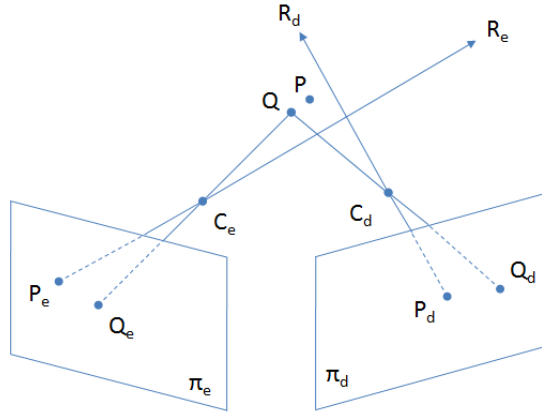


**Figura 3.5.** Exemplo de Correspondência de Pontos em Imagens Estéreo.

Dado um conjunto de câmeras com parâmetros conhecidos e no mínimo dois pontos correspondentes  $P_e$  e  $P_d$  nos planos de imagem  $\pi_e$  e  $\pi_d$ , é possível obter as coordenadas tridimensionais do ponto real  $P$  que foi projetado da cena. Isso pode ser feito por meio da interseção de dois raios  $R_e$  e  $R_d$ , como mostra a Figura 3.6. Infelizmente, a solução obtida por essa simples triangulação não é interessante na prática, já que erros na etapa de determinação dos parâmetros das câmeras e na correspondência de pontos irão causar um deslocamento significativo do ponto real  $P$ .

Diversos métodos para resolver essa triangulação com mais robustez são descritos





**Figura 3.6.** Triangulação e Solução Algébrica para Reconstrução.

em Hartley & Sturm (1997). Um dos métodos apresentados, relativamente simples e rápido, é o de quadrados mínimos linear, que usa propriedades puramente algébricas para obter resultados. Seja um ponto  $P$  e sua projeção  $P_\alpha$ , em um plano de imagem  $\pi_\alpha$ , como representados em coordenadas homogêneas na Equação 3.6. Considere ainda as relações existentes entre o ponto tridimensional  $P$ , sua projeção  $P_\alpha$  e a matriz projetiva  $M_n$ , de uma câmera  $n$ , como apresentadas na Equação 3.7. A notação  $M_n^{iT}$  simboliza a linha de índice  $i$  em uma matriz projetiva de uma câmera  $n$ . Substituindo  $k$ , as relações expressas na Equação 3.8 são obtidas.

$$P_\alpha = k[X_{P_\alpha}, Y_{P_\alpha}, 1]^T, \quad \forall k \geq 0 \in \Re \quad (3.6)$$

$$P_\alpha = M_n P \quad kX_{P_\alpha} = M_n^{1T} P \quad kY_{P_\alpha} = M_n^{2T} P \quad k = M_n^{3T} P \quad (3.7)$$

$$M_n^{3T} P X_{P_\alpha} = M_n^{1T} P \quad e \quad M_n^{3T} P Y_{P_\alpha} = M_n^{2T} P \quad (3.8)$$

Tendo em vista essas relações, é possível construir um sistema linear homogêneo sobredeterminado do tipo  $Ax = 0$ . Ele determina, a menos de uma fator de escala  $k$ , as coordenadas tridimensionais de um ponto, a partir de no mínimo duas projeções e duas matrizes de câmeras conhecidas. Especificamente, no caso de duas matrizes de câmeras projetivas  $M_1$  e  $M_2$ , o sistema é escrito como observado na Equação 3.9. Esse sistema pode ter sua solução dos quadrados mínimos determinada por meio de decomposição em valores singulares (do inglês *Singular Value Decomposition* — SVD, ver Seção 3.2.1) escrevendo  $P = (X_P, Y_P, Z_P, 1)$ , forçando a coordenada  $W_P$ , que representa o fator de escala, ser igual a 1. Uma ilustração geométrica dessa solução é apresentada na Figura 3.6, em que as distâncias entre os pontos  $P_e Q_e$  e  $P_d Q_d$  são minimizadas para obter o

ponto tridimensional  $Q$ , mais próximo das coordenadas reais de  $P$  do que a interseção dos raios  $R_e$  e  $R_d$ . Finalmente, esse método de triangulação de quadrados mínimos é facilmente extensível para mais câmeras. Para isso é suficiente acrescentar as novas equações (restrições) do tipo  $P_\alpha = M_n P$  na Equação 3.9.

$$\begin{bmatrix} X_{P_e} M_1^{3T} - M_1^{1T} \\ Y_{P_e} M_1^{3T} - M_1^{2T} \\ X_{P_d} M_2^{3T} - M_2^{1T} \\ Y_{P_d} M_2^{3T} - M_2^{2T} \end{bmatrix} \begin{bmatrix} X_P \\ Y_P \\ Z_P \\ W_P \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (3.9)$$

### 3.2.1 Decomposição em Valores Singulares

Um problema clássico na área de visão computacional e engenharias é a resolução de sistemas de equações lineares homogêneas, como a Equação 3.9, da Seção 3.2. Sistemas desse tipo podem ter sua solução determinada por meio de uma técnica chamada SVD (Press et al., 1992). Ela é baseada em um teorema de álgebra linear que diz que qualquer matriz  $A$ ,  $m \times n$ , sendo  $m$  (número de linhas) maior ou igual a  $n$  (número de colunas), pode ser escrita ou decomposta por três matrizes  $U$ ,  $W$ ,  $V$  com propriedades matemáticas especiais.  $U$  é uma matriz coluna ortogonal  $m \times n$ ,  $W$  é uma matriz diagonal  $n \times n$ , com zeros e elementos relacionados aos autovalores de  $A^T A$ , e  $V$  é uma matriz ortogonal  $n \times n$  transposta. As Equações 3.10 e 3.11 exibem essa decomposição.

$$A = U W V^T \quad (3.10)$$

$$\begin{bmatrix} A_{1,1} & \dots & A_{1,m} \\ \vdots & \ddots & \vdots \\ A_{n,1} & \dots & A_{n,m} \end{bmatrix} = \begin{bmatrix} U_{1,1} & \dots & U_{1,m} \\ \vdots & \ddots & \vdots \\ U_{n,1} & \dots & U_{n,m} \end{bmatrix} \begin{bmatrix} W_{1,1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & W_{n,n} \end{bmatrix} \begin{bmatrix} V_{1,1} & \dots & V_{1,n} \\ \vdots & \ddots & \vdots \\ U_{n,1} & \dots & U_{n,n} \end{bmatrix} \quad (3.11)$$

Se o número de equações linearmente independentes que compõem  $A$  for igual ao número de incógnitas,  $m = n$ , a matriz  $W$  vai ter exatamente um elemento  $d_i$  em sua diagonal com valor zero. A solução exata para o sistema é encontrada na coluna de  $V$  correspondente ao índice  $i$ . As implementações de SVD geralmente reordenam as matrizes decompostas para que os elementos pertencentes à diagonal de  $W$  estejam em ordem decrescente. Portanto, em uma decomposição em valores singulares ordenada, a solução exata do sistema encontra-se na última coluna de  $V$ .

Caso o número de equações linearmente independentes que compõem  $A$  seja maior

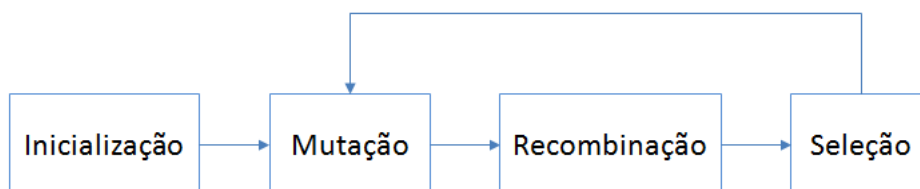
que o número de incógnitas,  $m > n$ , o sistema é sobredeterminado e não existe uma solução exata. Logo, não existirá um elemento  $d_i$  pertencente à diagonal de  $W$  com valor zero. Particularmente, nesse caso, é interessante obter uma solução que minimize a soma dos quadrados dos erros de todas as variáveis. Essa solução é a mais provável em situações em que o ruído é do tipo gaussiano. A solução dos quadrados mínimos é encontrada também na última coluna de  $V$  em uma decomposição em valores singulares ordenada.

### 3.3 Evolução Diferencial

Evolução diferencial é um algoritmo estocástico, baseado em população e gerações, para otimização de funções lineares, não-lineares e não diferenciáveis em espaços contínuos, desenvolvido por Storn & Price (1997). É um método de otimização global usado nas áreas de engenharia, estatística, computação e finanças (Storn, 2009). Especificamente, seu uso é interessante em problemas em que uma solução analítica é difícil ou até mesmo impossível.

Um problema de otimização pode ser definido da seguinte maneira. Seja um vetor  $V = (v_1, v_2, \dots, v_n)$ , com  $n$  variáveis, pertencente a  $\mathfrak{R}^n$  e uma função de custo  $f(V) : \mathfrak{R}^n \mapsto \mathfrak{R}$ . O objetivo é descobrir um vetor  $V$  que minimize ou maximize o valor resultante de  $f(V)$ , sujeito a restrições de outros dois vetores de limites, o inferior  $L = (l_1, l_2, \dots, l_n)$  e o superior  $H = (h_1, h_2, \dots, h_n)$ , também pertencentes a  $\mathfrak{R}^n$ .

A evolução diferencial é uma técnica que faz parte de uma classe de algoritmos conhecida como evolutivos, que também inclui algoritmos genéticos, estratégias evolutivas e programação evolutiva. Portanto, ele compartilha o fluxo geral de um algoritmo evolutivo exibido pela Figura 3.7. São quatro as etapas existentes: *inicialização*, *mutação*, *recombinação* e *seleção*.



**Figura 3.7.** Esquema Geral de Algoritmos Evolutivos.

A fase de inicialização consiste em produzir, em uma primeira geração  $g = 0$ , uma população com  $p$  indivíduos (um dos parâmetros de entrada do algoritmo), cada um deles podendo ser uma solução para o problema de otimização. Nessa fase, cada

indivíduo  $I_{(g,i),0 \leq i < p}$  possui um vetor de variáveis gerado aleatoriamente por uma distribuição uniforme, dentro dos limites impostos pelos vetores  $L$  e  $H$  e pelo número de dimensões (variáveis)  $d$ .

Nas duas etapas seguintes, cada elemento  $I_{(g,i,j),0 \leq j < d}$  do vetor de variáveis de um indivíduo  $I_{(g,i)}$  pode sofrer uma mutação ou ser recombinado, produzindo um provável substituto  $\rho_{(g+1,i)}$  para a próxima geração. A mutação é feita por meio da diferença ponderada entre dois elementos dos vetores de dois indivíduos diferentes selecionados aleatoriamente, somada a um terceiro, como mostra a Equação 3.12.  $F$  é um outro parâmetro de entrada, conhecido como fator de mutação, e varia idealmente entre  $[0,3; 0,9]$ . O propósito dessa operação é aumentar o espaço de busca das soluções.

$$\rho_{(g+1,i,j)} = I_{(g,a_1,j)} + F(I_{(g,a_2,j)} - I_{(g,a_3,j)}), a_1 \neq a_2 \neq a_3 \neq i \quad (3.12)$$

Diferentemente da mutação, a recombinação tem como objetivo aproveitar algumas variáveis de  $I_{(g,i,j),0 \leq j < d}$ , de boas soluções obtidas por indivíduos de gerações anteriores. Um outro parâmetro de entrada é a constante de recombinação  $Cr$ , que indica a probabilidade  $[0, 1]$  dessa operação ocorrer. Ou seja, uma parte das variáveis do indivíduo  $\rho_{(g+1,i)}$  sofre mutação e outra sofre recombinação.

A etapa final é a seleção e o seu propósito é remover os piores indivíduos da população para a próxima geração. Um indivíduo  $I_{(g,i)}$  somente será substituído por  $\rho_{(g+1,i)}$  se o valor da função de custo  $f(\rho_{(g+1,i)})$  for menor que  $f(I_{(g,i)})$ , caso contrário o provável substituto é descartado e o indivíduo original permanece na geração  $g + 1$  (ver Equação 3.13). Após essa etapa, o algoritmo continua voltando a etapa de mutação até que uma condição de parada seja atingida, como um número de gerações limite ou um valor mínimo para a função de custo. Portanto, o resultado final é o conjunto de variáveis do indivíduo que tem menor custo  $f(I_{(g,i)})$  na última iteração.

$$I_{(g+1,i)} = \begin{cases} \rho_{(g+1,i)} & \text{se } f(\rho_{(g+1,i)}) < f(I_{(g,i)}) \\ I_{(g,i)} & \text{se } f(\rho_{(g+1,i)}) \geq f(I_{(g,i)}) \end{cases} \quad i = 1, 2, \dots, p \quad (3.13)$$

### 3.4 Limiarização

Como definido em Gonzalez & Woods (2006), uma imagem digital pode ser manipulada em dois tipos de domínios: o espacial e o de transformadas. No primeiro, os métodos de processamento atuam diretamente nos elementos da imagem, nos *pixels*. Já no segundo, deve-se primeiramente converter a imagem para o domínio da transformada escolhida, realizar o processamento e, somente depois, realizar uma transformação inversa para o domínio espacial para obter o resultado final.

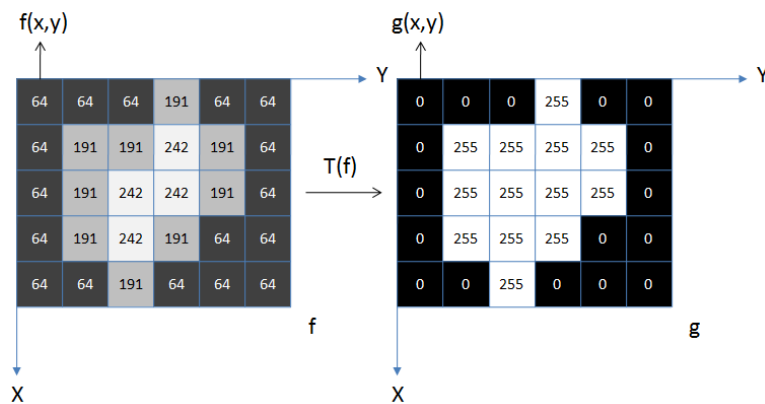
Especificamente, técnicas aplicadas no domínio espacial ainda podem ser divididas com base no tipo de atuação nos elementos de imagem. Se para processar um *pixel* é preciso somente conhecer seu valor, o método de processamento é considerado uma transformação de intensidade. Caso seja necessário saber o valor de outros elementos da sua vizinhança, o método pertence a categoria de filtragem espacial. Limiarização (do inglês *thresholding*) é uma técnica que atua no domínio espacial e é considerada uma transformação de intensidade importante para problemas de segmentação, alteração de contraste e coloração artificial.

Seja uma imagem  $f$ , de largura  $x$  e altura  $y$ , e um valor de intensidade  $f(x, y)$  associado a cada *pixel* da imagem. Uma operação de limiarização simples consiste na aplicação de uma função especial  $T$  sobre cada  $f(x, y)$  gerando um outro *pixel*  $g(x, y)$  em uma nova imagem  $g$  de mesma dimensão, como mostra a Equação 3.14. A função  $T$  é composta por  $i$  faixas de operação com limiares  $r_i$  que transformam a intensidade do elemento de imagem para determinado valor  $v_i$ , como na Expressão 3.15.

$$g(x, y) = T[f(x, y)] \quad (3.14)$$

$$T[f(x, y)] = \begin{cases} v_0 & f(x, y) \leq r_0 \\ v_1 & r_0 < f(x, y) \leq r_1 \\ \vdots & \vdots \\ v_i & r_{i-1} < f(x, y) \leq r_i \end{cases} \quad (3.15)$$

A Figura 3.8 ilustra a aplicação de uma limiarização em duas faixas, ou seja, binária, descrita pela Expressão 3.16. Em uma cena simples, ela poder ser útil na separação de um único objeto com relação ao fundo.



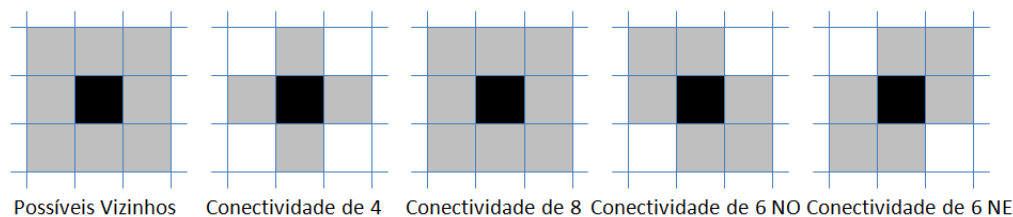
**Figura 3.8.** Exemplo de Limiarização Binária.

$$T[f(x, y)] = \begin{cases} 0 & f(x, y) \leq 190 \\ 255 & 190 < f(x, y) \leq 255 \end{cases} \quad (3.16)$$

### 3.5 Componentes Conectados

Como apresentado em Steger et al. (2008) e Umbaugh (2005), a segmentação de objetos somente por limiarização não é suficiente para cenas que possuem mais de um objeto em um fundo. Em geral, nessas situações, como por exemplo, vários marcadores reflexivos em uma cena, o ideal é retornar individualmente cada um deles, como uma região separada e bem definida de *pixels*. Para isso, é necessário identificar quais elementos de uma imagem estão conectados e pertencem a uma mesma região, baseado no conceito de vizinhança ou adjacência.

Um *pixel* pode ter até oito vizinhos: dois vizinhos na horizontal, dois vizinhos na vertical e quatro vizinhos na diagonal. Dessa maneira, é possível definir três tipos básicos de conectividade: conectividade de quatro, conectividade de oito e conectividade de seis. A figura 3.9 ilustra essas definições.



**Figura 3.9.** Conectividade de *Pixels*.

O uso unicamente da vizinhança de oito *pixels* ou da vizinhança de quatro *pixels* leva a um problema conhecido como o *dilema da conectividade*. Seja uma imagem binária como a Figura 3.10. Utilizando uma conectividade de quatro, existem quatro objetos separados por um fundo constituído de cinco componentes. Isso contraria a lógica, já que os objetos deveriam ser separados por um único componente de fundo, não vários. Caso a conectividade de oito seja usada, existe apenas um objeto, uma curva fechada, e um único componente de fundo. A razão também é contrariada dessa maneira, uma curva fechada deveria separar o fundo em dois componentes, não somente um.

A solução para esse problema está no uso de adjacência mista (conectividade de oito para objetos e conectividade de quatro para fundo, ou vice-versa) ou na adjacência de seis. Após a definição de um desses tipos de adjacência, um algoritmo de rotulação pode ser aplicado na imagem para separar os objetos existentes. Todos os *pixels* da

0	0	255	0	0
0	0	255	0	0
255	255	0	255	255
0	0	255	0	0
0	0	255	0	0

Figura 3.10. Dilema da Conectividade.

imagem são varridos no processamento e assinalados com um número identificador, dependendo do objeto que fazem parte. A Figura 3.11 mostra o fluxograma de um algoritmo de rotulação com adjacência de seis *pixels* baseado em Umbaugh (2005).

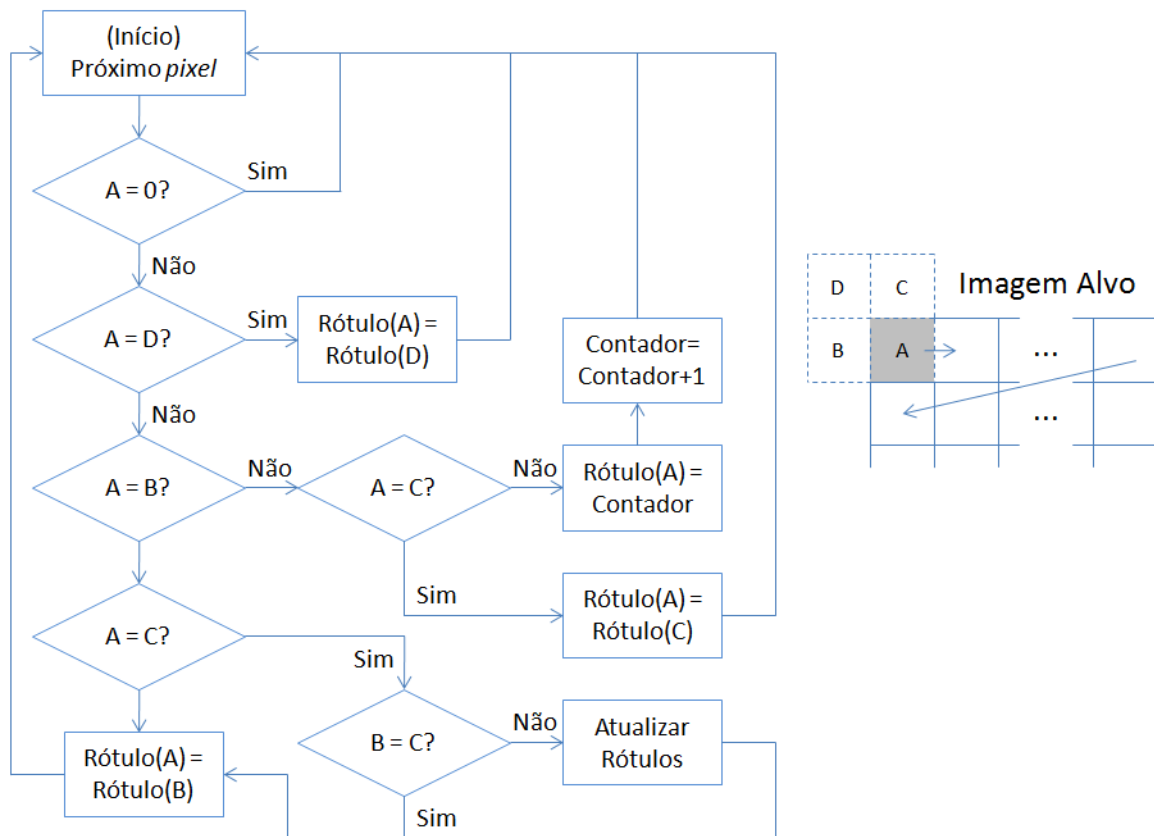


Figura 3.11. Fluxograma do Algoritmo de Rotulação com Adjacência de Seis *Pixels* (Umbaugh, 2005).

Se *B*, *C* ou *D* ultrapassarem os limites das bordas da imagem alvo, seus valores são considerados como de fundo, isto é, zero. Devido ao método tradicional de varredura de *pixels* em uma imagem (esquerda para direita e cima para baixo), podem aparecer situações em que um *pixel* tenha vizinhos (*B* e *C*) com rótulos diferentes. Isso indica

que dois objetos distintos estão conectados e, portanto, constituem um mesmo objeto. Essa situação é tratada corretamente por meio da função de atualização de rótulos, que mantém uma tabela daqueles que são correspondentes.

### 3.5.1 Propriedades de um Componente Conectado

A partir do algoritmo de rotulação, apresentado na seção anterior, é possível extrair algumas características específicas de cada objeto existente na cena. Duas características que ajudam na localização e na classificação desses objetos são a área (tamanho relativo) e a posição do seu centróide. Para determiná-las facilmente, é preciso definir primeiramente a Função 3.17 (ver Umbaugh (2005)) em uma imagem já rotulada  $I$ , com  $i$  objetos e coordenadas  $x, y$ .

$$I_i(x, y) = \begin{cases} 1 & I(x, y) = i \\ 0 & I(x, y) \neq i \end{cases} \quad (3.17)$$

Em uma imagem de dimensões  $N \times N$  *pixels*, a área  $A_i$ , medida em *pixels*, do  $i$ -ésimo objeto pode ser calculada pela Equação 3.18. O centro da área  $A_i$  ou, mais especificamente, o centróide do  $i$ -ésimo objeto, pode ser obtido pela determinação do ponto médio da distribuição espacial de seus *pixels*. As coordenadas  $(\bar{x}_i, \bar{y}_i)$  desse ponto são calculadas pela Equações 3.19.

$$A_i = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} I_i(x, y) \quad (3.18)$$

$$\bar{x}_i = \frac{1}{A_i} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} x I_i(x, y) \quad \bar{y}_i = \frac{1}{A_i} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} y I_i(x, y) \quad (3.19)$$

## 3.6 Estimador Alfa-Beta

Como descrito em Yoo & Kim (2003), Alfa-Beta é um estimador desenvolvido na década de 50, popular em problemas de rastreamento e filtragem, devido a sua modelagem simples e seu custo computacional reduzido. Ele é composto por dois estados, sendo que um deles é calculado pela integração do outro no tempo. Seja o primeiro estado a posição  $x$  e o segundo estado a velocidade  $v$ . Assumindo que a velocidade é constante em um intervalo pequeno de tempo entre medidas  $\Delta t$ , as Equações 3.20 e 3.21 mostram como os valores da posição estimada  $x_p$  e da velocidade estimada  $v_p$  podem ser obtidos em um instante de tempo  $k + 1$ .



$$x_p(k+1) = x_s(k) + \Delta t v_s(k) \quad (3.20)$$

$$v_p(k+1) = v_s(k) \quad (3.21)$$

Na grande maioria dos intervalos de tempo, a posição prevista  $x_p$  não é igual à posição observada  $x_m$ , devido aos ruídos e à própria dinâmica de movimento do alvo. Portanto, um erro residual  $r$  existe, e é calculado pela Equação 3.22 em um instante de tempo  $k$ . Esse erro descreve o impacto das perturbações ocorridas na cena e na trajetória do objeto observado. As Equações 3.23 e 3.24 mostram como os valores previstos são suavizados ( $x_s$  e  $v_s$ ) com o erro calculado.

$$r(k) = x_m(k) - x_p(k) \quad (3.22)$$

$$x_s(k) = x_p(k) + \alpha r(k) \quad (3.23)$$

$$v_s(k) = v_p(k-1) + \frac{\beta}{\Delta t} r(k) \quad (3.24)$$

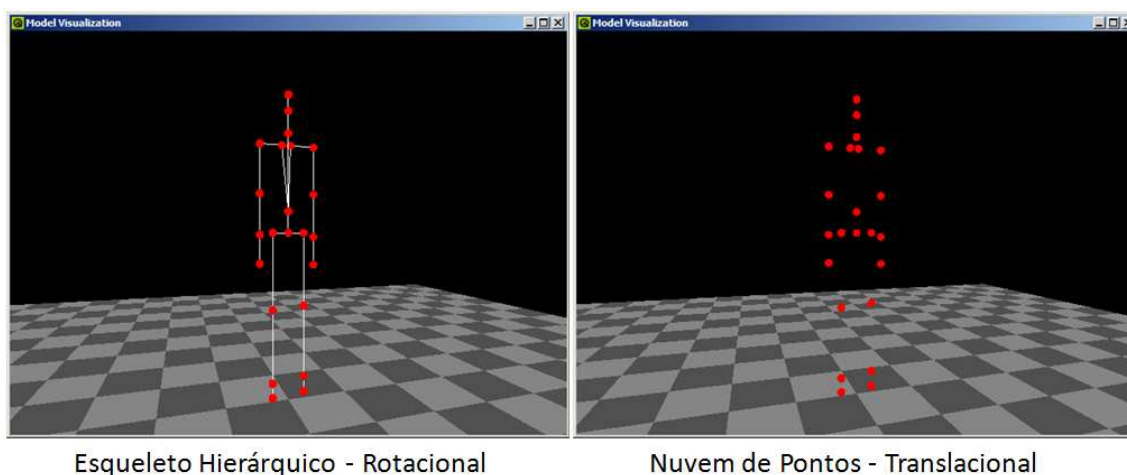
As constantes  $\alpha$  e  $\beta$  controlam a intensidade dessa suavização. Quanto maior o valor delas, mais rápida é a resposta do estimador a mudanças bruscas de trajetória. Por outro lado, quanto menor o valor delas, melhor será a resposta em relação à diminuição do ruído presente. Portanto, a escolha dos valores dessas constantes é um compromisso entre a suavização de ruído e a velocidade de resposta a uma nova trajetória. Finalmente, os valores de  $\alpha$  e  $\beta$  precisam ser definidos experimentalmente, dependendo da aplicação.

### 3.7 Dados de Captura de Movimento

Segundo Kitagawa & Windsor (2008), dois são os principais tipos de dados gerados por um sistema de captura de movimento. O primeiro é o translacional, em que um conjunto de coordenadas cartesianas  $(x, y, z)$ , ordenadas no tempo, é armazenado para cada ponto de interesse rastreado. O segundo tipo, o rotacional, define um sistema de coordenadas cartesianas com origem no centro de massa da estrutura alvo de captura e, a partir dele, armazena hierarquicamente rotações relativas dos outros pontos de interesse, em uma ordem previamente determinada (como rotação no eixo  $X$ , depois no eixo  $Y$  e finalmente no eixo  $Z$ ).

Diversos são os tipos de arquivo existentes para armazenar essas informações de uma seção de captura de movimento, como o BVH, o C3D, o TRC e o ASF (Menache, 2000). A diferença mais importante é o tipo de dados suportados, geralmente nuvem de pontos para dados translacionais e esqueleto hierárquico para dados rotacionais (ver Figura 3.12). Existem ainda outras diferenças, como os tipos de metadados inclusos, mas usualmente não são muito relevantes.

Como já explicado no Capítulo 1, devido ao método de obtenção de dados, os sistemas ópticos produzem dados translacionais nativamente, enquanto sistemas de prótese, magnéticos e acústicos produzem dados rotacionais. Dados de translação são mais apropriados para animação facial, animação de marionetes e aplicações em tempo real. Já dados de rotação são mais apropriados para a construção de um esqueleto, útil no caso de captura de corpo inteiro.



**Figura 3.12.** Tipos de Dados Gerados por Sistemas de Captura de Movimento.

### 3.8 Considerações

Neste capítulo, foram apresentados formalmente diversos conceitos teóricos necessários para a concretização do objetivo geral deste trabalho, a construção de um *software* de captura de movimento óptica de código livre. Grande parte das ferramentas aqui apresentadas são simples e bem maduras em suas respectivas áreas. Porém, a combinação delas pode resultar em uma aplicação útil e poderosa. O Capítulo 4 – Metodologia é responsável por descrever como esse arcabouço apresentado foi organizado e adaptado para alcançar os objetivos propostos neste trabalho.

# Capítulo 4

## Metodologia

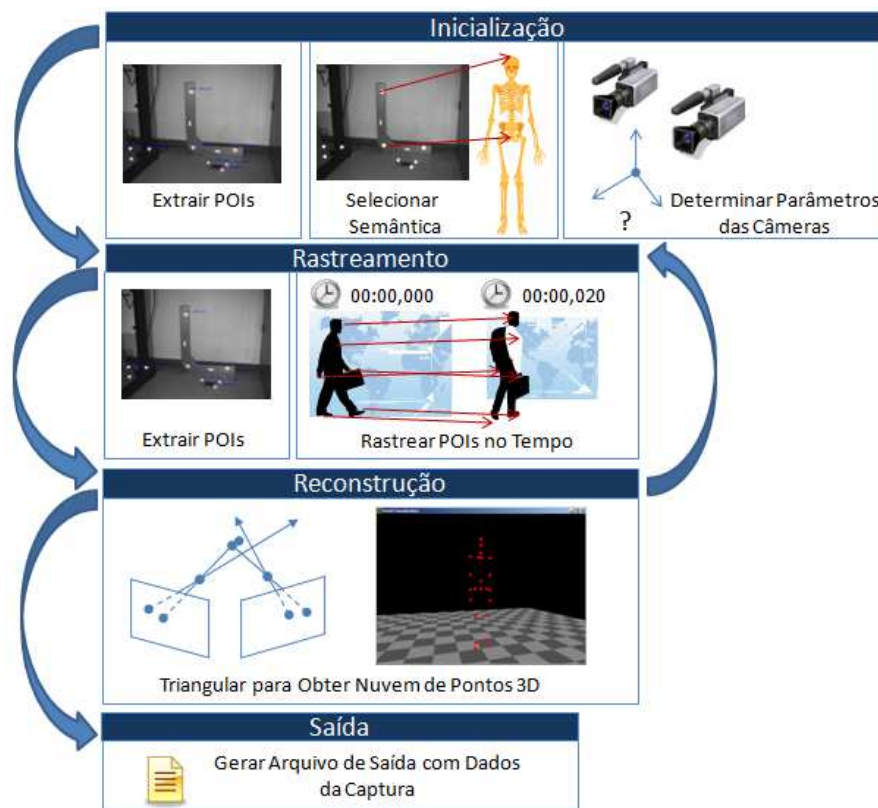
Este é o capítulo que evidencia o uso dos conceitos teóricos apresentados no capítulo anterior. Especificamente, o papel de cada ferramenta é explicado no contexto de suas etapas relevantes no fluxo de captura de movimento, imprescindível para a aplicação desenvolvida neste trabalho. Outros tópicos abordados são a arquitetura e as decisões de implementação tomadas. Finalmente, uma breve descrição dos pacotes e das principais classes que compõem o software construído é feita.

### 4.1 Fluxo de Captura de Movimento

Como descrito no Capítulo 1, o fluxo da captura de movimento é complexo, mas pode ser separado basicamente por quatro passos: *inicialização*, *rastreamento*, *reconstrução* e *saída*. O fluxograma exibido pela Figura 4.1 ilustra quais são os objetivos de cada etapa, assim como implementadas no *software OpenMoCap*.

A inicialização consiste em, primeiramente, extrair todos os POIs presentes nas imagens fornecidas pelas câmeras conectadas ao sistema. Posteriormente, a partir de um modelo previamente definido de um alvo de captura, como uma pessoa, cada POI recebe uma semântica. Nesse exemplo em particular, as semânticas podem ser cabeça, esterno, ombro direito, ombro esquerdo, dentre outras articulações do esqueleto humano. O passo final dessa etapa é descobrir os parâmetros das câmeras. Ao final desse processo, a aplicação está em um estado válido, com todos os dados necessários para prosseguir para as próximas etapas, que começam efetivamente a captura de movimento.

A primeira fase do rastreamento ocorre de modo semelhante à inicialização: os POIs tem seus centróides calculados a partir da imagem de cada câmera. A diferença é que nessa fase eles já possuem semânticas associadas. O propósito é não perder essa



**Figura 4.1.** Fluxograma da Captura de Movimento no *OpenMoCap*.

ligação estabelecida na inicialização, ou seja, deseja-se literalmente perseguir no tempo a posição de cada POI, associando a semântica anterior correta.

A reconstrução é o passo seguinte e consiste em, a partir dos POIs com semânticas associadas e os parâmetros conhecidos das câmeras, triangular a posição daqueles pontos correspondentes para determinar suas respectivas profundidades na cena. O resultado é uma nuvem de pontos tridimensionais, que é passada quadro a quadro para a etapa de saída. Nela, os dados obtidos são escritos em um arquivo com formato padrão de captura de movimento.

Em uma seção de captura de movimento, a inicialização é geralmente realizada uma única vez, enquanto o rastreamento, a reconstrução e a saída são processos contínuos. Ou seja, antes de começar a gravar movimento, o sistema deve ser inicializado, depois, a cada quadro das câmeras do sistema, a sequência ordenada (rastreamento, reconstrução e saída) é realizada ininterruptamente, até que se decida parar de gravar movimento. As próximas seis seções deste capítulo explicam a abordagem desenvolvida para realização dessas etapas no *OpenMoCap*, relacionando os conceitos teóricos apresentados no capítulo anterior.

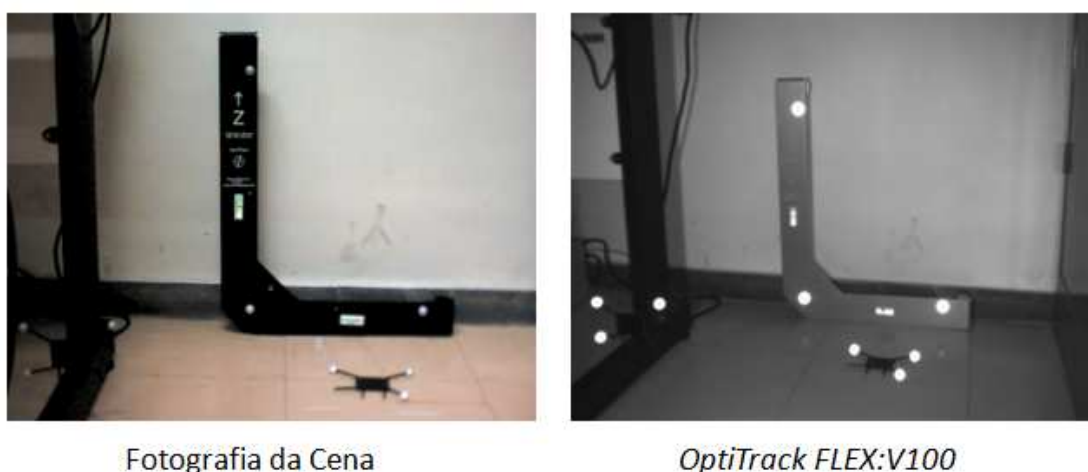
## 4.2 Extração de POIs

Este é um trabalho de captura óptica de movimento que utiliza marcadores passivos retroreflexivos e câmeras com LEDs infravermelhos, com um filtro no mesmo comprimento de onda. Todo o *hardware* utilizado no desenvolvimento da aplicação, exceto o computador e seus periféricos, são fabricados pela OptiTrack (2008). A Figura 4.2 exibe os produtos utilizados, marcadores, duas câmeras modelo *OptiTrack FLEX:V100* com custo aproximado de 600 dólares cada e outros acessórios.



**Figura 4.2.** *Hardware Utilizado da NaturalPoint.*

A Figura 4.3 compara imagens de uma mesma cena com marcadores obtidas em escala colorida por uma câmera digital comum e em escala de cinza produzida pelas câmeras da OptiTrack (2008). Claramente os marcadores são realçados, apesar de outros pontos também apresentarem alta intensidade de branco, como o líquido dos níveis.



**Figura 4.3.** Comparação Fotografia e Imagem Obtida pela Câmera *OptiTrack FLEX:V100*.

A extração parcial dos pontos evidenciados pelos marcadores é feita por meio da aplicação de uma limiarização binária, que recebe como parâmetro um limiar, definido manualmente pelo usuário da aplicação. Esse limiar assume valores entre 0 e 255, intensidades possíveis para um *pixel* em uma imagem em escala cinza de 8 *bits*. A princípio, esse procedimento é suficiente para uma separação inicial entre o fundo e objetos de interesse.

Porém, o intuito é separar unicamente os POIs detectados e eliminar ruídos, isto é, partes da imagem com alta intensidade de branco não relacionadas aos marcadores. Para isso, um algoritmo de cálculo de componentes conectados com adjacência de 6 *pixels* (como apresentado na Seção 3.5) é aplicado. Na prática, ele é ligeiramente modificado para aumentar sua eficiência e reduzir ruído.

Primeiramente, a limiarização é feita em paralelo, ou seja, a imagem não é varrida duas vezes para aplicar as operações de limiarização e rotulação. Cada elemento de imagem analisado já é marcado como sendo fundo ou objeto e, assim, o algoritmo de componentes conectados continua seu fluxo normal. Em seguida, é feito o cálculo imediato da área dos componentes conectados, possibilitando a identificação daqueles que provavelmente não são marcadores. Eles são removidos com base nos limites de área mínima e máxima, configurados pelo usuário a partir de uma porcentagem da área total da imagem. Esses valores devem ser determinados experimentalmente, já que a distância do alvo de captura afeta diretamente a escala dos marcadores e portanto, suas respectivas áreas em *pixels*. Somente os componentes restantes tem seus centróides calculados.

A Figura 4.4 exemplifica todo o processo de extração de POIs. A imagem da esquerda é a original, sem qualquer processamento, obtida diretamente da câmera. A imagem do meio ilustra como seria a limiarização binária, com um limite alto de intensidade. Nela existem doze objetos destacados, nove marcadores e três componentes de ruído (líquidos dos níveis na imagem original). Por fim, a última imagem exhibe um círculo ao redor dos centróides calculados de cada POI detectado. Nesse caso, o ruído foi eliminado por meio da remoção de componentes com área inferior a 0.02% do quadro original.

Finalmente, algumas sugestões para melhorar a extração de POIs, ainda não implementadas na versão atual do *OpenMoCap*, são apresentadas a seguir. Filtros do espectro visível podem ser utilizados nas câmeras para aumentar a robustez do *software* em relação à iluminação ambiente, como um pré-processamento em *hardware*. Outra ideia para ajudar na redução de ruído é solicitar ao usuário da aplicação que informe o número de marcadores existentes na cena capturada, removendo objetos excedentes. Por fim, métodos de binarização adaptativos como o de Otsu (Gonzalez & Woods,



**Figura 4.4.** Detecção de POIs.

2006) podem retirar a necessidade de se determinar um limiar fixo de segmentação, útil quando o usuário não é especialista, apesar de talvez não ser viável devido ao tempo de processamento requerido.

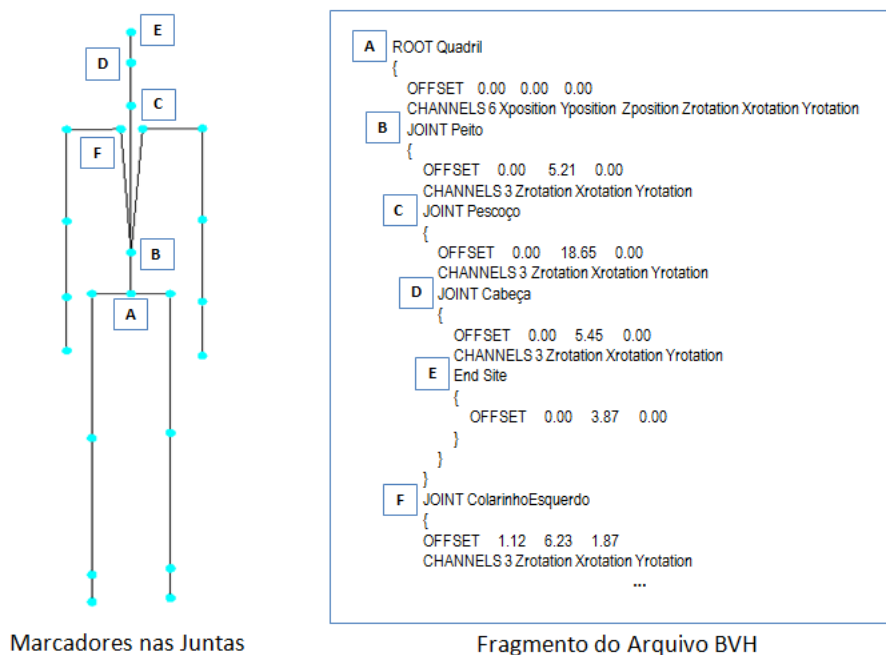
### 4.3 Seleção de Semântica dos POIs

A partir dos POIs detectados pela etapa de extração, é necessário associar uma semântica a cada um deles a fim de identificá-los unicamente de maneira simples. Nomes representativos devem ser escolhidos baseados no alvo de captura. A imagem do esqueleto de palito na Figura 4.5 exemplifica a localização e a semântica de alguns POIs significativos em um corpo humano, posicionados nas articulações ou juntas.

O *OpenMoCap* possui um importador de esqueletos escritos no formato *Biovision Hierarchy* (BVH). Esse é um tipo de arquivo padrão de captura de movimento, que armazena informações estáticas e dinâmicas em duas seções separadas: hierarquia e movimento (Menache, 2000). A última parte, a de movimento, não é relevante para o importador, já que contém apenas dados sequenciais, ordenados no tempo, de rotações relativas dos POIs.

A imagem da direita na Figura 4.5 exibe um trecho da primeira parte de um arquivo BVH comumente usado em capturas de corpo inteiro. A relação hierárquica existentes entre os POIs é representada por blocos, definidos por chaves. A primeira informação para cada POI é o nome de sua junta, sua semântica. Em seguida, é encontrado seu deslocamento translacional relativo a uma junta pai (se existir), que dá origem ao comprimento do osso entre as duas juntas. Por fim, as grandezas que serão medidas no tempo (parte dinâmica do arquivo) são informadas para cada POI.

Outra maneira fácil de entrada de semântica é um arquivo simples de texto, con-



**Figura 4.5.** Sugestão de Posicionamento de Marcadores e Representação Hierárquica em BVH.

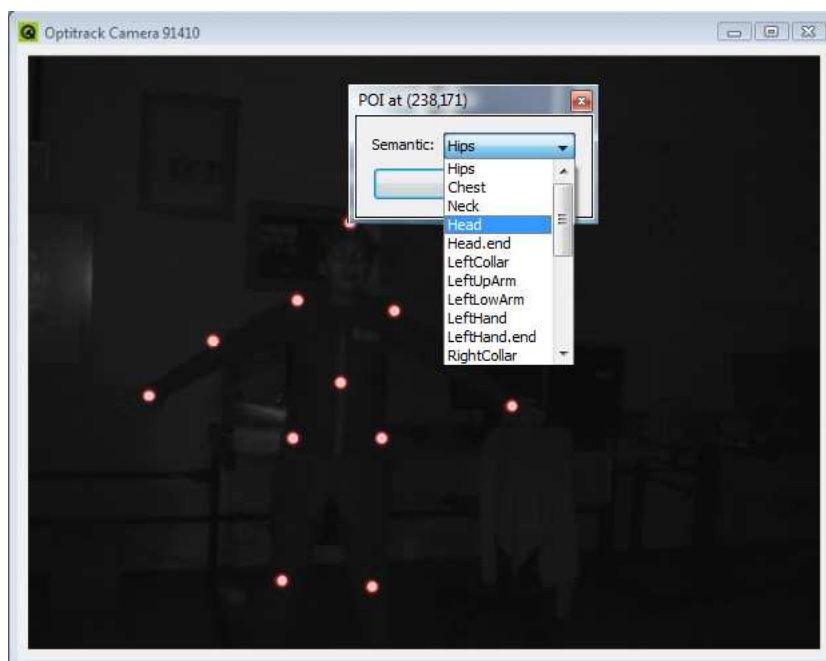
tendo em cada linha um nome. Na versão atual da aplicação desenvolvida neste projeto, as informações de relacionamento entre POIs não são utilizadas, apenas o nomes das juntas, apesar do importador considerá-las. Portanto, a atribuição de uma semântica a um POI é feita de maneira simplificada, manualmente, por meio da interface gráfica, como mostra a Figura 4.6.

## 4.4 Determinação dos Parâmetros das Câmeras

Baseado em de la Fraga & Vite Silva (2008), a abordagem deste trabalho para a determinação dos parâmetros das câmeras baseia-se na modelagem dessa tarefa como um problema de otimização, resolvido por evolução diferencial. A ideia básica é selecionar alguns dos parâmetros das câmeras como variáveis e definir uma função de custo sobre elas, utilizando informações de semântica e localização dos POIs, disponibilizadas pelos processos descritos nas Seções 4.2 e 4.3. O resultado final são matrizes de projeção estimadas para cada câmera, que minimizam a função de custo escolhida.

Os parâmetros de cada câmera  $c$  selecionados para serem variáveis do problema de otimização são: a distância focal  $f_c$ , o vetor de rotação  $R_c = [r_c^x, r_c^y, r_c^z]$  e o vetor translação  $T_c = [t_c^x, t_c^y, t_c^z]$ . Diferentemente do que foi apresentado na Seção 3.1, a rotação sofrida pelo sistema de coordenadas da câmera em relação ao do mundo é





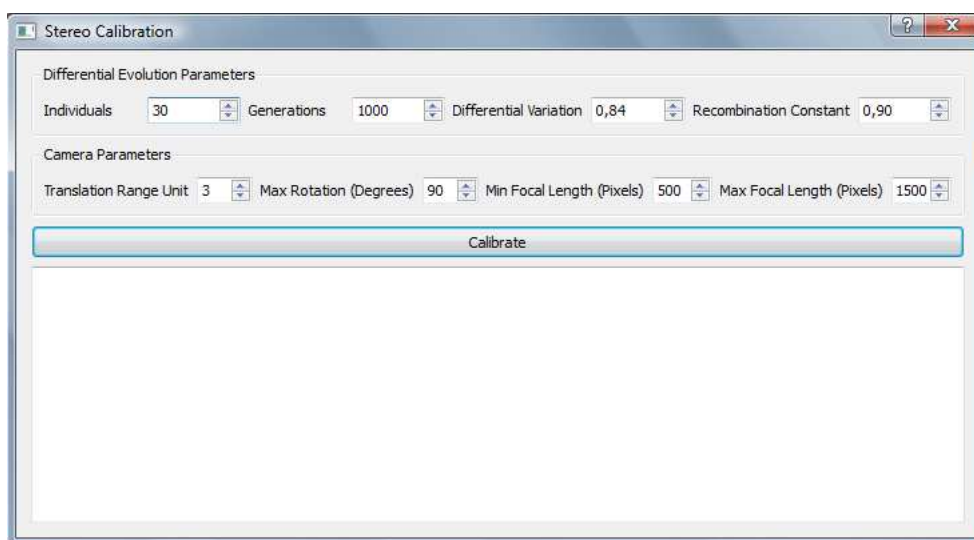
**Figura 4.6.** Seleção de Semântica de um POI.

representada neste momento por um vetor, pois é mais fácil de ser estimado do que uma matriz. Na verdade, as duas representações de rotação são intercambiáveis por meio da *Transformação de Rodrigues* (Bradski & Kaehler, 2008).

Visando simplificar o problema, na versão atual do *OpenMoCap*, é considerado que apenas duas câmeras idênticas são utilizadas em uma seção de captura de movimento. Além disso, supõe-se também que a óptica das câmeras é quase perfeita, ou seja, distorções e aberrações cromáticas das lentes são mínimas e não relevantes. Portanto, é necessário estimar somente um valor de distância focal e apenas um vetor de translação e um de rotação, já que existem somente duas câmeras iguais e uma delas é vista como a origem do sistema de coordenadas do mundo.

Os limites inferiores e superiores para as variáveis são definidos manualmente pelo usuário, por meio da interface de determinação de parâmetros das câmeras, exibida na Figura 4.7. O primeiro grupo de parâmetros controla o algoritmo de evolução diferencial. O número de indivíduos, o número de gerações, a variação diferencial e a constante de recombinação podem ser alterados.

O segundo grupo de parâmetros é onde são definidos efetivamente os limites inferiores e superiores. O campo *Translation Range Unit* define a distância máxima de translação em cada eixo de coordenada, em unidade de medida, da primeira câmera para a segunda. O campo *Max Rotation* determina a rotação máxima, em graus, do eixo de coordenadas da segunda câmera em relação a primeira. A rotação mínima não



**Figura 4.7.** Interface Gráfica para Determinação de Parâmetros das Câmeras.

precisa ser configurada, ela é fixa em zero grau. Por fim, os campos *Min Focal Length* e *Max Focal Length* especificam a distância focal mínima e máxima, respectivamente, em *pixels*.

Após o estabelecimento das variáveis e seus limites em um problema de otimização, é necessário escolher uma função de custo. Ela deve avaliar objetivamente a qualidade de uma solução estimada. Conseqüentemente, ela poderá ser usada como um fator determinante para o critério de parada do algoritmo de evolução diferencial.

Tendo em vista as informações disponíveis dos POIs, como correspondência e localização, é possível definir uma função de custo objetiva, utilizando alguns conceitos matemáticos vistos no Capítulo 3. Especificamente, a função recebe como entrada um indivíduo a ser avaliado e um conjunto de pares ordenados. Cada par é composto por POIs com mesma semântica e as respectivas coordenadas bidimensionais de seus centróides.

A partir dos valores das variáveis pertencentes ao indivíduo alvo da avaliação de custo, duas matrizes de projeção são construídas. Elas representam a orientação e posicionamento estimado das duas câmeras do sistema. Por meio dessas matrizes e um par de POIs correspondentes, e utilizando a triangulação resolvida por SVD, apresentada no Capítulo 3, projeta-se um ponto tridimensional. Após esse passo, calcula-se a reprojeção desse ponto tridimensional nos planos de imagem das câmeras, usando a Equação 3.2, também descrita no Capítulo 3. A soma das distâncias euclidianas, em *subpixels*, entre os pontos reprojetados e os centróides reais do par ordenado, é conhecida como erro de reprojeção.

O custo total de um indivíduo, isto é, de uma possível solução para o problema de otimização, é obtido por meio da soma dos erros de reprojeção resultantes do processo descrito no parágrafo anterior, aplicado a cada par ordenado de POIs. Quanto menor for esse resultado final, melhor é a solução. Ou seja, procura-se obter matrizes projetivas que minimizem o erro de reprojeção global. Caso algum dos valores das variáveis de um indivíduo ultrapasse os limites definidos pelo usuário, soma-se ao custo final uma constante de penalização, que invalida a solução.

O número de gerações é a condição de parada escolhida para esse passo de determinação de parâmetros de câmeras. Isso é feito porque ele é linearmente proporcional ao tempo de execução do algoritmo de evolução diferencial. Na prática, esse número vai sendo incrementado até que se atinja convergência dos parâmetros estimados das câmeras, dependendo do arranjo delas e da quantidade de correspondências disponíveis.

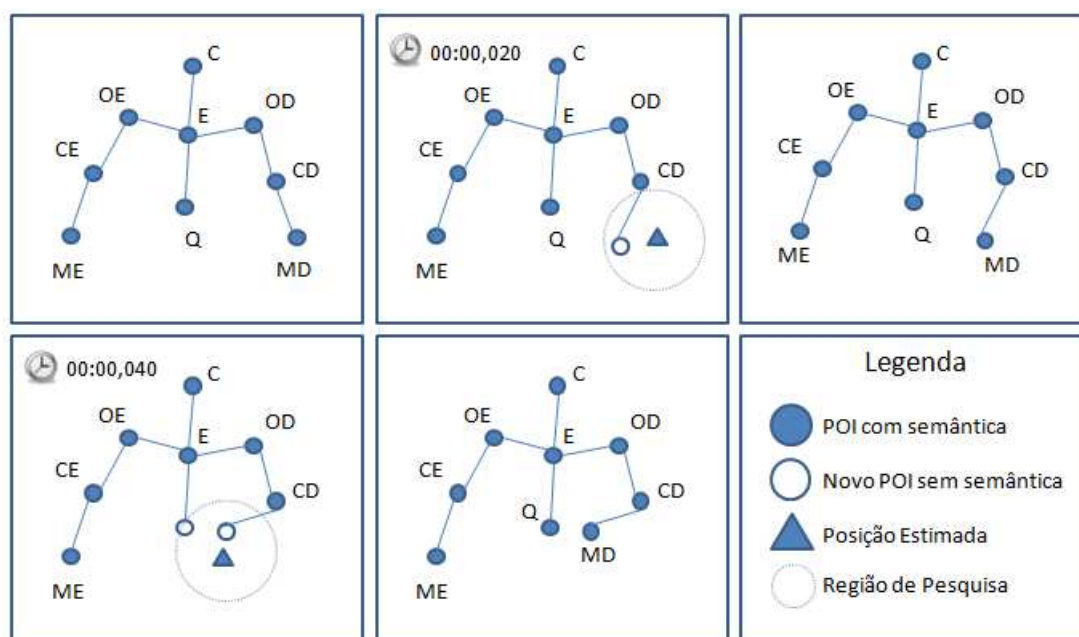
## 4.5 Rastreamento

O rastreamento é a tarefa responsável por manter as semânticas atribuídas aos POIs no tempo, tentando ser o mais robusto possível em relação às trajetórias de movimento observadas. Ele auxilia o *software* de captura de movimento a prosseguir em seu estado correto de inicialização. Por fim, é uma das etapas contínuas no fluxo de captura, processada quadro a quadro pelo *OpenMoCap*.

Na implementação desenvolvida neste trabalho, o rastreamento já começa a ser aplicado a partir do momento em que o primeiro POI tem uma semântica selecionada, não apenas quando o usuário solicita a gravação de movimento. Isso é possível pois o processamento é realizado apenas localmente, isto é, somente informações existentes no quadro anterior de uma mesma câmera são utilizadas para determinar a semântica de cada POI extraído em um novo quadro.

O estimador alfa-beta, descrito na Seção 3.6 deste texto, é utilizado para prever a próxima localização de um POI. A Figura 4.8 ilustra esquematicamente como o rastreamento é feito. As imagens marcadas com o tempo representam movimentos de POIs ocorridos na cena. Aquelas sem marca de tempo são o resultado do processamento realizado pelo rastreador implementado.

No instante em que um quadro novo é obtido, as estimativas de localização dos POIs são casadas com os novos centróides extraídos. Esse casamento atualiza a posição de um POI inicializado (com semântica) por meio da menor distância euclidiana entre a posição estimada e o centróide de um novo POI extraído. Porém, existe uma distância máxima, configurável pelo usuário, permitida entre esses dois pontos, exemplificada



**Figura 4.8.** Exemplo do Rastreamento.

pelos círculos pontilhados na Figura 4.8. Caso não exista nenhum novo centróide detectado dentro da região delimitada, a nova posição considerada é a prevista pelo estimador alfa-beta.

## 4.6 Triangulação

A triangulação é o processo responsável por estimar a posição tridimensional de um POI por meio de suas projeções nos planos de imagem das câmeras e suas respectivas matrizes de projeção. Assim como a etapa descrita na seção anterior, ela é contínua, processada quadro a quadro pela aplicação de captura de movimento. Mas, diferentemente do rastreamento, o processamento não é local, mas sim global. Logo, informações de todas as imagens obtidas pelas câmeras em um mesmo instante de tempo precisam ser reunidas para obter o resultado desejado, que são as coordenadas tridimensionais dos POIs.

Na Seções 3.2 e 3.2.1, foram apresentados os conceitos teóricos fundamentais à realização da triangulação. O exemplo apresentado nessas seções descreve exatamente a solução implementada, já que na versão atual do *OpenMoCap* apenas duas câmeras são consideradas.

Como explicado anteriormente na Seção 4.4, o método de determinação de parâmetros de câmeras utilizado neste projeto não estima matrizes projetivas com base em

pontos tridimensionais com coordenadas métricas. Ou seja, as matrizes projetivas são obtidas sem o uso de pontos distribuídos em um sistema de coordenadas tridimensional métrico previamente conhecido, o *ground truth*. Portanto, o resultado final alcançado pela triangulação implementada é determinado a menos de uma fator de escala, o que não é ideal para a análise de erros em experimentos. Apesar disso, as distâncias relativas entre os pontos que formam a estrutura alvo de captura são mantidas, o que é geralmente suficiente para certos tipos de aplicação como animação e controle de personagens.

## 4.7 Geração de Arquivo de Saída

A transformação dos dados obtidos pelas sucessivas triangulações em um formato legível por outros programas é a última tarefa necessária para completar o fluxo de captura de movimento. Ela é essencialmente um processo único de conversão, mas é atualizado continuamente a cada triangulação feita. Na aplicação desenvolvida neste trabalho, após solicitação do usuário de interrupção de uma seção de gravação de movimento, um arquivo de saída é gerado.

Na Seção 3.7, foram apresentados os dois tipos possíveis de dados gerados por uma aplicação de captura de movimento e os diversos modelos de arquivos existentes foram citados. Tendo em vista que o resultado produzido pelo processo de triangulação é uma nuvem de pontos com semântica associada, a versão atual do *OpenMoCap* suporta apenas saída das informações coletadas em um formato padrão não hierárquico de captura de movimento, o TRC, desenvolvido pela Motion Analysis Corporation (2009). A Figura 4.9 exibe um fragmento de um arquivo exemplo gerado pela aplicação.

O arquivo de saída é dividido basicamente em duas partes, cabeçalho e dados. Na primeira, existem informações sobre o nome do arquivo gerado, a quantidade de dados obtidos na captura por segundo, a quantidade de imagens obtidas por segundo pelas câmeras, o número total de quadros processados, o número total de marcadores presentes e a unidade de medida utilizada. Particularmente, na versão atual do *OpenMoCap*, esse último parâmetro citado é sempre configurado para milímetros, apesar de não ter significado real, já que os pontos tridimensionais são recuperados sem uma escala definida.

Ainda no cabeçalho existem informações úteis para a identificação de alterações ou processamentos posteriores à captura. Essas são a taxa original de amostragem de dados, o índice original do quadro inicial e o número original total de quadros processados. Finalmente, as últimas informações existentes na primeira parte do arquivo são

PathFileType 4 (X/Y/Z) output.trc									
DataRate	CameraRate	NumFrames	NumMarkers	Units	OrigDataRate	OrigDataStartFrame	OrigNumFrames		
25.00	25.00	100	10	mm	25.00	0	100		
Frame#	Time	A			A.end	B			
		X1	Y1	Z1	X2	Y2	Z2	X3	Z3
1	0.00	-0.963505	0.567625	5.21397	-1.15364	-0.216878	4.87087	-0.270564	-0.7271 4.46509
2	0.04	-0.966132	0.573169	5.21397	-1.15364	-0.216878	4.87087	-0.270564	-0.7271 4.46509
3	0.08	-0.965992	0.578647	5.21397	-1.15364	-0.216878	4.87087	-0.270564	-0.7271 4.46509
4	0.12	-0.963653	0.582631	5.21397	-1.15364	-0.216878	4.87087	-0.270564	-0.7271 4.46509
5	0.16	-0.966262	0.588164	5.21397	-1.15364	-0.216878	4.87087	-0.270564	-0.7271 4.46509 ...
6	0.20	-0.968691	0.593558	5.21397	-1.15364	-0.216878	4.87087	-0.270564	-0.7271 4.46509
7	0.24	-0.968542	0.599021	5.21397	-1.15364	-0.216878	4.87087	-0.270564	-0.7271 4.46509
8	0.28	-0.966199	0.602946	5.21397	-1.15364	-0.216878	4.87087	-0.270564	-0.7271 4.46509
9	0.32	-0.97114	0.604555	5.21397	-1.15364	-0.216878	4.87087	-0.270564	-0.7271 4.46509

Figura 4.9. Exemplo de Arquivo Gerado no Formato TRC.

títulos para as colunas de dados de captura e semânticas dos POIs.

A segunda parte do arquivo é usualmente composta por muitas linhas de coordenadas tridimensionais de POIs, organizadas de acordo com o cabeçalho descrito previamente. Cada uma dessas linhas contém ainda o índice do quadro em que as coordenadas foram calculadas e o respectivo instante de tempo. Por fim, apesar de ser um formato simples, ele é importado nativamente (sem *plugins* ou *scripts*) pelo 3ds Max, da empresa Autodesk (2009).

## 4.8 Parâmetros do *OpenMoCap*

A lista a seguir resume todos os parâmetros existentes na aplicação desenvolvida neste trabalho que devem ser determinados para começar uma seção de captura de movimento. É importante ressaltar que a maioria deles deve ser determinada experimentalmente para atingir melhores resultados.

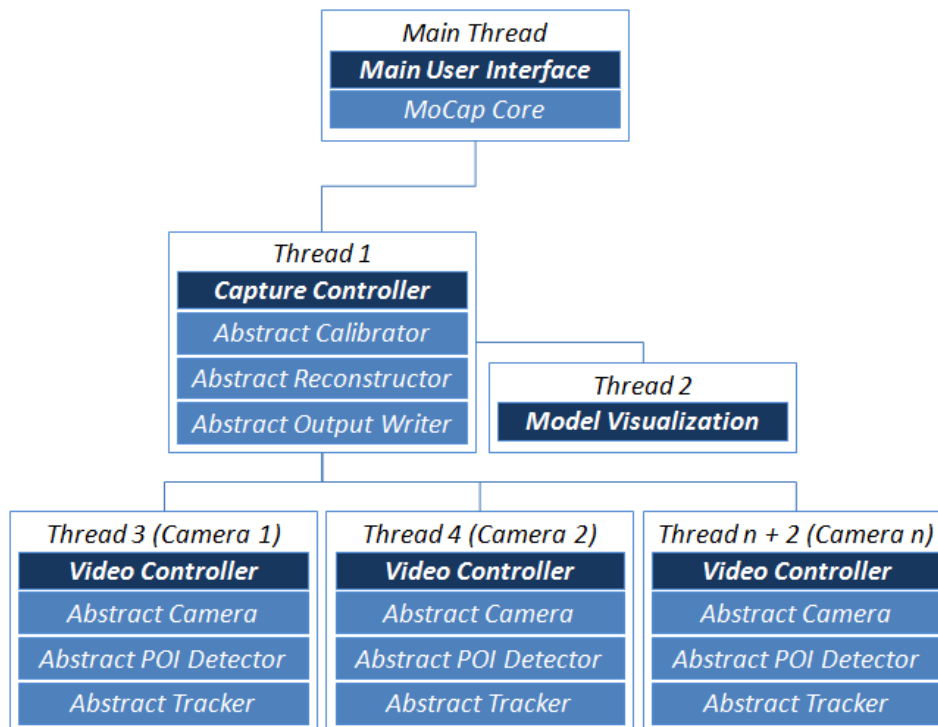
- Semântica dos marcadores em cada câmera.
- Limiar de binarização para uma separação inicial dos marcadores do fundo da cena.
- Limiar de área mínima e máxima de um marcador em relação a área máxima de um quadro.
- Alfa e Beta para o estimador usado no rastreamento.
- Distância máxima em *pixels* no rastreador para determinar área de pesquisa.

- Limiares para a estimação dos parâmetros das câmeras por evolução diferencial.

## 4.9 Arquitetura da Aplicação *OpenMoCap*

O *OpenMoCap* foi construído visando alcançar todos os objetivos propostos no Capítulo 1, empregando bons princípios de construção de *software*, descritos em McConnell (2004) e em Kernighan & Pike (1999). Portanto, diversas decisões de arquitetura e implementação foram tomadas e são apresentadas a seguir.

A Figura 4.10 ilustra a arquitetura definida para a aplicação por meio de um diagrama de módulos. O *software* e o código foram desenvolvidos totalmente em inglês, almejando universalizar seu uso e o desenvolvimento de extensões. A separação dos módulos por fluxos de execução (do inglês *threads*) foi feita para aproveitar a tendência atual da indústria de processadores de produzir *chips* com cada vez mais núcleos. Tarefas empreendidas em tempo real, como a detecção de POI, o rastreamento, a triangulação e a visualização, puderam ser paralelizadas.

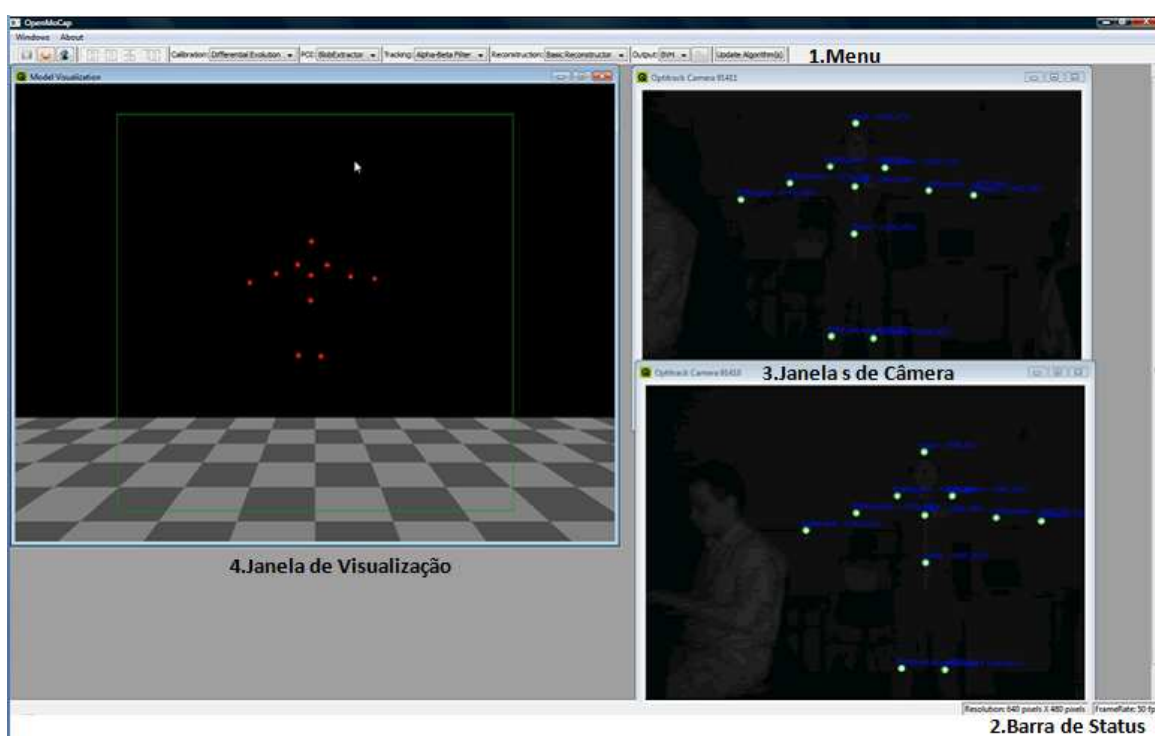


**Figura 4.10.** Diagrama da Arquitetura do *OpenMoCap*.

O fluxo principal de execução (*Main Thread*) é composto pelo núcleo da aplicação (*MoCap Core*) e pela interface gráfica principal (*Main User Interface*). O núcleo é responsável por inicializar corretamente todos os outros fluxos de execução e seus

módulos e por disponibilizar também uma ligação entre eles por meio dos controladores de captura e de vídeo. Além disso, é um repositório central de informações, como qual é a configuração das câmeras conectadas, quais são os algoritmos disponíveis para realização do fluxo de captura e quais são as semânticas possíveis para os POIs.

A interface gráfica principal é ilustrada pela Figura 4.11, obtida do *OpenMoCap* em execução. Ela é responsável por exibir dados de captura e receber solicitações do usuário, acionando funções específicas dos controladores, necessárias à gravação de movimento. Existem basicamente quatro componentes de interação com o usuário: o menu, a barra de *status*, as janelas de câmera e a janela de visualização.



**Figura 4.11.** Interface Gráfica Principal do OpenMoCap.

O menu permite, por meio de botões, o começo e o término de uma seção de captura e a estimação dos parâmetros das câmeras. Ele informa também o tempo de captura total e os algoritmos utilizados. Por outro lado, a barra de *status* exibe apenas a resolução das câmeras ligadas ao *software* e a taxa de quadros sendo processada por segundo. As janelas de câmera exibem as imagens obtidas por cada dispositivo existente no sistema e permitem a seleção de semântica dos POIs detectados. A janela de visualização exibe uma prévia em tempo real do movimento que está sendo gravado, por meio de um fluxo separado composto pelo módulo de visualização (*Model Visualization*). O modelo de interface de múltiplos documentos (do inglês *Multiple Document*



*Interface*) foi adotado por proporcionar um espaço comum e flexível para o uso das janelas existentes.

Cada uma das janelas de câmera é processada exclusivamente por um fluxo de execução, coordenado por um controlador de vídeo (*Video Controller*). Por sua vez, cada um desses controladores possui uma instância de câmera, do algoritmo de detecção de POI e do algoritmo de rastreamento, que são executados sequencialmente. Na Figura 4.10 isso é representado pelos módulos *Abstract Camera*, *Abstract POI Detector* e *Abstract Tracker*. Eles definem uma interface comum e a mais genérica possível para as câmeras e os algoritmos de processamento bidimensionais, citados anteriormente.

Esse conceito de módulos abstratos que definem interfaces é muito importante para garantir a extensibilidade da aplicação, um dos objetivos propostos neste trabalho. Caso seja usado um novo modelo de câmera e esse equipamento não for compatível com a implementação atual, basta desenvolver funcionalidades específicas, como a obtenção de quadros e a mudança de resolução seguindo o padrão do módulo abstrato de câmera para aproveitar todo o fluxo já existente. Outro exemplo vantajoso é a possibilidade de substituir o algoritmo de detecção de POI por um baseado em reconhecimento de partes do corpo ao invés de intensidades de regiões, isto é, fazer com que o sistema possa capturar movimento sem marcadores. Portanto, é possível realizar alterações de otimização e novas funcionalidades, pontualmente e incrementalmente.

O último fluxo de execução ainda não descrito, presente na Figura 4.10, é o composto pelo controlador de captura (*Capture Controller*). Ele é o principal responsável pelo funcionamento correto da aplicação e executa efetivamente uma seção de captura, gerenciando e processando as informações produzidas pelos controladores de vídeo e atualizando o fluxo do visualizador. Baseado no mesmo princípio de módulos abstratos, o controlador de captura possui uma instância de um algoritmo de estimação de parâmetros de câmeras (*Abstract Calibrator*), uma de um algoritmo de reconstrução (*Abstract Reconstructor*) e uma de um algoritmo para gerar saída de dados (*Abstract Output Writer*)

Até o momento foram discutidas decisões de mais alto nível de arquitetura, relacionadas principalmente aos conceitos de abstração, generalidade, separação de interesses e desenvolvimento incremental. Porém, decisões de mais baixo nível, ligadas à implementação em si, também foram feitas para garantir o aproveitamento de bibliotecas de código livre já existentes, o padrão de codificação e a modularidade.

Dada a natureza da aplicação construída, uma biblioteca eficiente e madura de visão computacional foi especialmente útil para seu desenvolvimento. A *OpenCV*, atualmente mantida pela Willow Garage (2009), foi a biblioteca selecionada desse tipo. Ela foi escrita inicialmente na linguagem de programação C e C++ para exibir a

performance dos processadores da Intel Corporation (2009), sua criadora.

Infelizmente, os módulos existentes na *OpenCV* para o uso de câmeras não são robustos para mais de duas câmeras e não foram desenvolvidos de maneira orientada a objetos. Como um dos objetivos deste trabalho é construir uma aplicação extensível, uma biblioteca nova de captura de vídeo foi utilizada, a *videoInput* (Watson, 2009). Ela suporta até vinte câmeras por meio da interface *DirectShow* (Microsoft Corporation, 2009), usada em um grande número de dispositivos de entrada de vídeo. É interessante ressaltar que, atualmente, essa biblioteca está sendo integrada à *OpenCV*, em substituição aos módulos existentes para tratamento de entrada de vídeo.

Lamentavelmente, foi verificado que a interface *DirectShow* das câmeras *OptiTrack FLEX:V100* somente suporta um único dispositivo por vez, ou seja, por meio dela não é possível processar e obter imagem de mais de uma câmera conectada ao sistema em um mesmo instante de tempo. Por esse motivo, foi necessário a inclusão da API (do inglês *Application Programming Interface*) própria das câmeras no código para testar e avaliar o fluxo de captura de movimento. Mesmo assim, a biblioteca *videoInput* foi mantida para o futuro uso de câmeras genéricas.

A interface da aplicação com o usuário foi totalmente construída usando a biblioteca *Qt* (Trolltech, 2009). Ela é uma biblioteca de código livre madura, rápida e portátil para diversos sistemas operacionais. Além disso, é muito simples, o que ajudou no tempo de prototipação e construção da interface do *OpenMoCap*.

A integração dessas diversas ferramentas aqui citadas foi complexa. Muitas incompatibilidades tiveram que ser levantadas e tratadas. Os principais problemas encontrados foram as diferenças nos modelos de *threads* das bibliotecas de entrada de vídeo e a baixa performance das *threads* existentes em *Qt*. A correção deles pode ser feita por meio da seleção de um modelo de *thread* compatível com as duas bibliotecas de vídeo e o uso das *threads* da própria API do *Windows*.

Finalmente, a estrutura final em pacotes da aplicação construída é descrita sucintamente na próxima Seção (4.9.1).

### 4.9.1 Pacotes

**Pacote Raiz** contém apenas a classe núcleo da aplicação, a *Mocap*. Ela é responsável pela inicialização e configurações iniciais do programa. Além disso, é um elo de ligação entre as classes de visualização, de modelo, de algoritmos e de controladores.

**Pacote Calibration** contém todas as classes relacionadas à determinação ou estimação dos parâmetros de câmeras. Especificamente, possui a classe *AbstractCalibra-*

*tor*, que define uma interface comum para a implementação de novos algoritmos dessa categoria.

**Pacote *Controllers*** é composto por duas classes importantes para o correto funcionamento da aplicação. Elas são a realização dos módulos de controle de captura (*CaptureController*) e de controle de vídeo (*VideoController*). Assim, são responsáveis por orquestrar todos os algoritmos envolvidos para a gravação de uma seção de movimento.

**Pacote *Entities*** contém classes modelo relacionadas diretamente à captura óptica de movimento, mas não a uma etapa específica. Conceitos como câmera, ponto de interesse, esqueleto e junta são modelados. Especificamente, possui a classe *AbstractCamera*, base para adição de suporte a novas câmeras, principalmente quando não são compatíveis com as bibliotecas já utilizadas.

**Pacote *Enums*** contém classes que são enumerações robustas quando comparadas às disponíveis na linguagem de implementação. Representam os tipos de algoritmos existentes e estados da captura e dos vídeos, além de estado de retorno de funções.

**Pacote *GUI*** contém classes exclusivamente relacionadas à interface com usuário. Elas capturam solicitações do operador do *software*, como a seleção da semântica de POIs e oferecem visualizações da seção de captura.

**Pacote *Input*** contém classes responsáveis por ler semântica e estrutura de arquivos de captura de movimento. Na versão atual da aplicação, existe somente um leitor de BVH. O suporte a novos tipos de entrada de arquivos deve ser feito por meio da implementação de novas classes neste pacote.

**Pacote *Output*** contém classes responsáveis por escrever dados em arquivos de uma seção de captura de movimento. Na versão atual da aplicação, existe somente uma classe que gera arquivos no formato padrão TRC. Especificamente, a classe *AbstractOutputWriter* fornece uma interface comum para o desenvolvimento de novas classes de saídas de dados.

**Pacote *Reconstruction*** contém todas as classes relacionadas à reconstrução e triangulação dos POIs. Especificamente, possui a classe *AbstractReconstructor*, que define uma interface comum para a implementação de novos algoritmos dessa categoria.

**Pacote *Tracking*** contém todas as classes relacionadas à detecção de POIs e seu rastreamento. Especificamente, possui as classes *AbstractPOIFinder* e *Abstract-*

*Tracker*, que definem interfaces comuns para implementação de novos algoritmos das respectivas categorias.

**Pacote *Utils*** contém classes úteis à aplicação, mas não relacionadas diretamente ao fluxo de captura. Elas abstraem detalhes de *threads*, da representação de imagens, de temporizadores, de registro e de conversões de tipos de dados. Portanto, somente simplificam o uso dessas funcionalidades.

## 4.10 Considerações

Neste capítulo, foram apresentadas as soluções desenvolvidas neste trabalho para a realização completa do fluxo de captura de movimento. Cada etapa necessária à execução de uma seção de gravação de movimento foi explicada detalhadamente e relacionada a seus respectivos conceitos teóricos. Além disso, a arquitetura definida para a aplicação foi descrita sucintamente por uma abordagem *top-down*, isto é, de um nível mais alto de abstração, seus módulos, até o nível mais baixo de classes e linguagem de implementação. O próximo capítulo apresentará alguns experimentos avaliados qualitativamente e outros quantitativamente, examinando os resultados obtidos pelo *OpenMoCap*.

# Capítulo 5

## Resultados Experimentais

Neste capítulo, são apresentados os experimentos desenvolvidos para avaliar o fluxo de captura de movimento executado pelo aplicativo construído. A solução comercial da *NaturalPoint* escolhida para comparação de alguns aspectos é descrita brevemente, assim como todo o *hardware* e acessórios utilizados nos testes realizados. Por fim, os resultados quantitativos e qualitativos obtidos são discutidos na ordem das etapas da captura de movimento.

### 5.1 *NaturalPoint*

A *NaturalPoint* é uma empresa norte americana fundada em 1997, com o intuito de desenvolver e fornecer soluções comerciais de alta qualidade de rastreamento óptico e dispositivos de interação humano-computador alternativos ao mercado mundial. Atualmente, ela comercializa três linhas de produtos, a *TRACKIR*, a *SmartNav* e a *OptiTrack*.

As duas primeiras são métodos alternativos de entrada para computador por meio de rastreamento de movimento de marcadores colocados usualmente em um boné ou capacete vestido pelo usuário. A *TRACKIR* é utilizada para o controle de jogos, enquanto a *SmartNav* foi desenvolvida para auxiliar deficientes na utilização de um computador sem as mãos. Finalmente, a *OptiTrack* é a linha de produtos focada na captura e rastreamento óptico de movimento e, portanto, a que mais se relaciona a este trabalho, como já comentado no Capítulo 2.

### 5.1.1 *NaturalPoint Tracking Tools*

Dentre os três *softwares* existentes na linha *OptiTrack*, o aplicativo usado neste trabalho para comparar o resultado de etapas, como detecção de centróides, rastreamento e estrutura tridimensional, foi o *Tracking Tools*. Os outros aplicativos da linha não oferecem maneiras fáceis de comparar essas etapas com o *OpenMoCap*, já que possuem foco somente na produção de dados hierárquicos, de esqueleto e de face, em uma seção de captura.

O *Tracking Tools* permite visualizar os centróides, em *subpixels*, da projeção dos POIs em cada câmera conectada ao sistema. Ele possibilita também a exportação no formato CSV (do inglês *Comma-separated values*) de coordenadas tridimensionais de translação dos POIs rastreados em tempo real. Por fim, não há necessidade de rotulá-los ou adicioná-los a uma estrutura previamente definida para gravar movimento.

A aplicação somente funciona com câmeras da *NaturalPoint* (o modelo usado nos experimentos é descrito na Seção 5.2.1) e necessita de no mínimo três desses dispositivos para capturar movimento. Além disso, as câmeras utilizadas devem ser do mesmo modelo. Essas restrições existem porque o *Tracking Tools* delega parte do fluxo de captura, particularmente a detecção de centróides de POIs e o rastreamento, para o *hardware* dedicado das câmeras. Isso torna possível gravar movimento a uma taxa de cem quadros por segundo.

Infelizmente, esse programa comercial não permite realizar sua cadeia de processamento em vídeos previamente gravados. Isso prejudica as análises feitas neste trabalho, já que uma comparação ideal não é possível. Os resultados obtidos pelas duas abordagens, em sua maioria, não foram obtidos sobre uma cena exatamente igual, mas bastante semelhante, como descrito nos experimentos.

O passo inicial para capturar movimento em uma cena com esse *software* comercial é estimar os parâmetros das câmeras. Esse procedimento é feito movendo um marcador dentro do campo de visão das mesmas por alguns minutos. Posteriormente, um processo de otimização é executado e a rotação e translação entre os dispositivos de entrada de vídeo é estabelecida. Finalmente, deve-se mostrar um objeto (fornecido também pela *NaturalPoint*) com dimensões conhecidas e que forme um plano para determinar a origem do sistema de coordenadas e uma relação métrica de escala. Feita essa calibração, é necessário apenas solicitar, por meio da interface, o começo da gravação de movimento.

## 5.2 Hardware Comum

Os experimentos realizados neste trabalho foram feitos utilizando um conjunto de marcadores reflexivos e dois tipos de equipamentos comuns, descritos a seguir.

### 5.2.1 Câmeras *NaturalPoint OptiTrack FLEX:V100*

As câmeras usadas na realização dos experimentos são da *NaturalPoint*, o modelo é o *OptiTrack FLEX:V100*. Elas são compostas por um sensor CMOS preto e branco com resolução de 640 por 480 *pixels*, capaz de adquirir imagens a uma velocidade de 100Hz. Possuem ainda um processador de imagem embutido, que realiza operações de limiarização e rastreamento em tempo real. Além disso, na sua carcaça de metal existem LEDs que emitem luz no comprimento de onda infravermelho, sincronizados com o obturador. Por fim, a conectividade e transferência de dados é feita por uma interface USB (do inglês *Universal Serial Bus*).

### 5.2.2 Computador

O computador utilizado pelas duas aplicações nos experimentos possui a configuração exibida na Tabela 5.1. Seu processador possui quatro núcleos, permitindo exercitar a arquitetura com múltiplos fluxos implementada pelo *OpenMoCap*. Os outros componentes da configuração contribuem para que não haja nenhum fator severamente limitante na avaliação dos testes realizados.

**Tabela 5.1.** Configuração do Computador Utilizado nos Experimentos.

Processador	Intel Core 2 Quad Q6600 2,4GHz Cache L2 8MB FSB 1066MHz
Memória Principal (RAM)	4GB DDR2 800Mhz
Placa Gráfica	MSI Geforce 8800 GTX 768MB GDDR3 384 bits
Disco Rígido	Samsung 250GB 7200RPM Cache 16MB
Controladores USB 2.0	8
Sistema Operacional	Microsoft Windows Vista SP2 32 bits

## 5.3 Experimentos e Resultados

### 5.3.1 Estabilidade e Precisão dos Centróides 2D

A estabilidade dos centróides bidimensionais dos POIs e seu correto posicionamento são aspectos importantes em um sistema de captura de movimento. Um ponto estático na cena deve, idealmente, não apresentar flutuações em suas coordenadas no tempo. Caso ele apresente esse problema, o movimento gravado sofrerá com “saltos” com o alvo de captura parado. Além disso, o posicionamento incorreto dos centróides afeta diretamente a reconstrução, já que a profundidade de um ponto depende diretamente dessa medida. Portanto, todos esses erros propagam para as etapas posteriores do fluxo de captura e diminuem a precisão do resultado final.

As câmeras utilizadas e o ambiente em que a gravação de movimento ocorre são, usualmente, os grandes responsáveis pela introdução desses efeitos indesejados. Felizmente, os dispositivos *OptiTrack FLEX:V100* utilizados na construção do *OpenMoCap* são de ótima qualidade e a combinação de LEDs infravermelhos, filtro infravermelho e marcadores reflexivos ajudam na quase completa remoção dos ruídos. Apesar disso, ainda existem situações em que eles estão presentes.

Este experimento visa aferir a qualidade do módulo de detecção de POI desenvolvido neste trabalho, comparando-o diretamente com o resultado obtido pelo *hardware* da câmera. Para isso, as configurações do algoritmo da câmera (*hardware*) foram igualladas às do detector de POI (*software OpenMoCap*), como mostra a Tabela 5.2.

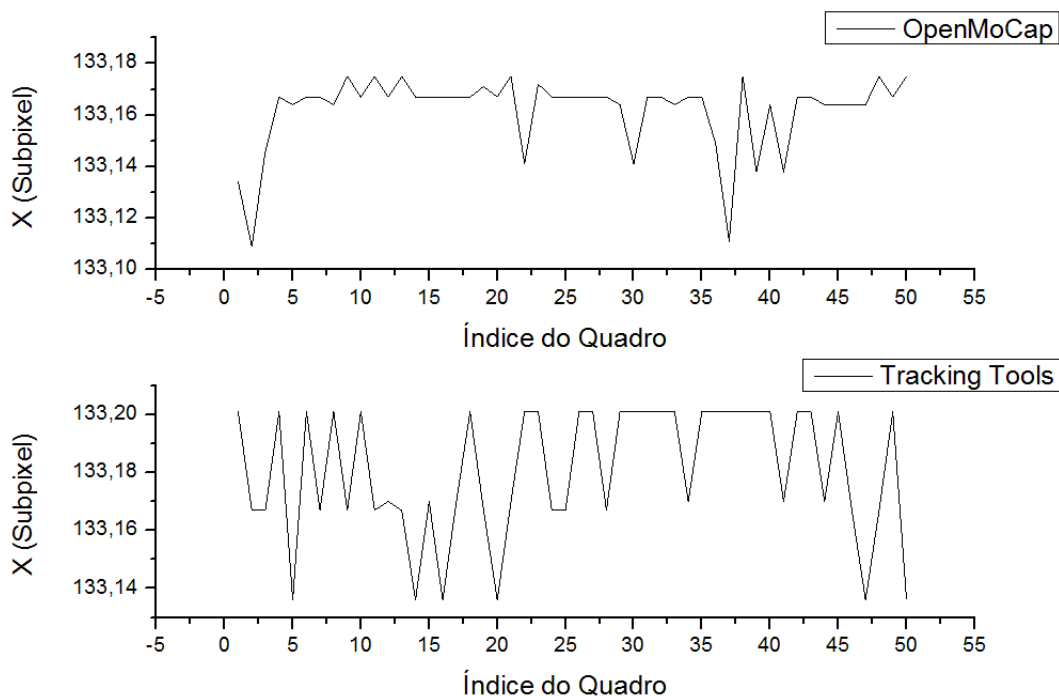
**Tabela 5.2.** Configuração para Detecção de POIs.

<i>Parâmetro</i>	<i>Valor</i>
Resolução	640 x 480 <i>pixels</i>
Velocidade de Processamento	25 quadros por segundo
Limiar de Intensidade	230
Área mínima	0,005% da área total da imagem
Área máxima	0,400% da área total da imagem

Um marcador foi colocado em uma posição específica sobre um suporte apoiado no chão de um sala com luz fluorescente comum. A localização foi escolhida por representar um local em que havia ruído, já que na maioria das situações ele não ocorre. Uma mesma câmera, montada em um tripé para evitar qualquer deslocamento, foi utilizada pelas duas abordagens na captura de quadros dessa cena. A Figura 5.1



ilustra, por meio de gráficos, o ruído gerado em 50 quadros nas coordenadas X do centróide, determinado pelo *OpenMoCap* e pelo algoritmo da câmera.



**Figura 5.1.** Gráficos das Coordenada X dos Centróides Obtidos pelo *OpenMoCap* e pelo *Tracking Tools*.

Para tentar caracterizar o ruído gerado, o teste de normalidade de *Shapiro-Wilk* (Boslaugh & Watters, 2008) foi aplicado sobre os centróides calculados pelas duas abordagens desse ponto estático durante 40 segundos ou 1000 quadros. Seu resultado foi negativo para todas as coordenadas, ou seja, os valores de ruído presentes não seguem uma distribuição normal. Logo, uma análise estatística de ordem foi feita e não uma de momento. A Tabela 5.3 sumariza essa análise.

**Tabela 5.3.** Análise Estatística Comparativa de Detecção de Centróides 2D.

em <i>subpixels</i>	<i>OpenMoCap</i>	<i>Câmera</i>
Percentil 5 de X	133,164	133,136
Mediana de X	133,167	133,167
Percentil 95 de X	133,201	133,201
Percentil 5 de Y	352,826	352,826
Mediana de Y	352,838	352,844
Percentil 95 de Y	352,906	352,856

Os valores dessa tabela indicam que a diferença entre as coordenadas dos centróides determinadas pelo algoritmo desenvolvido no *OpenMoCap* e o algoritmo utilizado pelo *Tracking Tools* é mínima, sendo da ordem de centésimos de *pixels*. A mediana, que determina qual é o valor mais comum em uma amostra de dados, é igual na coordenada X e muito similar na coordenada Y para os métodos. Finalmente, os intervalos de percentis mostrados sugerem que o *OpenMoCap* tem ruído menor na coordenada X do que o *Tracking Tools*, porém possui ruído maior na coordenada Y.

### 5.3.2 Rastreamento 2D

A avaliação do módulo de rastreamento construído neste trabalho foi feita por meio da comparação de trajetórias. O movimento escolhido para essa análise foi o de um pêndulo simples. Isso foi feito porque o caminho ideal percorrido pelo objeto de interesse nessa situação pode ser facilmente descrito por um arco de uma circunferência.

O experimento é descrito a seguir. Uma linha de náilon, de aproximadamente 1m, teve uma de suas extremidades atada a um suporte metálico fixo. A sua outra extremidade foi amarrada a um peso de aproximadamente 50 gramas, para manter-se esticada. No centro desse peso, um marcador foi preso. Por fim, uma câmera foi colocada a uma distância de 1,5m do suporte e, aproximadamente no meio da trajetória esperada do movimento (A coordenada X do centróide do POI na imagem era 318,102).

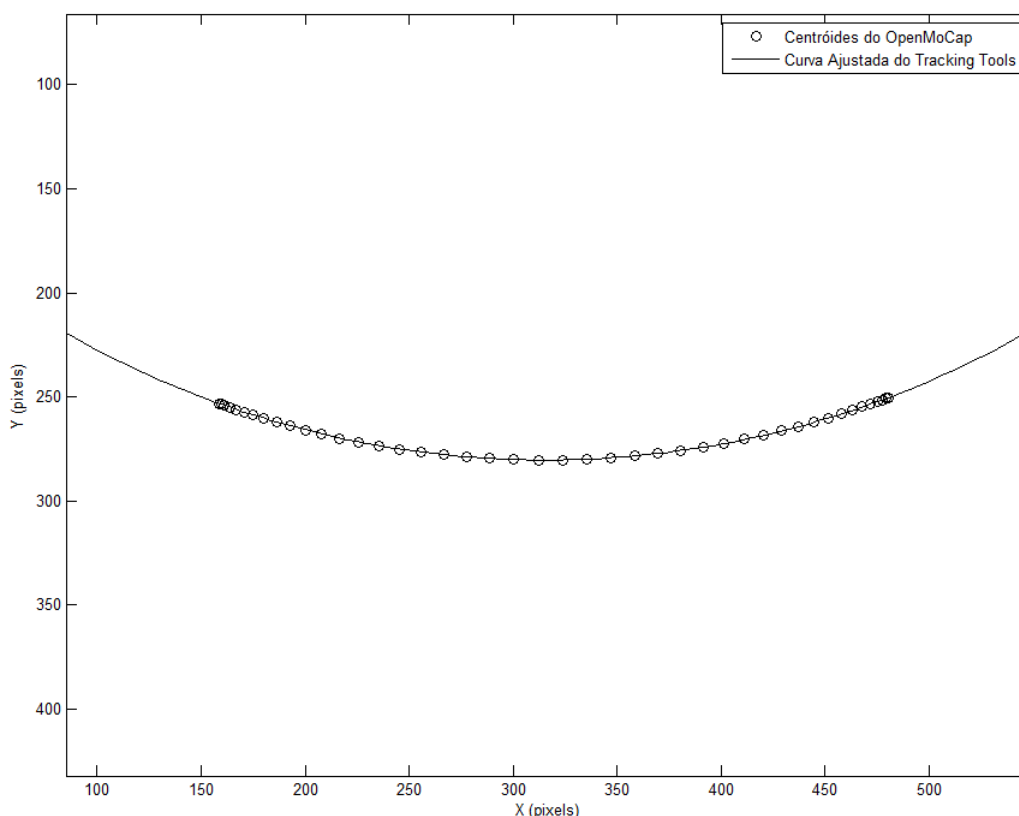
O *OpenMoCap* e *Tracking Tools* foram configurados da mesma forma que o experimento anterior, como mostra a Tabela 5.2. A única diferença foi a velocidade em que os quadros foram obtidos da câmera, neste experimento a 50Hz. Além disso, os parâmetros alfa e beta do rastreador do *OpenMoCap* foram fixados em 1,0 e 0,8, respectivamente.

Devido a dificuldade de se realizar o movimento do pêndulo exatamente em um plano paralelo ao da câmera, uma série de quatro períodos foi capturada pelo *Tracking Tools* para ajustar uma curva, descrevendo a trajetória do POI, considerada como verdadeira. Os parâmetros encontrados são exibidos na Tabela 5.4.

**Tabela 5.4.** Parâmetros Encontrados no Ajuste de Curva da Trajetória do Pêndulo no *Tracking Tools*.

<i>Parâmetro</i>	<i>Valor (pixels)</i>
Coordenada X do Centro	315,769
Coordenada Y do Centro	-186,891
Comprimento do Pêndulo	467,290

Considerando que existe atrito e o posicionamento da câmera não é perfeito, a coordenada X do centro encontrada, mostra que o movimento é bem caracterizado, tendo em vista a proximidade com a coordenada do sistema em repouso, citada previamente. Como não é possível compartilhar a câmera entre os dois programas comparados em um mesmo instante de tempo, capturas intercaladas do movimento foram feitas. A Figura 5.2 mostra um gráfico com a curva ajustada, considerada o *ground truth* da trajetória, e um período aleatório obtido pelo *OpenMoCap* em aproximadamente um segundo.



**Figura 5.2.** Comparação de Trajetórias do *Tracking Tools* e *OpenMoCap*.

Duas observações são particularmente interessantes nesse gráfico. A primeira é em relação a qualidade do movimento capturado pelo *OpenMoCap*. A trajetória obtida pela abordagem desenvolvida neste trabalho é exatamente a esperada de um pêndulo simples. Nas extremidades, a velocidade do pêndulo é menor, portanto existe um agrupamento dos centróides. Já quando o pêndulo aproxima-se da posição de repouso, mais veloz é seu movimento, logo a distância entre centróides vai aumentando gradativamente. Além disso, a simetria está presente no caminho percorrido, indicando preliminarmente que a arquitetura do *OpenMoCap* está bem sincronizada e é capaz de

capturar movimento com suavidade, sem saltos inesperados.

A segunda observação é quantitativa e está relacionada ao deslocamento existente entre os centróides obtidos pelo *OpenMoCap* e a curva ajustada com os dados da solução comercial. Uma análise estatística foi feita sobre esse erro e o resultado é revelado na Tabela 5.5. Considerando todo o ruído existente, os valores comprovam qualidade semelhante ao *Tracking Tools*.

**Tabela 5.5.** Deslocamento Entre os Centróides Obtidos pelo *OpenMoCap* e a Trajetória Ajustada do Pêndulo pela Solução Comercial.

<i>Erro</i>	<i>Valor(es) (pixels)</i>
Mínimo e Máximo	0,004 e 0,254
Mediana	0,087
Percentil 85	0,125

### 5.3.3 Estimação de Parâmetros de Câmeras

Empiricamente, foi verificado que a estimação de parâmetros de câmeras desenvolvida neste trabalho funciona bem somente em situações em que os dois dispositivos de entrada de vídeo estão praticamente em um mesmo plano e a rotação relativa entre eles é pequena. A principal razão para isso acontecer é o número limitado de correspondências de pontos disponível para o processo de otimização usando evolução diferencial.

Aparentemente, o aplicativo comercial *Tracking Tools* utiliza um mecanismo semelhante de estimação de parâmetros de câmeras. A diferença está na maneira de aquisição dos pontos correspondentes, que servem como restrições para o problema de otimização. Como citado na Seção 5.1.1, a medida que o usuário movimenta um único marcador durante determinado tempo, vários pontos correspondentes estão sendo coletados. Portanto, com uma quantidade grande de pontos, a convergência para uma solução pode ser mais facilmente atingida.

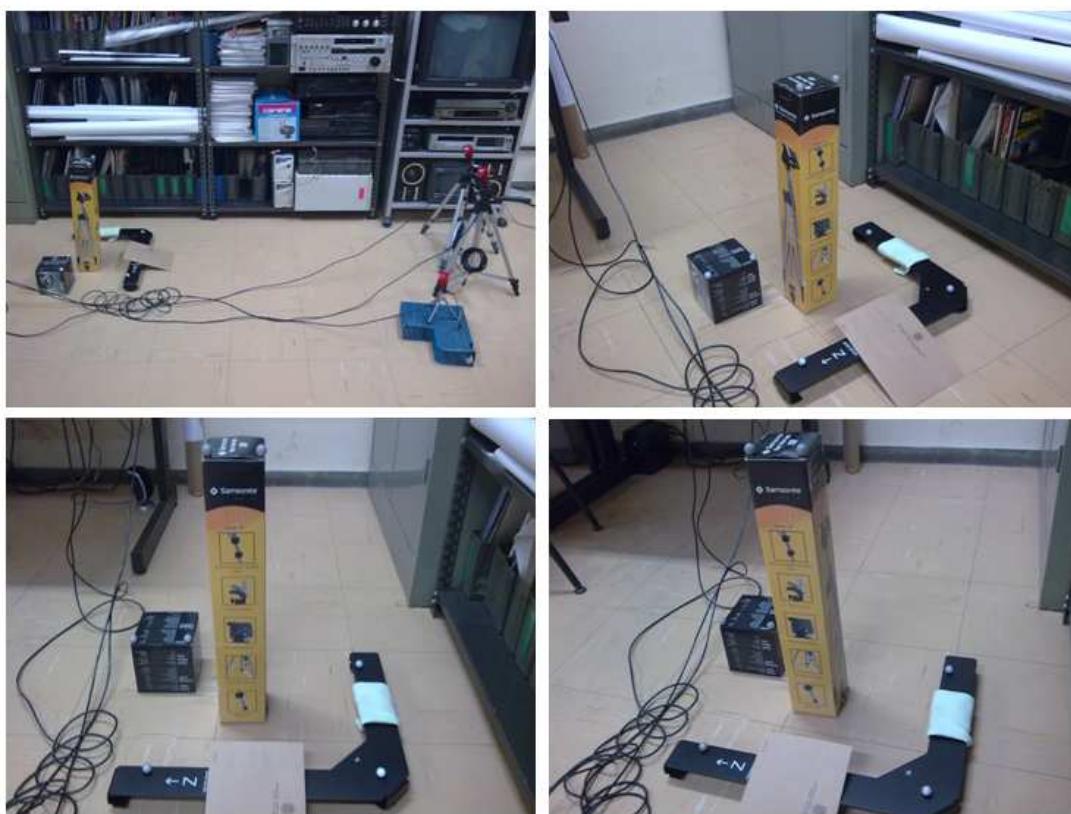
### 5.3.4 Estrutura 3D

A comparação direta das coordenadas tridimensionais de pontos de uma estrutura reconstruída pelas abordagens deste trabalho e a do aplicativo *Tracking Tools* não é possível. Isso acontece porque a estrutura resultante do *OpenMoCap* não contém um fator de escala real e o sistema de coordenadas em que os pontos estão descritos é o da câmera. Enquanto na solução comercial, o fator de escala real existe e o sistema

de coordenadas em que os pontos são descritos pode ser configurado pela definição do plano de chão, como citado na Seção 5.1.1.

Apesar dessa dificuldade, este trabalho propõe uma maneira para avaliar a qualidade da estrutura obtida. Um ponto tridimensional no espaço pode ser determinado unicamente por meio da interseção de quatro esferas, por trilateração (Doukhnitch et al., 2008). Logo, a ideia proposta neste trabalho para avaliar as estruturas tridimensionais obtidas pelo *OpenMoCap* é a comparação de no mínimo quatro distâncias euclidianas para cada ponto presente na estrutura capturada com as distâncias calculadas pelo aplicativo *Tracking Tools*.

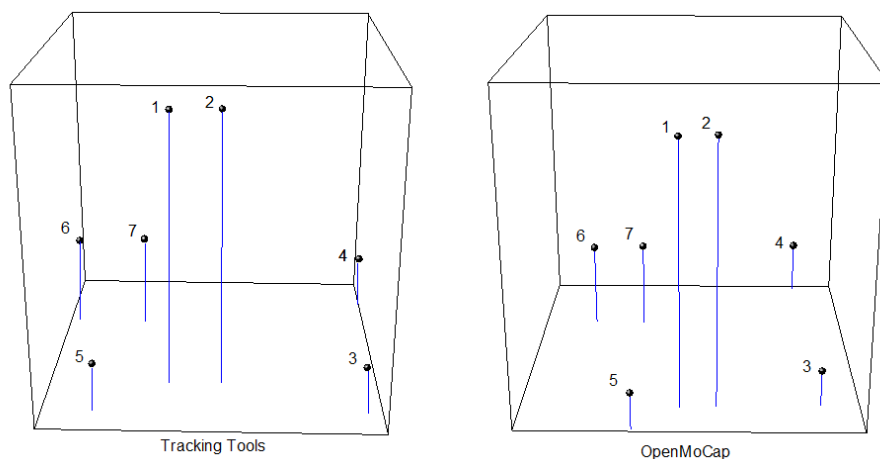
A Figura 5.3 exibe fotografias de vários ângulos da cena escolhida para a comparação de estrutura. Os marcadores são os pontos cinza e definem uma estrutura tridimensional. Existem dois no topo da caixa predominantemente amarela, mais dois no topo da caixa preta e três colocados sobre o suporte preto em forma de "L". No total, a estrutura alvo de captura é formada por 7 pontos de interesse.



**Figura 5.3.** Cena Capturada com Estrutura de Marcadores.

A cena foi mantida estática e foi capturada pelas duas aplicações mantendo as câmeras na mesma posição com tripés. Uma diferença entre as condições iniciais das

duas abordagens foi o uso de uma terceira câmera pelo aplicativo *Tracking Tools*, número mínimo necessário para seu correto funcionamento. Outra foi a determinação de parâmetros das câmeras: no *OpenMoCap* foram usadas apenas as 7 correspondências de POIs na cena, enquanto no *software* da *NaturalPoint* o processo inteiro de calibração foi feito. A Figura 5.4 compara qualitativamente as estruturas obtidas pelos dois programas.



**Figura 5.4.** Estrutura Tridimensional Gerada pelo *Tracking Tools* e *OpenMoCap*.

As estruturas recuperadas pelas duas abordagens se parecem à real, mas como as coordenadas do *Tracking Tools* possuem escala, elas puderam ser verificadas. Portanto, as distâncias entre os pontos obtidos pela solução comercial são consideradas o *ground truth* para a comparação. As Tabelas 5.6 e 5.7 foram construídas e exibem as distâncias euclidianas entre os pontos da estrutura no *OpenMoCap* e no *Tracking Tools*.

**Tabela 5.6.** Distâncias entre Pontos da Estrutura Obtida pelo *Tracking Tools* em metros.

	1	2	3	4	5	6	7
1		0.075	0.561	0.590	0.498	0.448	0.421
2			0.530	0.557	0.521	0.482	0.440
3				0.313	0.378	0.518	0.435
4					0.510	0.474	0.370
5						0.281	0.283
6							0.106

**Tabela 5.7.** Distâncias entre Pontos da Estrutura Obtida pelo *OpenMoCap*.

	1	2	3	4	5	6	7
1		0,380	2,876	4,162	2,571	3,064	2,883
2			2,708	4,028	2,677	3,192	2,947
3				3,142	1,986	3,297	2,875
4					4,069	2,668	2,176
5						2,685	2,606
6							0,576

Considerando que as estruturas são semelhantes e princípios já citados da trilatação, deve existir um fator multiplicativo escalar que aproxima as distâncias das duas abordagens. A Tabela 5.8 mostra a razão entre as distâncias encontradas do *OpenMoCap* e do *Tracking Tools*. Os valores foram normalizados entre 0 e 1 por meio da divisão de todos elementos pela razão máxima.

**Tabela 5.8.** Fator Multiplicativo entre Distâncias *OpenMoCap* e *Tracking Tools*.

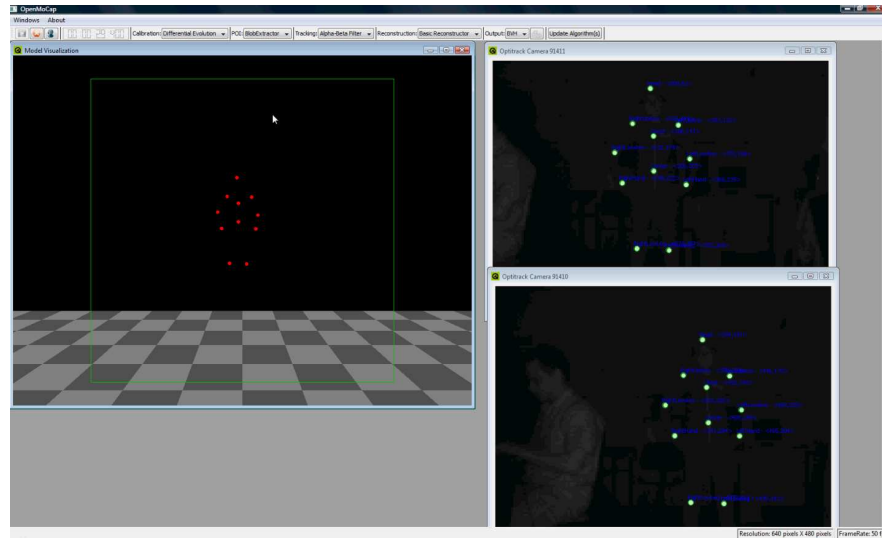
	1	2	3	4	5	6	7
1		0,504	0,510	0,702	0,514	0,681	0,682
2			0,508	0,719	0,512	0,659	0,668
3				1,000	0,524	0,634	0,658
4					0,794	0,561	0,585
5						0,951	0,916
6							0,540

Quanto menor a dispersão desse fator multiplicativo, melhor é a qualidade da estrutura reconstruída pelo aplicativo tema deste trabalho. A mediana calculada para os fatores desse exemplo é 0,658. Distâncias que possuem um fator multiplicativo muito diferente desse valor tem erro maior. Particularmente, os pontos 3, 4 e 5 são aqueles que possuem mais erro.

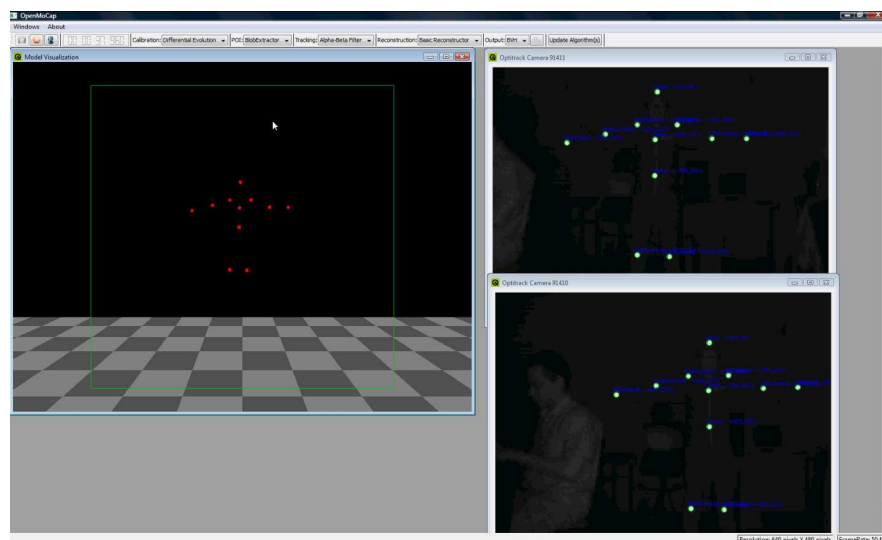
Considerando apenas o primeiro e o terceiro quartis dessa amostra, 0,524 e 0,702 respectivamente, a diferença máxima da media é 0,134. Logo, o erro máximo nessa parcela de dados é de aproximadamente 20%. Tendo em vista que o *OpenMoCap* tem uma câmera a menos e somente 7 pontos foram usados para estimar os parâmetros das câmeras, 20% é um erro máximo razoável. Portanto, aplicações simples que não requerem altíssima precisão, como animação de personagens e controle de jogos, podem ser beneficiadas.

### 5.3.5 Saída e Movimento 3D

Mesmo com todas as limitações deste trabalho, principalmente aquelas descritas na Seção 5.3.4, foi possível a captura em tempo real de movimentos simples de uma pessoa com marcadores. As Figuras 5.5, 5.6 e 5.7 exibem três quadros chaves de um vídeo da tela do *OpenMoCap* capturando movimento a 50 quadros por segundo. O vídeo completo e outras sequências produzidos podem ser encontrados em Flam (2009).

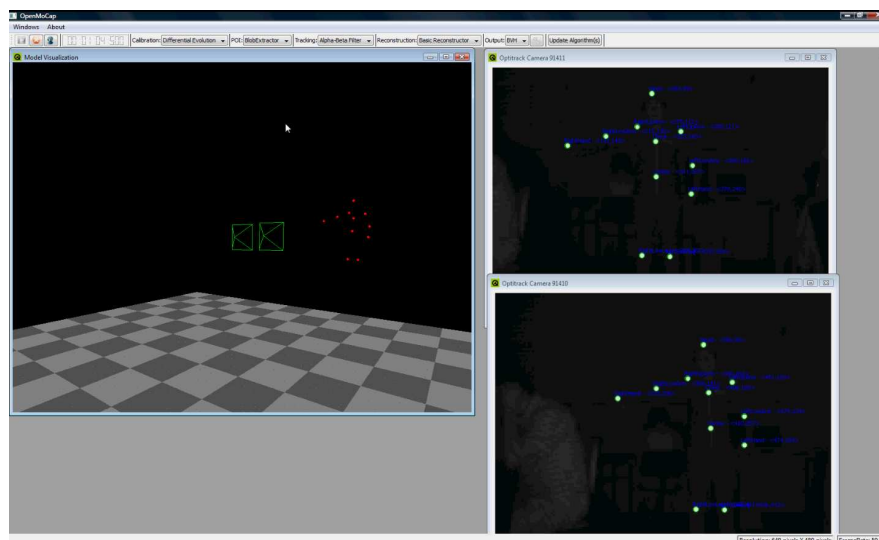


**Figura 5.5.** Primeiro Quadro Exemplo de Captura de Movimento pelo *OpenMoCap*.



**Figura 5.6.** Segundo Quadro Exemplo de Captura de Movimento pelo *OpenMoCap*.





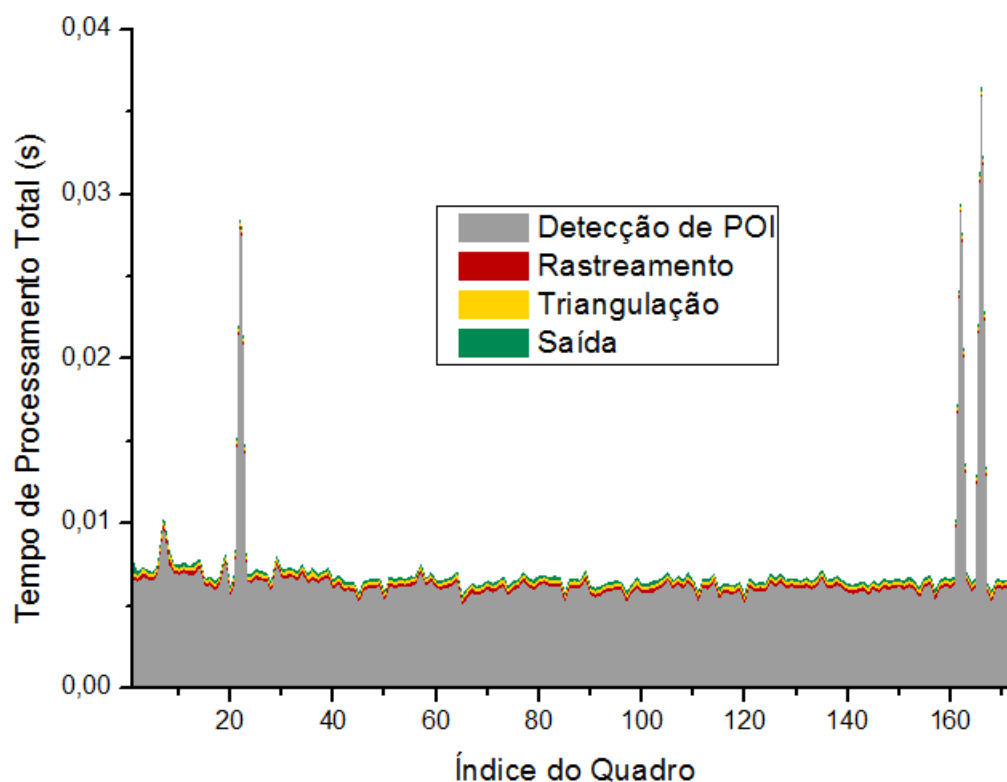
**Figura 5.7.** Terceiro Quadro Exemplo de Captura de Movimento pelo *OpenMoCap*.

### 5.3.6 Tempo de Processamento

Uma questão que deve ser respondida neste trabalho é se a arquitetura da aplicação proposta é capaz de capturar movimento em tempo real. Em outras palavras, qual é a taxa máxima de quadros por segundo que podem ser processados pelo fluxo construído no *OpenMoCap*. A Figura 5.8 é um gráfico área (Boslaugh & Watters, 2008) que mostra o tempo gasto em segundos por cada etapa e o tempo total em um determinado quadro. Para esse teste, os respectivos tempos foram obtidos durante um trecho de 175 quadros, de uma seção de captura, realizada a uma taxa de 50 quadros por segundo.

Uma conclusão imediata desse gráfico área é que quase todo o tempo de processamento do fluxo é gasto pela detecção de POIs. As outras etapas correspondem a parcelas muito pequenas do esforço total necessário para gravar movimento. Alguns testes foram feitos também variando o número de marcadores na cena, mas esse tipo de alteração teve impacto quase nulo no tempo total gasto pelos algoritmos implementados no *OpenMoCap*. Isso evidencia o foco dos fabricantes comerciais de sistemas ópticos passivos de captura de movimento em dispositivos que consigam obter por *hardware* os centróides dos POIs.

A outra observação que pode ser feita sobre a Figura 5.8, e que efetivamente determina a velocidade em que se pode gravar movimento em tempo real, é a respeito dos picos de processamento. O trecho de quadros exibido nesse gráfico foi especialmente selecionado por mostrar o maior pico de processamento detectado em várias seções de captura, aproximadamente 37 milissegundos. Esses eventos ocorrem basicamente

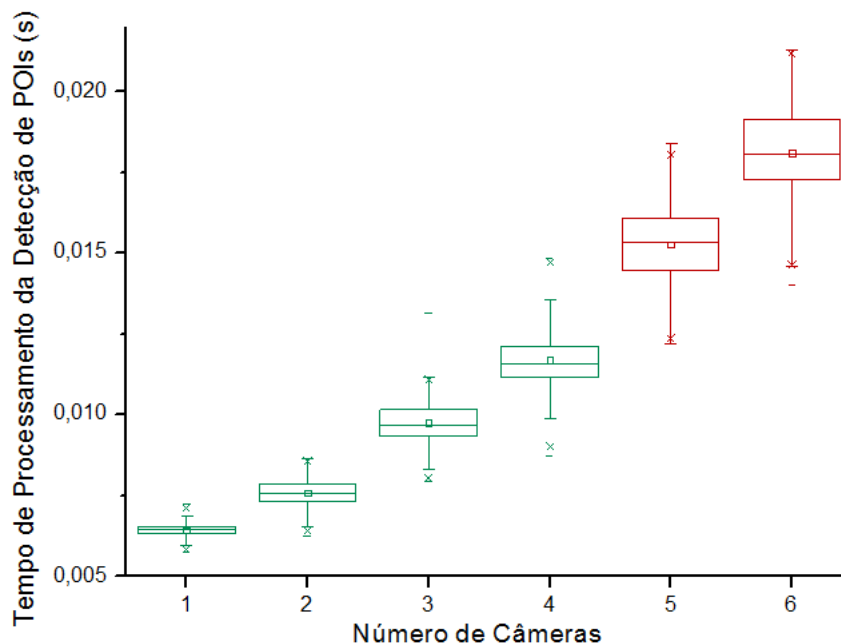


**Figura 5.8.** Gráfico Área do Tempo de Processamento Total do *OpenMoCap*.

pelo compartilhamento de recursos disponíveis de *hardware* com outras aplicações. O escalonador do sistema operacional não consegue voltar ao fluxo de execução do *OpenMoCap* no tempo necessário caso uma operação de entrada/saída muito intensa esteja sendo realizada, mesmo em um curto espaço de tempo.

Dependendo da aplicação desejada, pode ser que seja aceitável a perda de um quadro no momento de captura. É possível depois ainda realizar uma interpolação para preencher a informação perdida. Se essa situação for considerada para o uso do *OpenMoCap*, então é possível capturar movimento a uma taxa de 50 quadros por segundo, ou seja, um limite de 20 milissegundos por quadro. Caso contrário, como não foi registrado nenhum pico maior do que 37 milissegundos na máquina utilizada neste experimento, uma taxa de 25 quadros por segundo seria alcançada.

Uma contribuição da aplicação desenvolvida neste trabalho que também precisa ser validada é a arquitetura de múltiplos fluxos e o suporte a várias câmeras. Para isso, uma quantidade fixa de 20 marcadores foi colocada em uma cena observada por um número variável de câmeras conectadas ao *OpenMoCap*. A Figura 5.9 é um gráfico caixa (Boslaugh & Watters, 2008) do número de câmeras utilizadas e o respectivo tempo gasto na detecção de POIs, .



**Figura 5.9.** Gráfico Caixa do Tempo de Processamento de Detecção de POIs pelo Número de Câmeras *OpenMoCap*.

As caixas no gráfico representam os intervalos existentes entre o primeiro e o terceiro quartil das amostras, isto é, o local onde se encaixam 50% das medidas obtidas. O aumento no número de câmeras e, conseqüentemente, de *threads*, mostra também um aumento na largura das caixas, ou seja, uma maior dispersão. Esse fenômeno é esperado devido à questão de compartilhamento de recursos da máquina pelos fluxos de execução. O resultado é que, aumentando o número de câmeras, a confiabilidade do sistema cai e taxas cada vez menores de quadros por segundo podem ser processadas em tempo real.

Finalmente, outra informação interessante presente no gráfico da Figura 5.9 está relacionada às médias dos tempos gastos em processamento, simbolizados pelos pequenos quadrados dentro das caixas. Até a quarta câmera, o aumento na média com a soma de um fluxo de execução é cerca de 2 milissegundos, mas a partir daí, soma-se quase que o tempo todo gasto com uma câmera única. Portanto, a vantagem da arquitetura de múltiplos fluxos desenvolvida é evidenciada, já que a máquina utilizada nos testes tem quatro núcleos de processamento e os fluxos de execução das quatro câmeras são intensos.

## 5.4 Considerações

Neste capítulo, foram apresentados experimentos que validaram o fluxo e a arquitetura de captura de movimento construída neste trabalho. Apesar de existirem diversas limitações no *OpenMoCap*, o objetivo principal de oferecer uma solução de código livre extensível que possua a cadeia completa de processamento para a gravação de movimento foi atingido. No próximo capítulo, conclusões gerais sobre o projeto são feitas e várias sugestões para superar as limitações encontradas são dadas, como trabalhos futuros.

# Capítulo 6

## Conclusões

### 6.1 Objetivos Alcançados

Tarefas que incluem processamento de imagens em cenas reais, análise de vídeo em tempo limitado e modelagem matemática são usualmente complexas. Elas envolvem um tipo de *expertise* multidisciplinar e resultam no desenvolvimento de ferramentas poderosas, muito úteis nas suas diversas áreas de aplicação, como o cinema, a televisão, as novas mídias e os jogos digitais. A captura óptica de movimento, o tema central deste trabalho, é um exemplo de tarefa desse tipo. Mesmo diante dessa complexidade, diversas soluções comerciais já foram desenvolvidas com muito sucesso, tendo em vista a grande aplicabilidade da técnica. No Brasil, não há nenhum equipamento de captura de movimento em tempo real desenvolvido e poucos sistemas são encontrados em qualquer de suas aplicações. O custo desses equipamentos é muitas vezes proibitivo, os sistemas são fechados e não existe pessoal capacitado para o seu uso e para manutenção.

Este trabalho focou na construção de um sistema de código livre para a captura óptica de movimento. Ele está inserido em um projeto em desenvolvimento no NPDI, com apoio da FAPEMIG e do CNPq, de construção de um sistema de captura de movimento robusto, que seja capaz de atender às demandas de geração de bancos de dados de movimento para o audiovisual e os jogos digitais. Para isso, foi criada uma metodologia para a realização da tarefa, baseada em fundamentos teóricos do processamento digital de imagens, da visão computacional, da computação gráfica e da programação. Especificamente, foram usados conceitos de modelo de câmera, reconstrução, evolução diferencial, limiarização, componentes conectados e estimadores.

A aplicação desenvolvida, o *OpenMoCap*, é autônoma, no sentido que inclui todos os componentes necessários para completar a cadeia de processamento existente na gravação de movimento. Sua arquitetura é flexível e extensível, ou seja, permite a

substituição e adição de módulos específicos aproveitando todo o fluxo já implementado. Além disso, ela toma vantagem de múltiplos fluxos de execução e é apropriada para tempo real. Por fim, possui uma interface gráfica simples e funcional.

Os resultados experimentais comprovam os objetivos alcançados por este trabalho e, apesar de não ser tão robusto e preciso quanto uma solução comercial, o aplicativo pode ser útil na geração de animações simples no atual estágio. Melhorias que já estão sendo estudadas e implementadas, baseadas nos experimentos e na construção do código, deverão tornar o sistema mais robusto e preciso. Esse trabalho também alavancou novos esforços de pesquisa e contribuiu no aumento de *expertise* na área pelo grupo de pesquisa, que poderão trazer muitos resultados positivos no futuro próximo.

## 6.2 Trabalhos Futuros

Este trabalho incentiva a pesquisa e o desenvolvimento da técnica de captura óptica de movimento. Diversas ideias de melhoramentos surgiram ao longo de sua construção, mas não puderam ser implementadas devido aos recursos e ao tempo disponíveis no momento. Algumas dessas melhorias são citadas abaixo, como sugestões de continuidade para o projeto.

- Estender as etapas de estimação de parâmetros de câmeras e de triangulação para mais de duas câmeras. Criar um mecanismo para obtenção mais fácil de pontos correspondentes para o processo de otimização.
- Utilizar câmeras genéricas, já suportadas pela aplicação, para implementar algoritmos de detecção de partes do corpo ou construção de malhas tridimensionais para captura de movimento sem marcadores.
- Desenvolver o passo de detecção de POIs usando GPU (do inglês *Graphics Processing Unit*) para diminuir sua intensidade de processamento. Outra opção é utilizar uma arquitetura de rede distribuída com múltiplos processadores baratos, como o *Intel Atom*.
- Obter a estrutura do alvo de captura em uma escala real e definir a origem do sistemas de coordenadas a partir de um plano.
- Fazer a inicialização automática de semântica dos POIs, usando casamento inexato de grafos, principalmente porque a importação de esqueletos no formato BVH já é feita.

- Aprimorar o algoritmo de rastreamento, usando filtro de *Kalman* (Grewal & Andrews, 2008) e adicionar a área do POI como um estado.
- Gerar a saída de dados em um formato hierárquico, de esqueleto, como o BVH.

# Referências Bibliográficas

- Animazoo (2008). Motion Capture Comparison Table. Este é um documento eletrônico disponível em: <http://www.animazoo.com/Comparisontable.aspx>. Acessado em: 2 de Dezembro de 2008. ix, 10, 11
- Ascension Technology Corporation (2009). Flock of Birds. Este é um documento eletrônico disponível em: <http://www.ascension-tech.com/>. Acessado em: 4 de Janeiro de 2009. ix, 9
- Autodesk (2009). Autodesk - 2d and 3d design and engineering software for architecture, manufacturing, and digital entertainment. Este é um documento eletrônico disponível em: <http://www.autodesk.com>. Acessado em: 10 de Maio de 2009. 48
- Boslaugh, S. & Watters, D. P. A. (2008). *Statistics in a nutshell*. O'Reilly & Associates, Inc., Sebastopol, CA, USA. 59, 67, 68
- Bradski, D. G. R. & Kaehler, A. (2008). *Learning OpenCV*. O'Reilly Media, Inc., Sebastopol, CA, USA. 43
- Castro, J.; Medina-Carnicer, R. & Galisteo, A. M. (2006). Design and evaluation of a new three-dimensional motion capture system based on video. *Gait and Posture*, 24(1):126 – 129. <http://dx.doi.org/10.1016/j.gaitpost.2005.08.001>. 19, 20, 21
- Cyganek, B.; Siebert, J. (2009). *An Introduction to 3D Computer Vision Techniques and Algorithms*. John Wiley & Sons, Ltd, Chichester, UK. 25
- da Silva, F. W. S. V. (1998). Um Sistema de Animação Baseado em Movimento Capturado. Dissertação de Mestrado, COPPE/UFRJ. 6, 12
- de la Fraga, L. & Vite Silva, I. (2008). Direct 3d metric reconstruction from two views using differential evolution. *IEEE Congress on Evolutionary Computation, 2008 (IEEE World Congress on Computational Intelligence)*., pp. 3266–3273. <http://dx.doi.org/10.1109/CEC.2008.4631240>. 42



- Doukhnitch, E.; Salamah, M. & Ozen, E. (2008). An efficient approach for trilateration in 3d positioning. *Computer Communications*, 31(17):4124–4129. <http://dx.doi.org/10.1016/j.comcom.2008.08.019>. 63
- Figueroa, P. J.; Leite, N. J. & Barros, R. M. L. (2003). A flexible software for tracking of markers used in human motion analysis. *Computer Methods and Programs in Biomedicine*, 72(2):155 – 165. [http://dx.doi.org/10.1016/S0169-2607\(02\)00122-0](http://dx.doi.org/10.1016/S0169-2607(02)00122-0). 18, 19, 20
- Flam, D. L. (2009). Openmocap sample videos. Este é um documento eletrônico disponível em: <http://www.npdi.dcc.ufmg.br/mocap/>. Acessado em: 26 de Junho de 2009. 66
- Forsyth, D. A. & Ponce, J. (2002). *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference. 26
- Furniss, M. (1999). Motion Capture. Este é um documento eletrônico disponível em: <http://web.mit.edu/comm-forum/papers/furniss.html>. Acessado em: 15 de Novembro de 2008. 7
- Gavrila, D. M. (1999). The visual analysis of human movement: a survey. *Computer Vision and Image Understanding*, 73(1):82–98. <http://dx.doi.org/10.1006/cviu.1998.0716>. 17
- Gomide, J. V. B. (2006). Captura Digital de Movimento no Cinema de Animação. Dissertação de Mestrado, EBA/UFMG. 7
- Gonzalez, R. C. & Woods, R. E. (2006). *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA. 30, 40
- Grewal, M. S. & Andrews, A. P. (2008). *Kalman Filtering : Theory and Practice Using MATLAB (3rd Edition)*. Wiley-Interscience. 19, 73
- Hartley, R. & Zisserman, A. (2003). *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA. 19, 22
- Hartley, R. I. & Sturm, P. (1997). Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157. <http://dx.doi.org/10.1006/cviu.1997.0547>. 8, 27
- Hexamite (2009). Ultrasonic Industrial Positioning Systems, and ranging. Este é um documento eletrônico disponível em: <http://www.ascension-tech.com/>. Acessado em: 4 de Janeiro de 2009. ix, 12

- Inition (2008). Motion Capture / Tracking from Inition. Este é um documento eletrônico disponível em: [http://www.inition.co.uk/inition/products.php?CatID\\_=11](http://www.inition.co.uk/inition/products.php?CatID_=11). Acessado em: 26 de Novembro de 2008. 8, 9, 10, 11, 12
- Intel Corporation (2009). Laptop, notebook, desktop, server and embedded processor technology - intel. Este é um documento eletrônico disponível em: <http://www.intel.com>. Acessado em: 26 de Maio de 2009. 52
- Kernighan, B. W. & Pike, R. (1999). *The practice of programming*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. 49
- Kitagawa, M. & Windsor, B. (2008). *MoCap for Artists: Workflow and Techniques for Motion Capture*. Focal Press, Burlington, MA, USA. 14, 16, 35
- Liverman, M. (2004). *The Animator's Motion Capture Guide: Organizing, Managing and Editing*. Charles River Media, Burlington, MA, USA. 7
- Mahajan, V. N. (1998). *Optical Imaging and Aberrations - Part I: Ray Geometrical Optics*. SPIE Press, Bellingham, WA, USA. 5
- McConnell, S. (2004). *Code Complete, Second Edition*. Microsoft Press, Redmond, WA, USA. 49
- Menache, A. (2000). *Understanding Motion Capture for Computer Animation and Video Games*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 1, 7, 14, 16, 36, 41
- Microsoft Corporation (2009). Microsoft corporation. Este é um documento eletrônico disponível em: <http://www.microsoft.com>. Acessado em: 26 de Maio de 2009. 52
- Moeslund, T. B. & Granum, E. (2001). A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231--268. <http://dx.doi.org/10.1006/cviu.2000.0897>. 1, 2, 17
- Moeslund, T. B.; Hilton, A. & Krüger, V. (2006). A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2):90--126. <http://dx.doi.org/10.1016/j.cviu.2006.08.002>. 1, 17
- Motion Analysis Corporation (2009). The industry leader for 3d passive optical motion capture. Este é um documento eletrônico disponível em: <http://www.motionanalysis.com/>. Acessado em: 5 de Fevereiro de 2009. 47

- Mova LLC (2008). Mova : Home. Este é um documento eletrônico disponível em: <http://www.mova.com/>. Acessado em: 20 de Novembro de 2008. 7
- Mulder, A. (1994). Human Movement Tracking Technology. Relatório Técnico 94-1, School of Kinesiology, Simon Fraser University. 12
- OptiTrack (2008). Optical motion capture and tracking :: Optitrack. Este é um documento eletrônico disponível em: <http://www.naturalpoint.com/optitrack/>. Acessado em: 2 de Dezembro de 2008. 18, 39
- PhaseSpace Inc. (2008). PhaseSpace Inc | Optical Motion Capture . Este é um documento eletrônico disponível em: <http://www.phasespace.com>. Acessado em: 13 de Novembro de 2008. ix, 6, 8, 18
- Press, W. H.; Teukolsky, S. A.; Vetterling, W. T. & Flannery, B. P. (1992). *Numerical recipes in C (2nd ed.): the art of scientific computing*. Cambridge University Press, New York, NY, USA. 28
- Raskar, R.; Nii, H.; deDecker, B.; Hashimoto, Y.; Summet, J.; Moore, D.; Zhao, Y.; Westhues, J.; Dietz, P.; Barnwell, J.; Nayar, S.; Inami, M.; Bekaert, P.; Noland, M.; Branzoi, V. & Bruns, E. (2007). Prakash: lighting aware motion capture using photosensing markers and multiplexed illuminators. *ACM Transactions on Graphics*, 26(3):36. <http://doi.acm.org/10.1145/1276377.1276422>. 6, 20
- RPM Produtora (2008). RPM Produtora. Este é um documento eletrônico disponível em: <http://www.rpm.com.br/web/home.asp>. Acessado em: 13 de Novembro de 2008. 2
- Stage (2009). Organic motion: Solutions. Este é um documento eletrônico disponível em: <http://www.organicmotion.com/solutions/stage>. Acessado em: 18 de Fevereiro de 2009. 18
- Steger, C.; Ulrich, M. & Wiedemann, C. (2008). *Machine Vision Algorithms and Applications*. Wiley-VCH, Weinheim, BW, DE. 4, 32
- Storn, R. (2009). Differential Evolution Homepage. Este é um documento eletrônico disponível em: <http://www.icsi.berkeley.edu/storn/code.html>. Acessado em: 1 de Maio de 2009. 29
- Storn, R. & Price, K. (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341--359. <http://dx.doi.org/10.1023/A:1008202821328>. 29

- Sturman, D. J. (1994). A Brief History of Motion Capture for Computer Character Animation. In *SIGGRAPH 94, Character Motion Systems, Course notes*. 1
- Sundaresan, A. & Chellappa, R. (2005). Markerless motion capture using multiple cameras. In *CVIIE '05: Proceedings of the Computer Vision for Interactive and Intelligent Environment*, pp. 15--26, Washington, DC, USA. IEEE Computer Society. <http://dx.doi.org/10.1109/CVIIE.2005.13>. 19
- Tanie, H.; Yamane, K. & Nakamura, Y. (2005). High marker density motion capture by retroreflective mesh suit. In *IEEE International Conference on Robotics and Automation*, pp. 2884–2889. 19
- Trolltech (2009). Qt Software - Code Less, Create More, Deploy Everywhere. Este é um documento eletrônico disponível em: <http://trolltech.com>. Acessado em: 4 de Janeiro de 2009. 52
- Trucco, E. & Verri, A. (1998). *Introductory Techniques for 3-D Computer Vision*. Prentice Hall PTR, Upper Saddle River, NJ, USA. 25
- Uchinoumi, M.; Tan, J. K. & Ishikawa, S. (2004). A simple-structured real-time motion capture system employing silhouette images. In *Proceedings of the IEEE International Conference on Systems, Man & Cybernetics*, pp. 3094–3098, The Hague, Netherlands. <http://dx.doi.org/10.1109/ICSMC.2004.1400814>. 19
- Umbaugh, S. E. (2005). *Computer Imaging: Digital Image Analysis and Processing*. CRC Press, Boca Raton, FL, USA. ix, 4, 32, 33, 34
- Vicon Motion Systems (2008). Motion Capture Systems from Vicon. Este é um documento eletrônico disponível em: <http://www.vicon.com>. Acessado em: 13 de Novembro de 2008. 6, 17
- Watson, T. (2009). videoInput Library. Este é um documento eletrônico disponível em: <http://muonics.net/school/spring05/videoInput/>. Acessado em: 4 de Janeiro de 2009. 52
- Willow Garage (2009). OpenCV - Wiki. Este é um documento eletrônico disponível em: <http://pr.willowgarage.com/wiki/OpenCV>. Acessado em: 4 de Janeiro de 2009. 51
- Yoo, J.-C. & Kim, Y.-S. (2003). Alpha-beta-tracking index ( $[\alpha]$ - $[\beta]$ - $[\lambda]$ ) tracking filter. *Signal Processing*, 83(1):169 – 180. [http://dx.doi.org/10.1016/S0165-1684\(02\)00388-2](http://dx.doi.org/10.1016/S0165-1684(02)00388-2). 34