AVALIAÇÃO DE MÉTODOS DE RASTREAMENTO DE MARCADORES PARA UM SISTEMA ÓPTICO DE CAPTURA DE MOVIMENTO

DANIEL PACHECO QUEIROZ

AVALIAÇÃO DE MÉTODOS DE RASTREAMENTO DE MARCADORES PARA UM SISTEMA ÓPTICO DE CAPTURA 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

Janeiro de 2011

Queiroz, Daniel Pacheco
Q3a Avaliação de métodos de rastreamento de marcadores para um sistema óptico de captura de movimento / Daniel Pacheco Queiroz. — Belo Horizonte, 2011
xxvi, 86 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais
Orientador: Arnaldo de Albuquerque Araújo

Co-orientador: João Victor Boechat Gomide.

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

CDU 519.6*82

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

FOLHA DE APROVAÇÃO

Avaliação de métodos de rastreamento de marcadores passivos para um sistema óptico de captura de movimento

DANIEL PACHECO DE QUEIROZ

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. ALEXEI MANSO CORREA MACHADO
Departamento de Ciência da Computação - PUC - MG

PROF. LUIZ CHAIMOWICZ
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 18 de fevereiro de 2011.

Dedico este trabalho à ciência, que ela o aproveite e faça bom uso.

Agradecimentos

A Deus por criar e me aproximar de tudo que será agradecido a seguir.

Ao meu orientador Arnaldo de Albuquerque Araújo e ao meu co-orientador João Victor Boechat Gomide, sem os quais nada disso seria possível.

Aos professores Alexei e Luiz Chaimowicz que integraram minha banca avaliadora, juntamente com meus orientadores, e foram extremamente atenciosos ao convite de participação.

A minha família - Atanásio, Sueli e Renata - por seu amor e carinho e por terem me proporcionado uma educação que me trouxe até aqui.

A Paula que esteve sempre ao meu lado em todos os momentos, com todo seu amor.

Ao David Flam pelo qualidade do seu trabalho, o qual foi imprescindível para a realização deste.

A todo o pessoal do Verlab pela prestatividade e vontade de ajudar, em especial: ao Professor Luiz Chaimowicz, Professor Mario, Vilar, Marcelo Borghetti, Douglas e Erickson.

A todos os colegas do NPDI pelo companheirismo, momentos alegres e discussões construtivas.

A todos os professores e colegas do DCC, desde a graduação, que fazem com que o departamento seja um dos melhores do país e forme profissionais de alto nível.

Aos funcionários do DCC que sempre estavam dispostos a resolver minhas pendências na graduação e no mestrado, em especial: Túlia, Maristela, Sheila, Renata, Flávia, Cláudia, Sônia e Alexandre.

A CAPES e ao programa REUNI por minha bolsa de mestrado e aos órgãos CNPq e FAPEMIG por contribuírem com o projeto como um todo.

E por fim, agradeço a todos que de alguma forma, direta ou indiretamente, contribuíram para que esse trabalho fosse realizado: obrigado!

Resumo

São muitas as aplicações para um sistema de captura de movimento. Sistemas que registrem o movimento humano podem ser empregados em diversas áreas, como medicina, esportes, segurança e entretenimento. Embora a utilidade e os benefícios proporcionados por esses sistemas seja grande, seu uso, principalmente no Brasil, é pouco difundido. Este trabalho se integra a um esforço maior para a criação do *OpenMoCap*, um *software* brasileiro e de código aberto para a captura de movimento, utilizando câmeras e marcadores passivos. Neste trabalho, três métodos de rastreamento são empregados na tarefa de perseguir marcadores passivos para a captura óptica de movimento. Os métodos avaliados e comparados são: o filtro de Kalman, o filtro Alfa-Beta e o algoritmo CONDENSATION. Para avaliar o desempenho dos métodos foram empregados testes de robustez da perseguição e medição do custo computacional. Considerando o desempenho geral, o filtro de Kalman foi escolhido para ser empregado no sistema de captura.

Palavras-chave: Captura de movimento, rastreamento de marcadores, filtro de Kalman, filtro Alfa-Beta, algoritmo CONDENSATION.

Abstract

There are many applications for a motion capture system. Systems that record human motion can be employed in several fields, such as medicine, sports, security and entertainment. Nevertheless the usefulness and benefits provided by these systems are many, its use is restrictedly disseminated, mainly in Brazil. This work is part of a larger effort to create the *OpenMoCap*, a Brazilian and open source software for motion capture, which uses cameras and passive markers. In this work, three tracking methods are used in the task of tracking passive markers for optical motion capture. The methods evaluated and compared are: the Kalman filter, the Alpha-Beta filter and the CONDENSATION algorithm. To evaluate the performance of the methods the robustness of the tracking was tested and the computational cost measured. Considering the overall performance, the Kalman filter was chosen to be employed in the capture system.

Keywords: Motion capture, marker tracking, Kalman filter, Alpha-Beta filter, CONDENSATION algorithm.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

3D 3 dimensões

BraMBLe *A Bayesian Multiple-Blob Tracker*

CAPES Coordenação de Aperfeiçoamento de Pessoal de Nível Superior

CNPq Conselho Nacional de Desenvolvimento Científico e Tecnológico

CONDENSATION ***CON****ditional* ***DENS****ity Propag****ATION*** *for Visual Tracking*

DCC Departamento de Ciência da Computação

FAPEMIG Fundação de Amparo à Pesquisa do Estado de Minas Gerais

fdp função densidade de probabilidade

LED *Light Emitting Diode*

MoCap *Motion Capture*

NPDI Núcleo de Processamento Digital de Imagens

POI *Point of Interest*

SVD *Singular Value Decomposition*

UFMG Universidade Federal de Minas Gerais

Sumário

Capítulo 1

Introdução

Este trabalho está inserido no projeto de desenvolvimento de uma aplicação de código aberto para a captura de movimento. O principal objetivo é tornar o sistema mais robusto e eficaz. Para isso, diversas medidas serão descritas neste documento. Dentre elas, a mais importante é a implementação e avaliação dos principais métodos empregados no rastreamento de sistema de captura de movimento. De forma que, com os resultados obtidos, possa-se escolher o melhor método de rastreamento para ser empregado no sistema.

1.1 Contextualização

Registrar o movimento humano é um desejo antigo de vários estudiosos. Desde o final do século XIX, há casos como o do fotógrafo Eadweard J. Muybridge, que para satisfazer a curiosidade de um magnata americano (fundador da Universidade de Stanford), fotografou o galopar de um cavalo. Com as fotos ficou provado que o cavalo deixa de tocar o solo com as quatro patas por alguns momentos, como mostra a Figura 1.1.

Naquele mesmo período, houve outros trabalhos relacionados com a captura de movimentos utilizando fotografias. Entre eles, Etienne-Jules Marey, inventor da câmera cronofotográfica de placa fixa em 1880, aparelho utilizado para registrar diversas fotografias ao longo do tempo em uma mesma placa fotográfica ou filme. Este tipo de aplicação da captura de movimento foi mais tarde classificado como *Análise* [Moeslund, 2000; Moeslund & Granum, 2001], na qual o objetivo é observar um movimento para analisá-lo posteriormente e compreendê-lo.

Desde o século XIX, diversos outros trabalhos tentaram registrar os movimentos em apenas duas dimensões [Menache, 2000], com fotografias e filmes, mas só com o uso de computadores foi possível capturar e reconstruir os movimentos em 3 dimensões

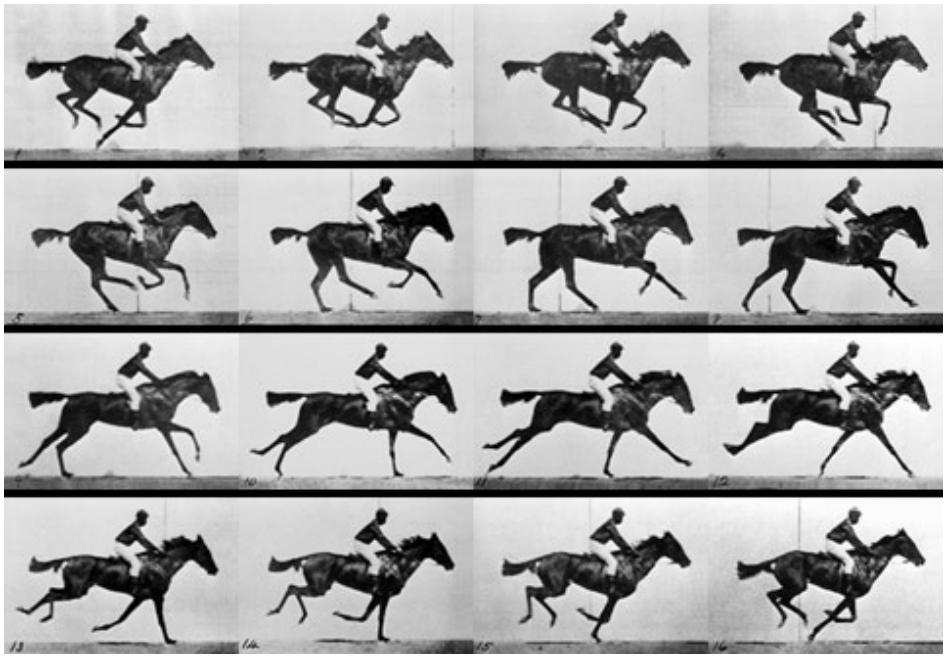

Figura 1.1. Experimento de Eadward Muybridge de 1878, para saber se o cavalo não toca o solo em algum momento do seu galopar.

(3D). Os primeiros trabalhos nessa área foram feitos seguindo metodologias próprias e experimentais, geralmente com o intuito de realizar a captura para um fim específico, como foi feito no comercial *"Brilliance"*, exibido em 1985 durante o intervalo do Super Bowl, jogo final da liga de futebol americano nos Estados Unidos [Gomide et al., 2010].

Esta última aplicação se enquadra na classificação de *Controle*, na qual o objetivo é utilizar a sequência de movimentos capturada para controlar certas estruturas, desde peças virtuais 3D a esqueletos utilizados para a animação de personagens virtuais. A captura de movimento ainda é empregada em uma terceira área de aplicação, chamada de *Vigilância*, na qual o objetivo é rastrear objetos ou pessoas com o intuito de prever a ocorrência de crimes como, por exemplo, roubo de carros, em que um sistema de vigilância pode observar um estacionamento na tentativa de detectar estes crimes.

Além da classificação da captura de movimento quanto à sua aplicação, de acordo com [Gomide, 2006] pode-se, também, classificá-los quanto ao princípio físico empregado no sistema, obtendo-se três classes: *sistemas mecânicos*, *sistemas magnéticos* e *sistemas ópticos*. Dentre os sistemas mecânicos tem-se diversas abordagens, entre elas sistemas como o *acústico*, *protético* e o *inercial*. Todos estes sistema utilizam **marcadores**, que são objetos no mundo físico que determinam os pontos a serem gerados no mundo virtual. Quatro das tecnologias mais comumente utilizadas são descritas a seguir [Flam et al., 2009; da Silva, 1997]:

Sistemas inerciais (mecânicos): Nesses sistemas os marcadores contêm giroscópios e acelerômetros que são colocados nos pontos de interesse da captura, como as juntas de um ator. Com isso é possível obter informação de rotação e posição dos pontos de interesse.

Vantagens: Não sofrem problemas de oclusão, típicos de sistemas ópticos; nem interferência por objetos metálicos, que ocorre em sistemas magnéticos.

Desvantagens: Necessitam de baterias nos marcadores; alto preço.

Figura 1.2. Sistema inercial — *IGS 190* [Animazoo, 2010].

Sistemas protéticos (mecânicos): São compostos de potenciômetros que, posicionados nas articulações de interesse, são capazes de fornecer suas posições e orientações. São equipamentos de medida absoluta, não fornecem a localização relativa dos objetos.

Vantagens: Altas taxas de captura (>120Hz); não são afetados por nada no ambiente; requerem pouquíssima calibração.

Desvantagens: Extremamente obstrutivos (dificuldade de movimentação com o aparelho); dificuldade de posicionamento do equipamento para gerar realismo.

Sistemas magnéticos: Neste tipo de sistema, os receptores são posicionados nos pontos de interesse, como as articulações do ator. Tais receptores medem a posição 3D e orientação das articulações através de uma antena transmissora, que emite um sinal de pulso. Cada receptor fica ligado à antena por um cabo.

Vantagens: Altas taxas de captura (>100Hz); não há oclusão; baratos.

Figura 1.3. Sistema protético — *Gypsy 6* [Animazoo, 2010].

Desvantagens: Obstrução provocada pelos cabos da antena; interferência causada por objetos metálicos.

Sistemas ópticos: Nestes sistemas, os pontos de interesse podem ser definidos com ou sem o uso de marcadores. No caso do uso de marcadores, esses podem ser ativos (emitem luz) ou passivos (apenas refletem a luz). Para a captura da posição dos marcadores são utilizadas câmeras que capturam a luz visível ou outras faixas do espectro eletromagnético, como o infravermelho. Sabendo-se a posição das câmeras, a posição dos marcadores em cada imagem e a semântica de cada um deles (necessárias para a associação de marcadores capturados por câmeras diferentes) é possível realizar a triangulação e obter as coordenadas 3D dos pontos. O sistema empregado neste trabalho utiliza essa tecnologia.

Vantagens: Altas taxas de captura (>400Hz); grande liberdade de movimentação; sem limite de marcadores.

Desvantagens: Muito caros; sofrem problemas de oclusão.

Os sistemas *Motion Capture* (MoCap) ópticos, segundo Moeslund & Granum [2001], seguem uma metodologia dividida em quatro fases: **inicialização**, **rastreamento**, **estimação de pose** e **reconhecimento**. A **inicialização** consiste em realizar as ações necessárias para que o sistema possa fazer uma correta interpretação da cena, essas ações seriam a calibração das câmeras e a definição da semântica dos pontos de interesse, por exemplo. O **rastreamento** consiste na manutenção da semântica atribuída na inicialização e, sendo o foco deste trabalho, será mais detalhado

Figura 1.4. Sistema magnético — *ReActor 2* [Ascension Technology Corporation, 2010].

adiante. Na **estimação de pose**, as coordenadas 3D dos pontos são recuperadas. Por fim, a fase de **reconhecimento** é definida pela aplicação onde está sendo empregada a captura de movimento, pois nessa fase os dados são analisados e utilizados para algum propósito específico [Gomide et al., 2010].

Especificamente, o sistema utilizado neste trabalho é composto por marcadores passivos refletivos à luz infravermelha e por câmeras com iluminação própria, sensíveis a esta faixa do espectro eletromagnético. Dessa forma, algumas peculiaridades deste

Figura 1.5. Sistema óptico - [Motion Analysis Corporation, 2010].

sistema são observadas, como a falta de uma diferença morfológica entre os marcadores e a ausência de cor nas imagens. Estas características impedem o uso de métodos de rastreamento que se baseiam em cor ou forma.

A principal função de um rastreador, isto é, do algoritmo de **rastreamento**, é manter a semântica de um marcador de um quadro de imagem para o outro. Esta semântica consiste em um nome associado ao marcador na fase de **inicialização** e que deve ter essa associação mantida durante todo o processo de captura, pois essa informação é necessária na etapa de **estimação de pose** (para a triangulação). Também na etapa de **reconhecimento**, a semântica é necessária para fornecer a ligação do ponto 3D gerado com o ponto no mundo real, para que se possam utilizar os pontos para algum propósito específico (como animar um esqueleto virtual). Portanto, um rastreador eficaz deve manter essa associação sempre, sendo robusto a casos de oclusão, que consiste na perda momentânea do marcador por uma das câmeras devido à interposição de um objeto entre ela e o marcador; como também a casos de troca da semântica de marcadores, que podem ocorrer por serem todos iguais.

Para solucionar esses problemas, neste trabalho foi avaliado o desempenho dos rastreadores mais utilizados para esse propósito. Para isso, foram implementados cada

um dos rastreadores avaliados e esses foram aplicados em situações propícias à ocorrência dos erros descritos anteriormente. Além de o rastreador ter de ser robusto aos erros, outra característica que foi determinante para a escolha do melhor algoritmo, foi seu desempenho computacional. Para a escolha do rastreador, esse deve ser considerado eficaz, realizando suas atribuições, mas também deve ser eficiente. A eficiência deve permitir que o sistema possa ser executado em tempo real, considerando o custo computacional de todas as outras fases da captura de movimento.

1.2 Motivação

A principal motivação deste trabalho é contribuir para o desenvolvimento de uma aplicação de código aberto para a captura de movimento. Este trabalho pretende complementar o desenvolvimento do sistema *OpenMoCap*, mensurando a qualidade do rastreador já empregado no sistema e avaliando as alternativas mais indicadas para este tipo de rastreamento.

Atualmente, os sistemas de captura de movimento são caros e, também por isso, existem poucos disponíveis no Brasil. Com o desenvolvimento desta tecnologia no Brasil, e a disponibilidade do código para outros desenvolvedores, pretende-se difundir o uso destes sistemas para todos que necessitem da captura de movimento, seja para fins médicos, esportivos ou para o entretenimento.

A disponibilidade do código também é muito importante para a área acadêmica. Pesquisadores de ciência da computação e visão computacional, ou até mesmo pesquisadores de outras áreas que necessitem dos serviços de um software de captura de movimento, podem se beneficiar deste sistema aberto. Estes cientistas poderão consultar o código e implementar suas aplicações relacionadas, ou utilizar o sistema em outros trabalhos acadêmicos.

1.3 Objetivos

O objetivo principal do presente trabalho é melhorar a eficácia dos processos envolvidos na captura óptica de movimento pelo *OpenMoCap*. Pretende-se mensurar as qualidades do rastreador atualmente utilizado e também compará-lo com os mais utilizados neste tipo de sistema, segundo a literatura. Assim, ao final do trabalho busca-se definir quais as melhores escolhas de um rastreador para este sistema, em diferentes situações.

Outro objetivo é aprender sobre o comportamento dos métodos utilizados e descrevê-lo durante a dissertação. Apesar da análise dos métodos de rastreamento

ser feita sobre um sistema específico, espera-se que o aprendizado sobre o comportamento de cada algoritmo seja útil em outras aplicações. Dessa forma, pretende-se que este trabalho possa servir como base de comparação entre os métodos empregados e sirva como uma fonte de análise ao se escolher um método de rastreamento para outros sistemas MoCap ou para sistemas que utilizem rastreamento em geral, desde que relacionados com o problema.

Como objetivo secundário, espera-se criar uma metodologia de testes para sistemas de captura de movimento. Devido à necessidade de se mensurar a qualidade dos rastreadores empregados neste trabalho, será necessário criar uma metodologia de testes que se aplique a estes sistemas. Esta metodologia deve avaliar as principais características necessárias a um bom rastreador de marcadores. Além disso, com a metodologia e o repositório de vídeos de testes criados, é possível que outros pesquisadores possam avaliar seus sistemas. Utilizando os princípios empregados neste trabalho, é possível termos uma referência para a comparação de rastreadores, até mesmo quando empregados em outras situações semelhantes.

Além dos objetivos diretamente relacionados com a proposta deste trabalho, também pretende-se implementar outras modificações no sistema *OpenMoCap*. Estas modificações visam deixar o sistema mais próximo das aplicações comerciais e incentivar seu uso acadêmico. Dentre as mais importantes, destacam-se: implementação da captura de movimento com múltiplas câmeras e inserção de câmeras simuladas por vídeo.

Quando foi projetado, o sistema *OpenMoCap* operava apenas com duas câmeras. Durante o período de realização dos trabalhos, optou-se por modificar o sistema para que operasse com múltiplas câmeras. Para isso, são necessárias modificações em diversos módulos do sistema, como no módulo responsável pela calibração das câmeras e pela triangulação dos pontos. Concluída esta modificação, o *OpenMoCap* passa a fazer capturas com um número maior de câmeras, como os sistemas comerciais, o que fornece uma captura de melhor qualidade.

A outra modificação do sistema, implementação de uma câmera simulada por vídeo, trás um importante diferencial para o sistema. Como para a avaliação dos métodos, é necessário a replicação das situações propensas a falhas, foi preciso realizar uma modificação no sistema para que ele suporte simular uma câmera através de vídeos previamente gravados. Isto possibilita que o sistema possa fazer uma captura de movimento sem câmeras, desde que se tenha vídeos que as simulem. Esta funcionalidade não está disponível nem mesmo nos sistemas comerciais e é muito importante para pesquisas na área, que necessitam de replicação dos resultados.

1.4 Estrutura da Dissertação

O Capítulo 1 apresenta uma revisão geral sobre a captura de movimento e os métodos empregados para gerar dados de movimento. O fluxo de trabalho do sistema óptico foi mais detalhado, já que é o método utilizado neste trabalho. Além disso, apresenta-se os objetivos deste trabalho e sua motivação.

No Capítulo 2, são apresentados trabalhos que descrevem métodos de rastreamento utilizados na captura de movimento ou áreas relacionadas. Nesta seleção discorre-se sobre sistemas comerciais e da área acadêmica, focando em suas etapas de rastreamento.

No Capítulo 3, o sistema *OpenMoCap* é detalhado, são apresentadas as etapas do funcionamento e a arquitetura do sistema utilizado para a comparação dos algoritmos.

No Capítulo 4, o primeiro dos métodos de rastreamento é explicado. Neste capítulo é descrito o funcionamento do filtro de Kalman, um estimador de estados recursivo e linear.

No Capítulo 5, o segundo método de rastreamento é explicado. Detalha-se o funcionamento do algoritmo CONDENSATION, que implementa um filtro de partículas não linear.

No Capítulo 6, o terceiro método de rastreamento é explicado. É descrito o funcionamento do filtro Alfa-Beta, um método de rastreamento simples e linear, sendo o rastreador inicialmente empregado no sistema *OpenMoCap*.

No Capítulo 7, de resultados experimentais, inicialmente, o hardware e os parâmetros dos algoritmos utilizados nos experimentos são explicitados. Posteriormente, são descritos os dois tipos de experimentos realizados, para medir a robustez e a carga computacional dos algoritmos. Finalmente, durante o relato dos testes seus resultados são comparados.

No Capítulo 8, o último desta dissertação, conclusões e sugestões de trabalhos futuros são apresentadas e discutidas.

Capítulo 2

Trabalhos Relacionados

Neste capítulo, são apresentados alguns trabalhos relacionados ao rastreamento de marcadores para a captura de movimento, ou relacionados ao rastreamento de outras entidades que podem ser relacionados aos marcadores da captura de movimento. Alguns sistemas comerciais de captura óptica de movimento são citados para que se tenha conhecimento sobre as diferentes aplicações disponíveis e o que se sabe sobre seus rastreadores. Também, são comentados trabalhos acadêmicos recentes, relacionados ao propósito deste trabalho.

2.1 Sistemas Comerciais

Os sistemas comerciais 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. Por serem sistemas que visam o lucro, para não ajudar possíveis concorrentes, seus códigos são fechados e nenhuma informação sobre o funcionamento dos rastreadores utilizados é divulgada.

A empresa *Vicon*[1] oferece diversos sistemas comerciais de captura óptica de movimento usando marcadores passivos e câmeras inteligentes. O rastreamento dos marcadores é feito por um *hardware* especial inserido nas câmeras. Um exemplo é o sistema *Vicon* MX composto por até dez câmeras capazes de detectar a posição dos marcadores a 515Hz, com resolução de 4 *megapixel*. Porém, o custo de um sistema desses é proibitivo para o uso doméstico ou mesmo em pequenos laboratórios, sendo superior a US$ 100.000,00.

[1] http://www.vicon.com

A *NaturalPoint*[2] possui em sua linha 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 V100R2, capazes de capturar pontos a 100Hz com resolução de 1 *megapixel*. A detecção dos pontos é feita pelo hardware da câmera que retorna ao software as coordenadas juntamente com a imagem capturada, o rastreamento é feito via software utilizando apenas as coordenadas retornadas. 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, US$ 6.000,00, quando comparado aos sistemas da *Vicon*, mas a resolução máxima e a taxa de captura são bem mais baixas.

Outro sistema óptico comercial de captura de movimento é o *IMPULSE*, produzido pela empresa *PhaseSpace*[3]. O sistema utiliza marcadores ativos, suportando até 256 marcadores compostos por *Light Emitting Diode*s (LEDs) que possuem uma identificação única, dessa forma, o problema de troca da semântica dos marcadores (*marker swapping*) não ocorre com esse sistema. As câmeras utilizadas no produto possuem resolução de aproximadamente 12 *megapixels* e conseguem capturar dados a uma velocidade de 480Hz. A faixa de preço dessa solução encontra-se entre o sistema da *Natural Point* e o da *Vicon*.

2.2 Trabalhos Acadêmicos

Nesta seção, são citados os trabalhos da área acadêmica relacionados com a captura de movimento que descrevem os algoritmos de rastreamento empregados. Outros trabalhos que descrevem o rastreamento em outros contextos, mas que podem ser aproveitados para a captura de movimento, também são citados. A maioria dos trabalhos são feitos para propósitos específicos e são compostos apenas por um software combinado a um *hardware* de propósito geral. Não foram encontrados trabalhos que comparassem diferentes métodos de rastreamento empregados na captura de movimento.

Uma extensa avaliação dos sistemas de captura de movimento foi feita por Gavrila [1999], Moeslund [2000], Moeslund & Granum [2001] e Moeslund et al. [2006]. Observa-se que entre os métodos mais utilizados para o rastreamento de marcadores estão o filtro de Kalman e o CONDENSATION. Ambos são métodos estocásticos de

[2]http://www.naturalpoint.com/optitrack
[3]http://www.phasespace.com

previsão da posição dos marcadores, sendo o primeiro um método baseado em álgebra linear e no modelo oculto de Markov, no qual o próximo estado só depende das leituras atuais e do estado anterior [Kalman et al., 1960]. Já o algoritmo ***CON****ditional* ***DENS****ity Propag****ATION*** *for Visual Tracking* (CONDENSATION) utiliza modelos de movimento não linear, diferentemente do filtro de Kalman padrão, permitindo modelos mais complexos [Isard & Blake, 1998].

Outras variações do filtro de Kalman também são encontradas na literatura. Entre elas, o filtro de Kalman estendido [Julier et al., 1995; Julier & Uhlmann, 1996, 1997], que trabalha com equações não lineares, sendo necessário realizar uma linearização da estimativa corrente utilizando séries de Taylor. Por esta variação ser custosa computacionalmente, outra forma também não linear foi proposta, chamada de filtro de Kalman *unscented*. Com este método não é necessário linearizar as equações. Isto é possível através do uso de uma técnica conhecida como transformação *unscented* [Welch & Bishop, 1995; Julier et al., 2002].

Em Sundaresan & Chellappa [2005], um sistema de captura de movimento sem marcadores é descrito. O sistema é baseado em múltiplas câmeras e o corpo humano é representado utilizando super-quádricas. Foram desenvolvidos algoritmos para capturar o modelo do corpo humano e estimar a posição inicial. O rastreamento é feito utilizando um filtro de Kalman estendido iterativo, modificado para trabalhar com a representação do corpo humano criada pelos autores. O algoritmo criado combina informação espacial e temporal de uma maneira nunca utilizada.

O *OpenMoCap*, sistema de captura de movimento no qual os diferentes rastreadores foram implementados e avaliados, utiliza o filtro Alfa-Beta [Yoo & Kim, 2003]. Este filtro preza por ser simples, e por isso possui um custo computacional baixo, sendo bastante eficiente. Neste filtro, a próxima posição é prevista baseando-se na velocidade calculada no estado anterior, considerando o movimento como retilíneo e uniforme. Esta velocidade prevista sofre uma suavização definida pelos parâmetros $alfa$ e $beta$, que são obtidos empiricamente.

Em Figueroa et al. [2003, 1998], é apresentada uma abordagem para rastrear marcadores para a análise de movimentos humanos (análise de marcha, esportes e outras aplicações na área biomédica). Neste trabalho, dois rastreadores foram avaliados: um deles é o filtro de Kalman, citado anteriormente, e o outro é chamado extrapolação, ambos considerando a posição bidimensional e velocidade dos marcadores. Este método é baseado em uma função de extrapolação para cada coordenada, cujos parâmetros são definidos experimentalmente. Utilizando como entrada os valores obtidos em quadros anteriores (no caso, três), é possível obter uma predição da posição do marcador para o quarto quadro. Entretanto, os experimentos de [Figueroa, 1998] demonstraram que o

filtro de Kalman foi superior à extrapolação. O sistema usa câmeras com resolução de aproximadamente 1 *megapixel* e é pós-processado, isto é, sequências de vídeos devem ser gravadas para depois se obter os resultados da seção de captura.

Castro et al. [2006] apresentam um sistema de captura de movimento baseado em marcadores passivos focado em análise de marcha, chamado *SOMCAD3D*. Assim como o trabalho desenvolvido por Figueroa et al. [2003], ele também é pós-processado. O rastreamento dos pontos de interesse é feito através da interpolação de curvas (*splines* cúbicas), utilizando as coordenadas dos pontos nos quadros anteriores. Em momentos de perda dos pontos de interesse, a intervenção do operador do sistema é necessária, e a interpolação é retomada utilizando as informações de antes da perda e do momento em que o ponto é novamente detectado.

Em Morais et al. [2005]; Morais [2005], um sistema para o rastreamento e contagem de peixes é apresentado. Novamente, o filtro de Kalman e o CONDENSATION são citados, mas os autores utilizam no seu trabalho um método chamado *A Bayesian Multiple-Blob Tracker* (BraMBLe) [Isard & MacCormick, 2001], considerado mais condizente com o problema. Este método é baseado no algoritmo CONDENSATION, juntamente com um modelo de observação probabilístico capaz de separar o plano de fundo dos objetos de interesse. Como no OpenMoCap, a segmentação entre objetos de interesse e plano de fundo já é realizada em outra etapa do processo, não seria preciso utilizar a etapa de treinamento do BraMBLe necessária para realizar esta separação. Portanto, apenas a tarefa de rastreamento realizada pelo próprio CONDENSATION seria necessária para este trabalho.

Zeng et al. [2009] descrevem um trabalho para a melhoria do rastreamento de objetos em vídeos utilizando o filtro de Kalman. Neste trabalho, a intenção dos autores é realizar o rastreamento em tempo real de objetos genéricos em vídeos. Segundo eles, uma das maiores causas de perdas do rastreamento se deve à oclusão, mudança de escala do objeto e troca do objeto rastreado pela proximidade de um similar. Além disso, a execução em tempo real pode ser comprometida pelo detector de características utilizado. Para resolver os problemas, eles focam no desenvolvimento de um detector de características mais robusto e utilizam o filtro de Kalman para diminuir a área de busca e, assim, poderem aplicar este método mais custoso em tempo real.

Em Marron et al. [2007] é feita uma comparação entre dois métodos de rastreamento, sendo um deles baseado no filtro de Kalman e outro no filtro de partículas (CONDENSATION). Neste trabalho, o propósito dos autores é rastrear múltiplos objetos. Para isso, o filtro de Kalman foi modificado com a inclusão de um algoritmo de associação probabilística de dados (PDA), o que, segundo os autores, deixou o método com um desempenho pior que o filtro de partículas. O filtro de partículas também

sofreu modificações com a inclusão de um algoritmo de agrupamento, que visa melhorar sua robustez. Ao final dos experimentos, os autores concluíram que o filtro de partículas foi mais robusto e que o filtro de Kalman foi prejudicado pela sensibilidade e custo do algoritmo PDA. Ainda assim, o filtro de Kalman é indicado para situações mais simples e o filtro de partículas para situações complexas, ambos executaram em tempo real.

Em A. Kamalvand & Tran [2004] o filtro de Kalman e o algoritmo CONDENSATION são empregados para rastrear mísseis. Segundo os autores, durante o lançamento, o míssil pode lançar partículas que confundem os rastreadores. Assim, a robustez à detecção errônea provocada por estas partículas foi avaliada, além da exatidão na identificação e o tempo necessário para identificar o alvo. Após os experimentos, concluiu-se que o filtro de Kalman foi mais robusto na identificação dos mísseis, apesar de gastar um tempo maior para identificar o alvo.

Bazzani et al. [2009] empregaram algoritmos baseados no filtro de Kalman e no filtro de partículas (CONDENSATION) para rastrear múltiplos alvos, com o propósito de vigilância. Os métodos empregados foram o filtro de Kalman multi hipóteses (MHKF) e o filtro híbrido de junção separável (HJS), baseado no filtro de partículas. Além dos métodos de rastreamento, também foi empregado um algoritmo para remover o fundo da cena e este resultado é passado para os métodos rastreadores. Os métodos foram avaliados em uma base de vídeos reais, PETS 2009. Ao final dos testes, os autores concluíram que ambos os métodos apresentaram resultados robustos, mas o método baseado no filtro de partículas apresentou um resultado ligeiramente melhor em manter a identificação do alvo após uma oclusão.

2.3 Considerações

Neste capítulo, foram descritos alguns sistemas comerciais de captura de movimento, relatando o que se sabe sobre o rastreamento de seus marcadores. Estes sistemas são os mais modernos encontrados na área e os mais utilizados por produtoras de filmes, comerciais e jogos. Todos estes sistemas conseguem realizar a captura em tempo real, ou seja, os resultados podem ser vistos simultaneamente com o movimento do objeto alvo. A visualização em tempo real ajuda a minimizar erros na captura, já que erros na calibração das câmeras ou troca de marcadores podem ser detectadas rapidamente. Estes sistemas são precisos e robustos, mas para isso necessitam sempre do conjunto de software proprietário e hardware especial. Desta forma, não é possível fazer combinações entre os diferentes sistemas, o que obriga a realizar a compra de

todo o sistema, o que geralmente é dispendioso.

Os trabalhos acadêmicos são geralmente criados para propósitos específicos e sua robustez não é comparada com sistemas comerciais. Alguns são voltados para o rastreamento de outras entidades, como peixes [Morais et al., 2005], ou para outros fins como análise de marcha [Figueroa et al., 2003]. Embora eles não estejam na área de captura de movimento, neles existem comparações entre outros rastreadores que foram produtivas para este trabalho.

Além destes trabalhos acadêmicos, destacam-se pesquisas na área de captura de movimento relatando que a maioria dos sistemas de captura de movimento utilizam o filtro de Kalman e o algoritmo CONDENSATION como rastreadores. Trabalhos visando comparar diferentes tipos de rastreadores para a captura de movimento não foram encontrados, de modo que se espera que este trabalho venha a auxiliar no desenvolvimento destes sistemas, ou de sistemas para outros fins, assim como trabalhos de outras áreas foram úteis para a realização deste.

Capítulo 3

O Sistema OpenMoCap

Neste capítulo, são descritos a estrutura e funcionamento do sistema de captura de movimento *OpenMoCap*. Este foi o sistema utilizado para se fazer a comparação entre os métodos de rastreamento e é para este sistema que se pretende encontrar o rastreador que melhor se aplica ao seu propósito. A Figura 3.1 exibe a interface do sistema.

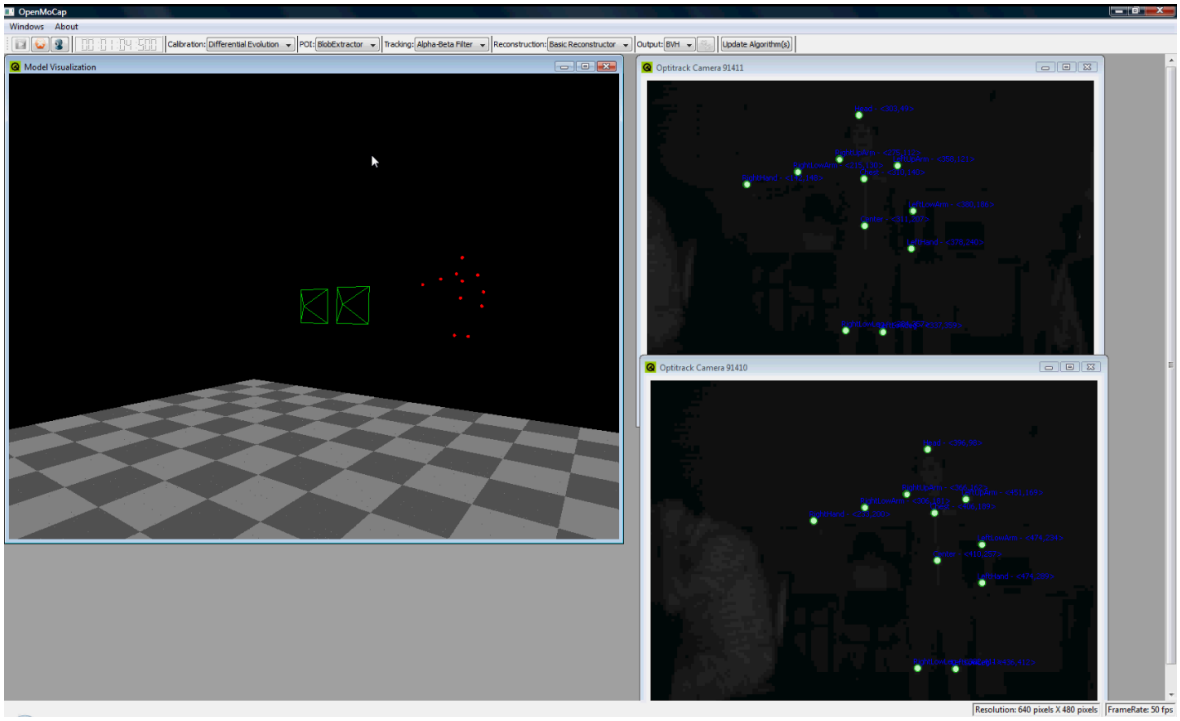

Figura 3.1. Interface do sistema OpenMoCap, [Flam, 2009].

3.1 Etapas da captura de movimento

O sistema *OpenMoCap* segue as mesmas etapas de captura de movimento descritas por Moeslund & Granum [2001] e citadas na Introdução, sendo elas: **inicialização**, **rastreamento**, **estimação de pose** ou **reconstrução** e **reconhecimento**. Estas etapas são exemplificadas pela Figura 3.2.

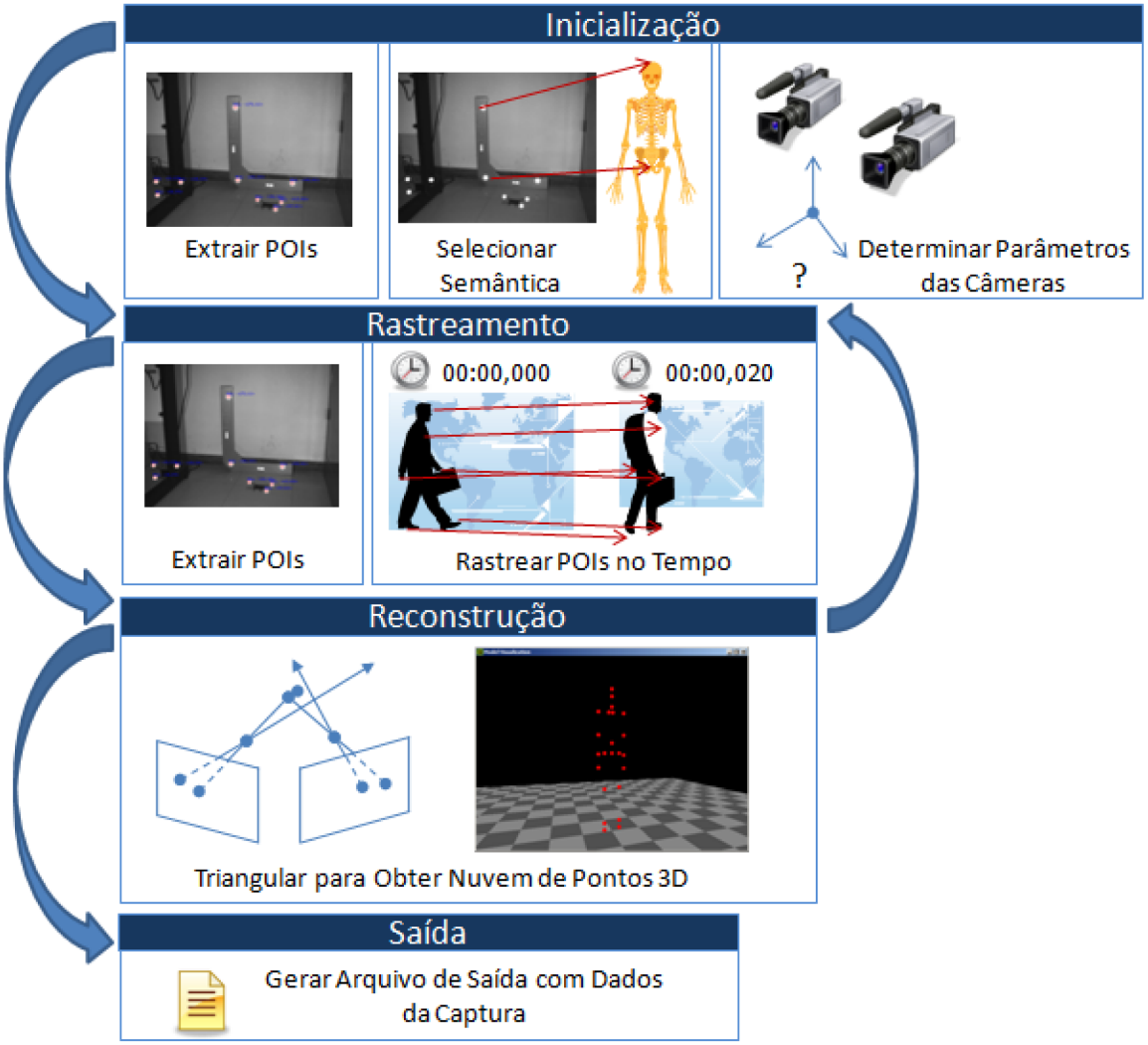

Figura 3.2. Fluxograma da captura de movimento no OpenMoCap, [Flam, 2009].

O fluxograma mostra que a **inicialialização** ocorre apenas na fase inicial da captura, enquanto as outras três etapas se alternam e se repetem durante toda a execução do programa, até que a gravação da captura do movimento seja interrompida. A seguir, as tarefas executadas em cada etapa são detalhadas.

3.1.1 Inicialização

Nesta etapa, são executadas todas as tarefas necessárias para que o sistema esteja preparado para a captura do movimento. A primeira que deve ser executada, antes que se dê início à captura do movimento, é a calibração das câmeras. A calibração das câmeras consiste em determinar a posição de cada câmera no espaço 3D, dado que os parâmetros intrínsecos das câmeras já tenham sido definidos. Esta informação é necessária para que se possa obter as coordenadas 3D de cada ponto capturado. Para efetuar a calibração foi utilizado o trabalho desenvolvido por Fraga & Silva [2008] que utiliza a evolução diferencial para calibrar um conjunto de câmeras.

Após a calibração das câmeras, passa-se para o passo seguinte da **inicialização**, que consiste em definir a semântica dos pontos de interesse (*Point of Interest*s (POIs), na sigla em inglês) correspondentes aos marcadores. Nesta etapa, todos os marcadores detectados pelas câmeras são exibidos na interface do programa como pontos de interesse, mas sem uma semântica definida. A partir de um modelo previamente carregado no sistema, como por exemplo um esqueleto que representa uma pessoa. Para cada ponto representando uma articulação deste esqueleto, deve-se definir a semântica dos pontos de interesse. Esta semântica define uma hierarquia, com o centro de massa no topo da mesma.

A definição da semântica de cada ponto é feita de maneira bastante simples neste sistema. Basta clicar sobre um ponto ainda sem semântica, que uma caixa de diálogo com todas as opções de semântica possíveis é exibida, permitindo que o usuário defina a semântica ou o nome daquele ponto, como mostrado na Figura 3.3. Esse processo deve ser feito para cada uma das câmeras, de forma que após este passo o sistema tenha como associar os pontos de diferentes câmeras para assim obter a coordenada 3D do ponto no mundo real.

Concluídos os passos de definição da semântica dos pontos de interesse e calibradas as câmeras, a etapa de **inicialização** está terminada. A partir deste ponto o sistema já está apto a realizar todas as outras etapas que serão repetidas até o final da captura do movimento.

3.1.2 Rastreamento

É nesta etapa que o trabalho descrito nesta dissertação está inserido. Durante toda a captura do movimento, começando a partir do momento que a semântica de um POI é definida, o **rastreamento** é realizado. As principais funções do rastreamento em um sistema de captura de movimento são: 1. Manter a semântica definida na etapa

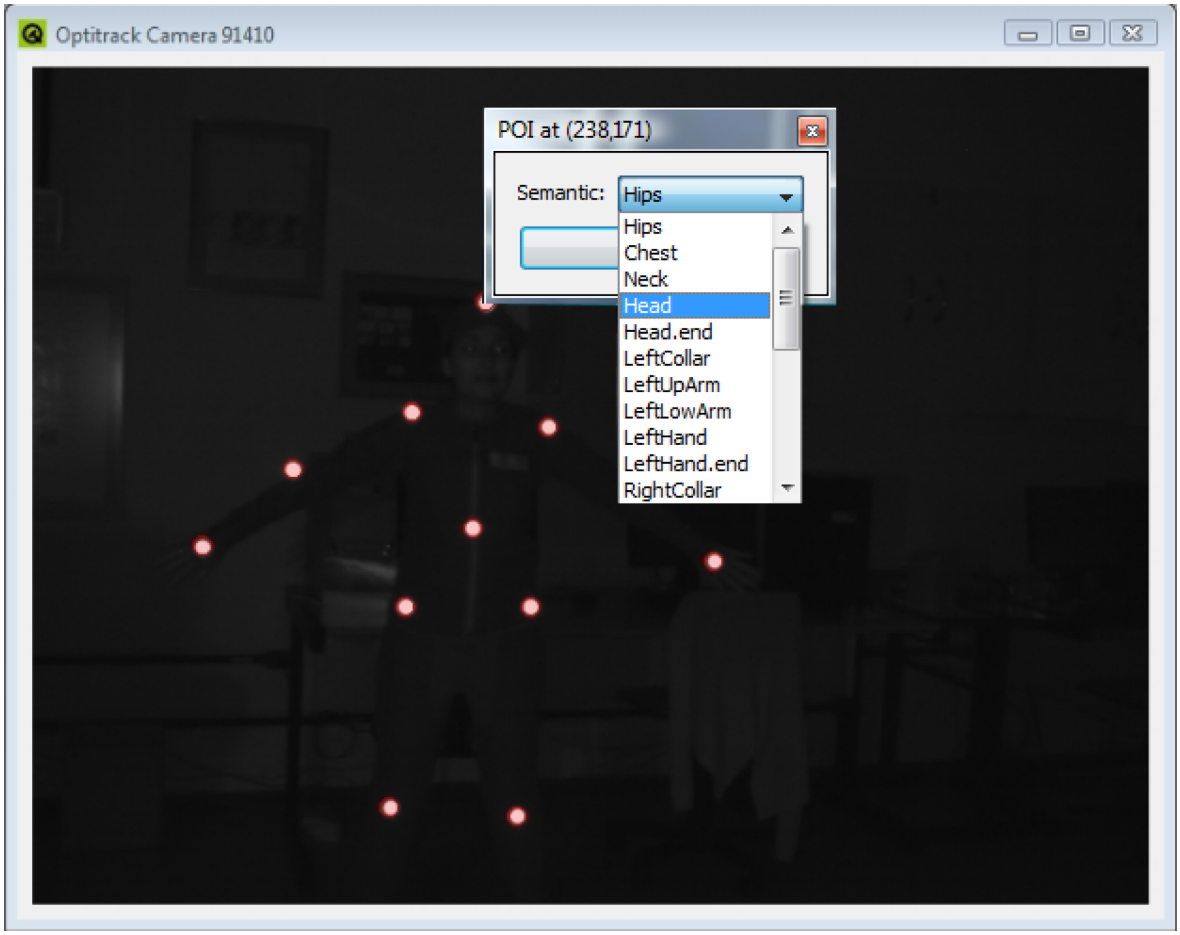

Figura 3.3. Janela de seleção de semântica dos POIs no OpenMoCap, [Flam, 2009].

de **inicialização** durante todo o processo de captura; 2. Reduzir a área de busca por POIs.

O rastreador inicialmente implementado neste sistema foi o filtro Alfa-Beta, também avaliado neste trabalho. Independentemente do método de rastreamento empregado, o funcionamento destes segue sempre o princípio de prever a próxima posição do marcador. A partir da previsão desta nova possível posição, que varia para cada método, uma área de busca é definida ao redor deste ponto. Além da posição do ponto previsto diferir para cada método, o tamanho desta área também se difere, variando inclusive durante a execução, em métodos que implementam uma área de busca variável.

A Figura 3.4 demonstra os passos do rastreamento, envolvidos entre cada quadro da captura. No primeiro quadro, um esqueleto com a semântica de cada ponto já definida é exibido. No quadro seguinte, um dos pontos se moveu e foi encontrado dentro da área de busca definida ao redor da posição estimada do POI. Por ter sido

encontrado apenas um ponto dentro desta área, a semântica do POI que estava sendo rastreado foi atribuída a este ponto. No próximo quadro, foram encontrados dois pontos dentro da área de busca, neste caso o sistema atribui a semântica ao ponto mais próximo da posição estimada pelo preditor empregado.

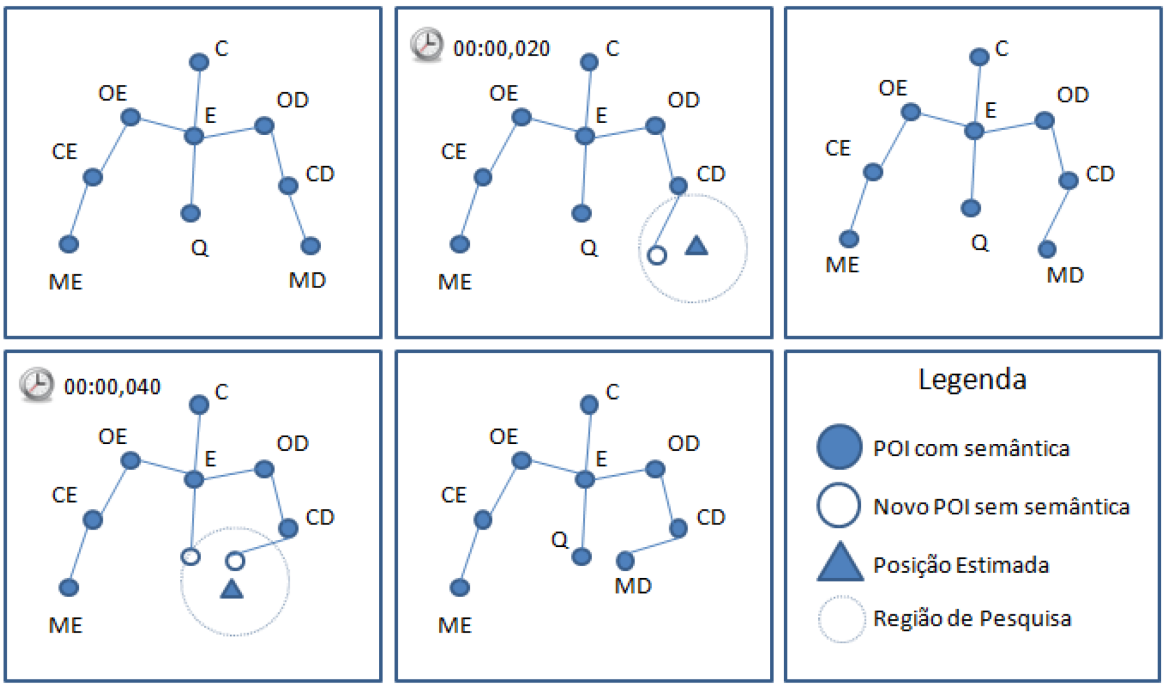

Figura 3.4. Etapas do rastreamento. Aquelas imagens marcadas com o tempo representam movimentos de POIs ocorridos na cena. As sem o tempo são o resultado do processamento do rastreador, [Flam et al., 2009].

No *OpenMoCap*, o rastreamento é um processo local, sendo realizado nos dados retornados por cada câmera. Assim, cada câmera possui a sua própria instância de um rastreador, que o utiliza para rastrear os pontos presentes na imagem gerada por ela. Sendo, portanto, um processo independente das outras câmeras e do número de câmeras utilizadas.

3.1.3 Estimação de pose

Esta etapa também é conhecida como **reconstrução** e é nela que as coordenadas 3D dos marcadores são obtidas. Utilizando os parâmetros obtidos na calibração das câmeras (realizada na etapa de **inicialização**) e a posição dos pontos em cada uma das câmeras, relacionados por suas semânticas, é possível obter as coordenadas no espaço 3D dos pontos de interesse.

A obtenção das coordenadas nos marcadores no mundo real é realizada através de um processo chamado de triangulação [Hartley & Sturm, 1997]. Neste processo,

a posição tridimensional dos POIs é obtida por meio de suas projeções nos planos de imagem das câmeras e suas respectivas matrizes de projeção. Estes dados compõem um sistema que pode ser resolvido por meio de decomposição em valores singulares, *Singular Value Decomposition* (SVD) [Press et al., 1992], obtendo-se as coordenadas procuradas.

O processo de **estimação de pose** também se repete durante todo o processo de captura. Por esta etapa ser realizada durante todo o processo de captura, o *OpenMoCap* é um sistema de captura em tempo real, de forma que a reconstrução dos movimentos do ator, em 3D, pode ser obtida ao mesmo tempo que os movimentos são realizados em frente às câmeras. Isto possibilita que erros de calibração ou troca de semântica dos marcadores sejam detectados durante a captura.

3.1.4 Reconhecimento

Esta é a etapa final do processo de captura. Durante o **reconhecimento**, as coordenadas tridimensionais dos pontos obtidas durante a etapa anterior são convertidas para um formato de saída conhecido. Entre os formatos mais utilizados, há o TRC [Motion Analysis Corporation, 2010], empregado pelo *OpenMoCap*, (Figura 3.5) e o BVH [University of Wisconsin-Madison, 2010] que encontra-se em fase de implantação.

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	Y3	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 3.5. Amostra do conteúdo de um arquivo TRC, [Flam, 2009].

Este arquivo é a saída final do processo de captura. Nele estão armazenados todos os movimentos realizados pelo ator, representados pelo movimento de cada um dos POIs gerados pelos marcadores. Esta sequência de movimentos será importada pelos programas que por ventura quiserem utilizar os movimentos capturados.

Embora esta seja a última etapa, ela também ocorre durante todo o processo de captura, pois o arquivo de saída é sempre atualizado com os novos dados obtidos em cada instante da captura de movimento.

3.2 Arquitetura do sistema

O sistema *OpenMoCap* foi implementado utilizando práticas de programação que o tornassem modular e paralelo. A modularidade tem o objetivo de permitir alterar ou substituir módulos facilmente como, por exemplo, substituir o detector de POIs por um que não utilize marcadores, ou implementar outros rastreadores, como foi feito neste trabalho. O paralelismo visa aproveitar a tendência de processadores com núcleos múltiplos e tornar o sistema escalável.

A Figura 3.6 demonstra como o paralelismo (as *threads*) e os módulos estão relacionados. Observando a figura, percebe-se que uma *thread* principal, contendo os módulos *Main User Interface* e *MoCap Core*, rege todas as outras. Em um nível mais abaixo na arquitetura, há outra *thread* que executa o módulo *Capture Controller* responsável por tarefas como a calibração das câmeras, estimação de pose e reconhecimento. No mesmo nível, também há a *thread* responsável pela visualização do resultado da captura em tempo real.

Em um último nível da arquitetura, há uma *thread* alocada para cada câmera instanciada no sistema. Cada um destes fluxos de execução das câmeras contém o módulo *Video Controller*, que por sua vez gerencia os módulos *Abstract Camera*, *Abstract POI Detector* e *Abstract Tracker*. Desta forma, cada câmera é responsável pela detecção e rastreamento dos seus pontos, sendo tarefas locais e independentes das outras câmeras. Esta forma de implementação permite uma grande escalabilidade do sistema, de forma que o número de câmeras é ilimitado desde que se tenha núcleos de processamento suficientes para alocar as *threads* de cada câmera.

As implementações necessárias para este trabalho foram realizas em sua maioria no módulo *Abstract Tracker*. Devido à alta modularidade do sistema, foi possível mensurar os custos computacionais apenas das tarefas envolvidas no rastreamento, sem a influência de outras tarefas necessárias à captura de movimento, como detecção de POIs ou visualização. Logo, os custos computacionais, relatados no Capítulo 7 de experimentos, só dizem respeito a este fluxo da execução.

Os algoritmos implementados neste trabalho, como todo o sistema *OpenMoCap*, foram codificados utilizando a linguagem C++ e bibliotecas que facilitassem o desenvolvimento. A principal ferramenta de código utilizada neste trabalho foi a biblioteca

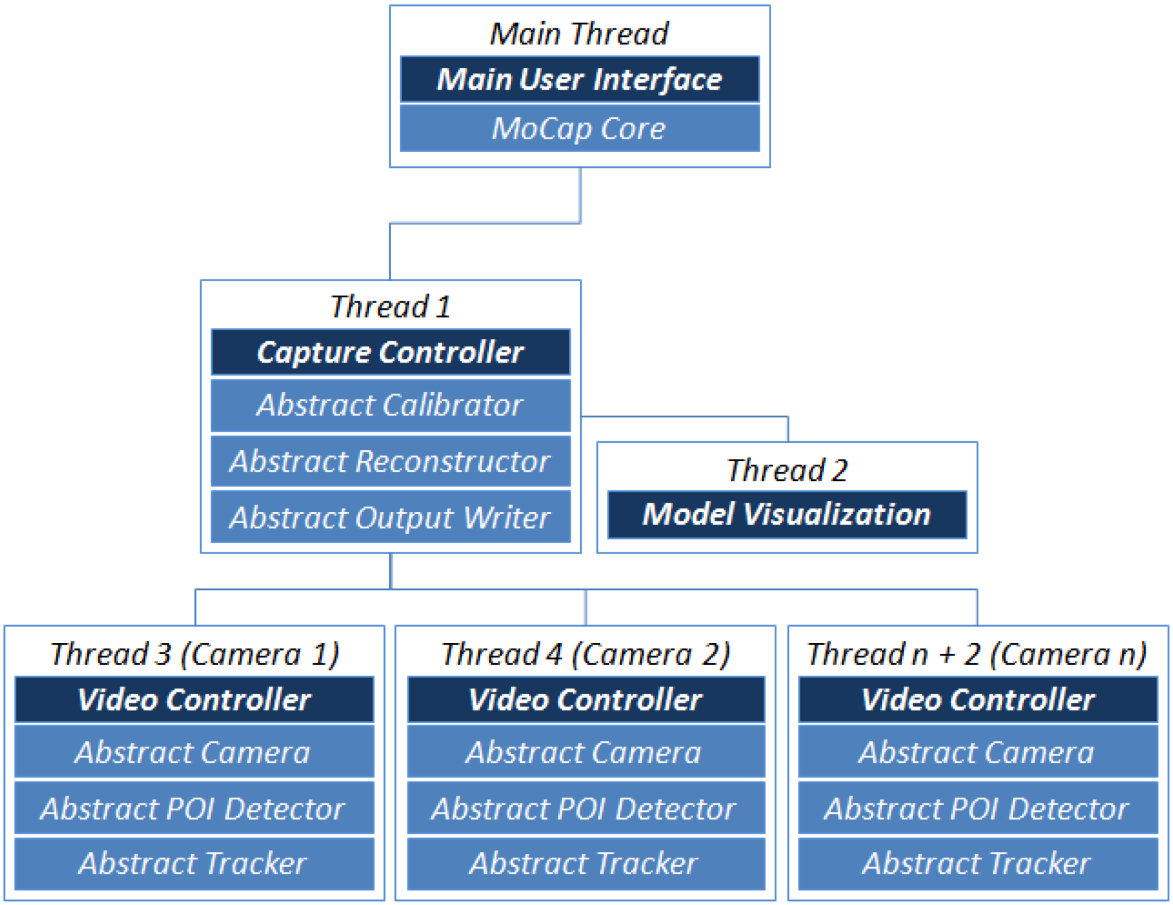

Figura 3.6. Diagrama da arquitetura do OpenMoCap, [Flam et al., 2009].

OpenCV, criada pela [Intel Corporation, 2010], para demonstrar o desempenho de seus processadores que utilizavam instruções especiais, e atualmente mantida por [Willow Garage, 2010]. Esta biblioteca possui as estruturas de armazenamento das variáveis utilizadas pelo filtro de Kalman e pelo algoritmo CONDENSATION.

3.3 Considerações

Neste capítulo, foram descritas as etapas de captura de movimento envolvidas no sistema *OpenMoCap*. Cada uma das quatro etapas: **inicialização**, **rastreamento**, **estimação de pose** ou **reconstrução** e **reconhecimento** foi explicada separadamente. Além disso, a arquitetura do sistema foi detalhada na segunda seção deste capítulo. Mostrou-se que o *OpenMoCap* é um sistema paralelo e escalável, utilizando para isso *threads* e uma implementação modular.

No próximo capítulo, o primeiro dos métodos de rastreamento empregados neste trabalho, o filtro de Kalman, será descrito em detalhes.

Capítulo 4

Filtro de Kalman

Neste capítulo, é descrito o funcionamento do filtro de Kalman. Inicialmente, são citadas suas principais características e o seu funcionamento é explicado através de um exemplo simples. Logo após, na segunda seção, aprofunda-se em suas definições matemáticas, sempre as relacionando com o exemplo inicial dado. Finalmente, na última seção, relacionam-se as modificações de instanciação necessárias para a aplicação do filtro ao problema de rastreamento de marcadores deste trabalho.

4.1 Descrição do método

O filtro de Kalman é um estimador recursivo de estados que utiliza equações lineares e se baseia no princípio de Markov [Kalman et al., 1960]. Embora seja um filtro linear e os movimentos dos marcadores nem sempre sigam um modelo linear, é possível utilizar este filtro pelo fato do intervalo de tempo entre as imagens das câmeras ser constante e muito pequeno. Desta forma, pode-se considerar pequenos deslocamentos lineares entre os quadros com uma mínima perda de qualidade.

A segunda característica do filtro de Kalman se refere ao princípio de Markov. Segundo este modelo, a predição do estado seguinte só depende do estado atual e das observações obtidas durante este estado. Assim, para predizer as variáveis de estado, no caso posição e velocidade do marcador, é necessário apenas conhecer a posição e velocidade anterior e obter a posição atual para fazer as correções necessárias.

Como pode ser observado pelo que foi descrito no parágrafo anterior, o filtro de Kalman pode ser dividido em duas partes: correção e predição, como exibe a Figura 4.1. Na primeira etapa, utilizando as observações atuais, a predição anterior é corrigida. Na segunda, um novo estado ou nova posição do marcador é prevista, utilizando o modelo

cinemático do sistema sobre a posição corrigida. De forma recursiva, estas etapas se alternam durante a execução do filtro.

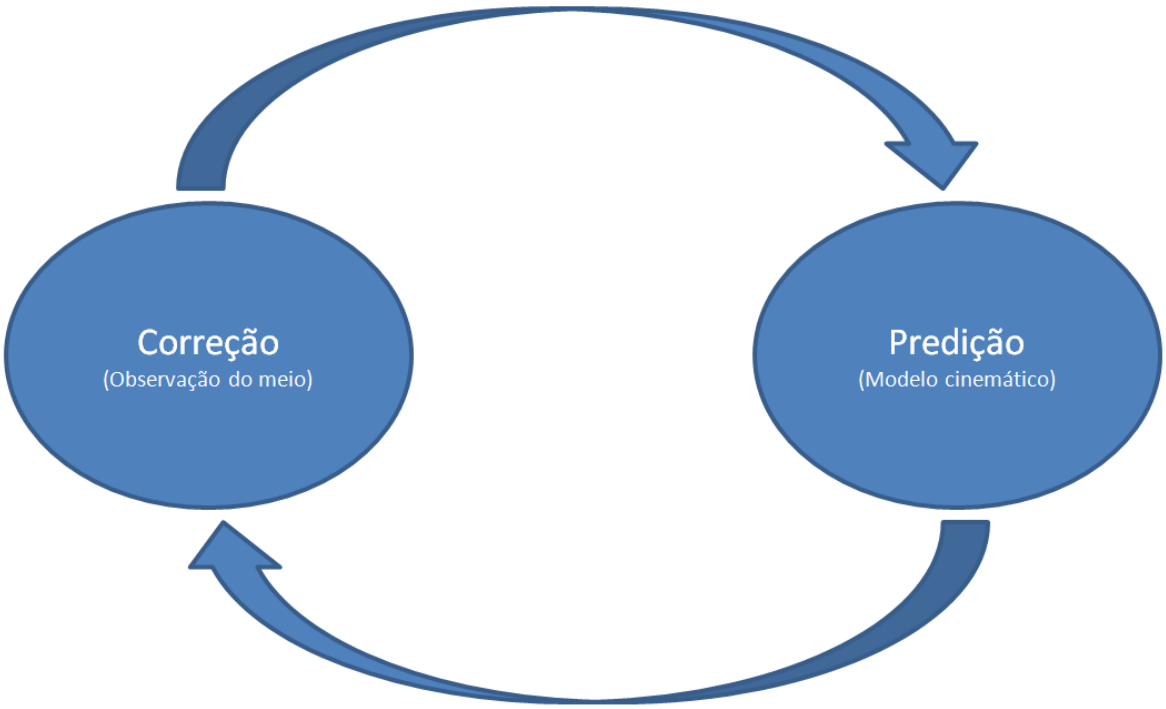

Figura 4.1. O ciclo do filtro de Kalman. Correção, quando a previsão anterior é corrigida utilizando a observação. Previsão, uma nova previsão é gerada utilizando o modelo dinâmico.

Antes de abordar as ferramentas matemáticas por trás das realizações do filtro de Kalman, será descrita uma situação fictícia na qual o uso deste filtro poderia auxiliar na localização de um ponto. Este exemplo, retirado de [Maybeck, 1979], demonstra um uso prático do filtro, que pode ser estendido para as necessidades deste trabalho.

Suponha que você está perdido no mar, à noite, e a única referência sobre a sua posição seja uma estrela. Como você não é um navegador ou astrônomo, você não possui muita certeza sobre a sua posição. Para simplificar, a sua posição será representada com apenas uma dimensão, gerando assim um estado de apenas uma variável x. A posição observada por você no tempo t_1 será definida por z_1. A incerteza sobre sua posição será representada pela gaussiana ao redor de sua posição estimada, com um desvio padrão σ_{z_1}, sendo mais larga quanto maior for a sua incerteza.

Na bibliografia sobre o filtro de Kalman, muitas vezes essa incerteza é citada como a variância ou, até mesmo, a covariância do modelo. O sentido real empregado é o de variância, como sendo o quadrado do desvio padrão. Assim, a variância de x é igual a σ^2. O uso do termo covariância também está correto, devido à propriedade que iguala a covariância de duas variáveis à variância, desde que as variáveis sejam as

mesmas, ou seja, $var(x) = cov(x, x) = (\sigma_x^2)$ (Apêndice A). Neste trabalho, trata-se a incerteza como variância.

O gráfico da Figura 4.2 mostra a função densidade de probabilidade (fdp) condicional no eixo y e a sua posição no eixo x. Esta função retorna a probabilidade de se estar em uma posição, dada uma previsão de posição feita em um tempo t. Pode-se perceber que a probabilidade de se estar na própria posição estimada é a maior, mas esta probabilidade diminui ao se afastar desta posição, seguindo uma distribuição normal. Para mais detalhes, verifique o Apêndice A.

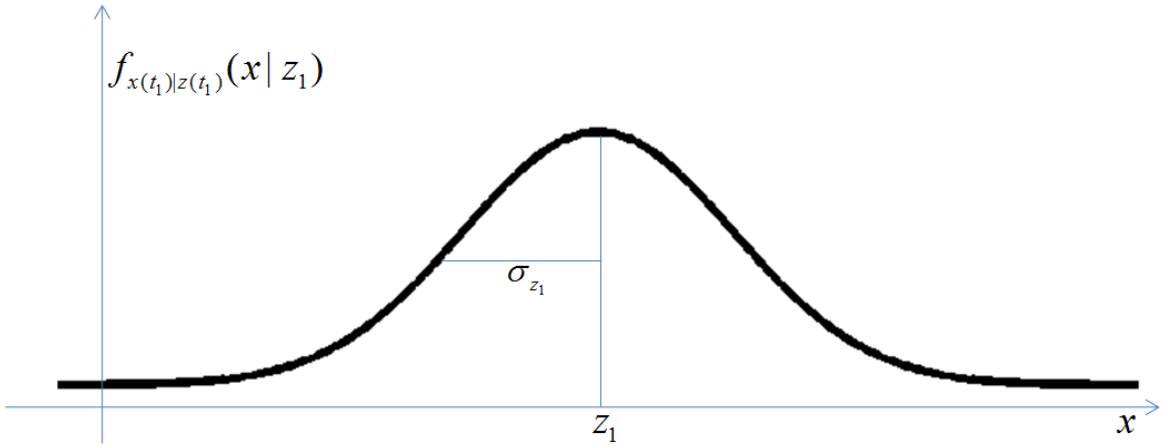

Figura 4.2. Função densidade de probabilidade condicional da observação z_1.

Agora, suponha que você pergunte a um amigo (que também está perdido com você), em um tempo $t_2 \cong t_1$, sobre a sua posição. Este amigo é um navegador experiente, que sabe inferir com mais certeza a posição a partir de estrelas. Desta forma, a representação da previsão do amigo será composta por uma gaussiana mais estreita, como demonstra a Figura 4.3.

Neste ponto, pode-se utilizar o filtro de Kalman. Com ele consegue-se obter uma estimativa da posição melhor que cada uma das previsões quando analisadas individualmente. Isto é possível porque mesmo estimativas ruins, com uma alta incerteza, trazem alguma informação. Sendo a incerteza analisada no momento de se juntar as duas estimativas, percebe-se que a estimativa com maior certeza contribui mais para a posição final, mas ainda assim a outra influencia no resultado. O resultado pode ser observado na Figura 4.4.

A forma de combinar as duas previsões e encontrar uma melhor que as duas iniciais, é feita através da Equação (4.1):

$$\hat{x}(t_2) = (\frac{\sigma_{z_2}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2})z_1 + (\frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2})z_2 = z_1 + (\frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2})(z_2 - z_1) \qquad (4.1)$$

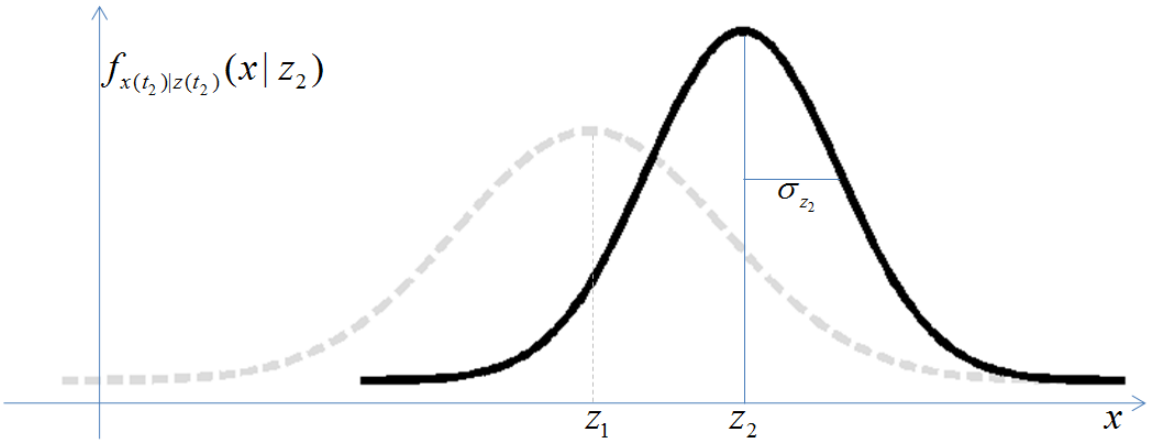

Figura 4.3. Função densidade de probabilidade condicional da observação z_2.

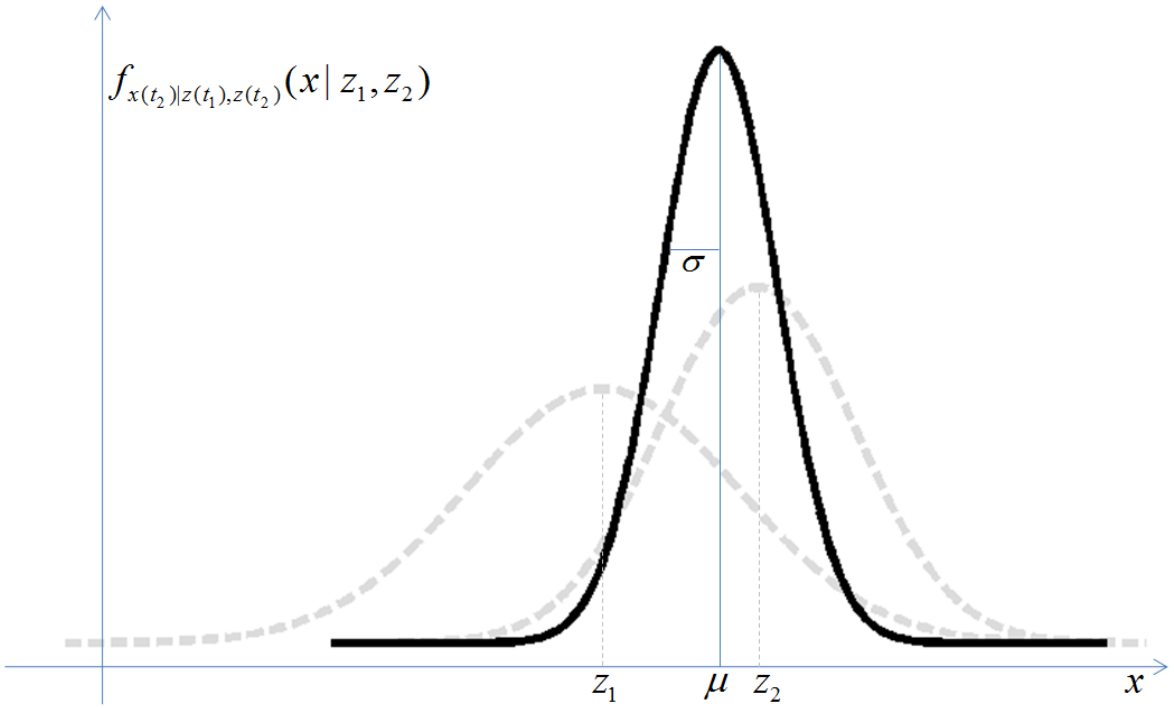

Figura 4.4. Função densidade de probabilidade condicional da fusão das observações z_1 e z_2.

Pode-se perceber o sentido desta equação fazendo algumas suposições de valores. Caso os valores de σ_1 e σ_2 sejam iguais, ou seja, a confiabilidade das duas previsões é a mesma, a equação retornaria a média das duas medidas, como seria esperado. No caso da incerteza de uma medida ser maior que a da outra, por exemplo, σ_1 ser maior que σ_2, a equação daria um peso maior à z_2, em detrimento de z_1, que é menos confiável.

A variância da combinação das duas observações é definido por:

$$\frac{1}{\sigma^2} = \frac{1}{\sigma_{z_1}^2} + \frac{1}{\sigma_{z_2}^2} \tag{4.2}$$

Nota-se que a nova variância (ou novo desvio padrão) obtida é sempre menor que as variâncias que a geraram, a não ser que uma delas seja infinitamente grande, fazendo com que o resultado seja igual à outra. Isto demonstra que o filtro de Kalman soluciona o método dos mínimos quadrados de forma ótima, segundo o critério de obter a menor incerteza possível.

A Equação (4.1) pode ser reescrita na forma utilizada nesta implementação como:

$$\hat{x}(t_2) = \hat{x}(t_1) + K(t_2)[z_2 - \hat{x}(t_1)] \tag{4.3}$$

Na qual,

$$K(t_2) = \frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} \tag{4.4}$$

Também, utilizando $K(t_2)$, conhecido como ganho de Kalman, para calcular a variância da combinação das duas observações, tem-se:

$$\sigma_x^2(t_2) = \sigma_x^2(t_1) - K(t_2)\sigma_x^2(t_1) \tag{4.5}$$

Estas equações informam que a estimativa ótima da posição $\hat{x}(t_2)$, obtida no tempo t_2, é igual à estimativa obtida no tempo t_1, $\hat{x}(t_1)$, corrigida pela diferença entre a observação atual z_2 e a estimativa anterior $\hat{x}(t_1)$. Sendo esta correção ponderada pelo ganho de Kalman, que considera a confiabilidade em cada uma delas.

Como estas equações realizam uma correção do estado anterior, elas são agrupadas na etapa de correção do filtro, mencionada no início do capítulo. Uma forma mais genérica destas equações e um diagrama relacionando as equações de cada fase serão mostrados na seção de definição das equações matemáticas do filtro de Kalman, Seção 4.2.

As demonstrações do exemplo, até agora, resolvem um problema estático, no qual as observações são feitas aproximadamente no mesmo instante de tempo ou em períodos muito próximos. No entanto, o filtro de Kalman prevê a incorporação de um modelo dinâmico, com o qual é possível realizar predições considerando o deslocamento dos objetos no tempo.

Ainda, considerando a situação hipotética descrita anteriormente, suponha que você se desloque por algum tempo antes de obter uma nova medição de sua posição. Considerando um deslocamento linear, este pode ser descrito por uma velocidade v e

também um erro q, associado a ela. Este erro deve ser modelado como ruído gaussiano branco, de média zero e desvio padrão σ_q, para que possa ser utilizado no filtro.

Assim, a nova previsão da posição, considerando o deslocamento, é dada por:

$$\hat{x}(t_3^-) = \hat{x}(t_2) + v[t_3 - t_2] \tag{4.6}$$

e

$$\sigma_x^2(t_3^-) = \sigma_x^2(t_2) + \sigma_q^2[t_3 - t_2] \tag{4.7}$$

Como expõe a Equação (4.7), este deslocamento fará com que o erro associado a este modelo dinâmico, seja incorporado ao erro da predição atual, fazendo com que a incerteza sobre a posição aumente com o deslocamento. Esta propriedade provoca o efeito de alargar a base da gaussiana à medida que o deslocamento ocorre, como é representado no gráfico da Figura 4.5.

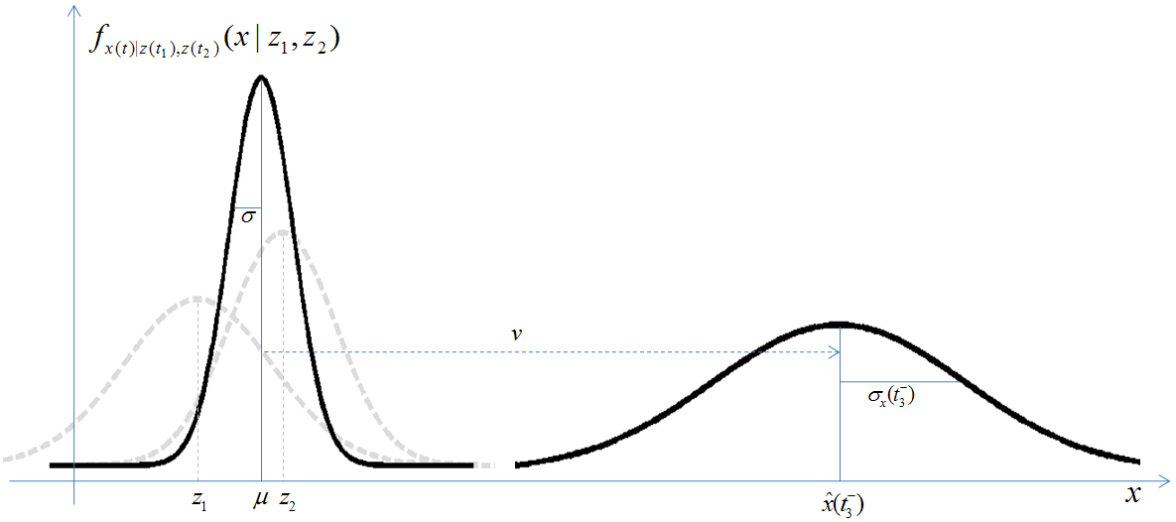

Figura 4.5. Deslocamento no tempo da fdp combinada.

Como nesta etapa um modelo dinâmico é utilizado para prever a posição ao longo do tempo, estas equações compõem a etapa de predição do filtro. Após esta etapa, uma nova medição será obtida e será dado inicio a uma nova etapa de correção, caracterizando a propriedade recursiva do filtro de Kalman.

4.2 Definições matemáticas

Tendo introduzido os conceitos do filtro de Kalman através do exemplo, pode-se retornar às definições matemáticas necessárias para utilizá-lo no rastreamento de marcado-

res. Considerando a divisão do filtro em duas etapas definidas anteriormente, correção e predição, as equações da etapa de correção são:

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H_k\hat{x}_k^-) \tag{4.8}$$

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{4.9}$$

$$P_k = (I_4 - K_k H_k) P_k^- \tag{4.10}$$

A Equação (4.8) se relaciona com a Equação (4.3) descrita no exemplo, sendo $\hat{x}_k$ o vetor de estado corrigido a ser obtido, $\hat{x}_k^-$ o vetor de estado obtido *a priori* e K_k o ganho de Kalman, obtido pela Equação (4.9) e que contém os erros associados às medições. O vetor z_k contém os valores das medições realizadas no instante k. A matriz H_k é uma matriz de conexão entre as medições (duas dimensões) e o vetor de estado (quatro dimensões). Os valores destas matrizes e vetores serão apresentados no final deste capítulo, Seção 4.3.

A Equação (4.9), se relaciona à Equação (4.4) do exemplo. Esta equação utiliza a matriz P_k^- que contêm a variância dos erros dos valores estimados e a matriz R_k que contêm a variância dos erros associados às medições, para calcular a matriz do ganho de Kalman, K_k.

A Equação (4.10) se relaciona à Equação (4.5) e se encarrega de realizar a correção da matriz de covariância de erros das variáveis de estado. A matriz P_k armazena a variância dos erros das variáveis que definem o estado do objeto, ora com valores estimados ou previstos (P_k^-), ora com valores corrigidos, P_k.

Logo após a etapa de correção, se segue a etapa de predição. Nesta etapa, o modelo dinâmico será utilizado para realizar uma estimativa, ao longo do tempo, da posição do marcador. A equações desta etapa são as seguintes:

$$\hat{x}_{k+1}^- = A_k\hat{x}_k \tag{4.11}$$

$$P_{k+1}^- = A_k P_k A_k^T + Q_k \tag{4.12}$$

A Equação (4.11) se relaciona à Equação (4.6) do exemplo, sendo responsável por aplicar as velocidades à posição dos marcadores e obter uma nova posição estimada. O estado estimado procurado é definido por $\hat{x}_{k+1}^-$ e o atual por $\hat{x}_k$, a matriz de transição A_k é definida a partir do modelo dinâmico e contêm o intervalo de tempo entre os

estados, no caso, o tempo entre os quadros de imagem da câmera.

A Equação (4.12) está relacionada à Equação (4.7), que realiza o cálculo do erro da predição. Para isso, além da matriz de transição A_k e da matriz de covariância dos erros corrigida P_k, é utilizada a matriz Q_k. A matriz Q_k representa a matriz de covariância de erros do modelo dinâmico, armazenando o erro esperado da velocidade.

Tendo definido as equações das duas etapas do filtro, pode-se representá-las em um diagrama como o mostrado na figura 4.6.

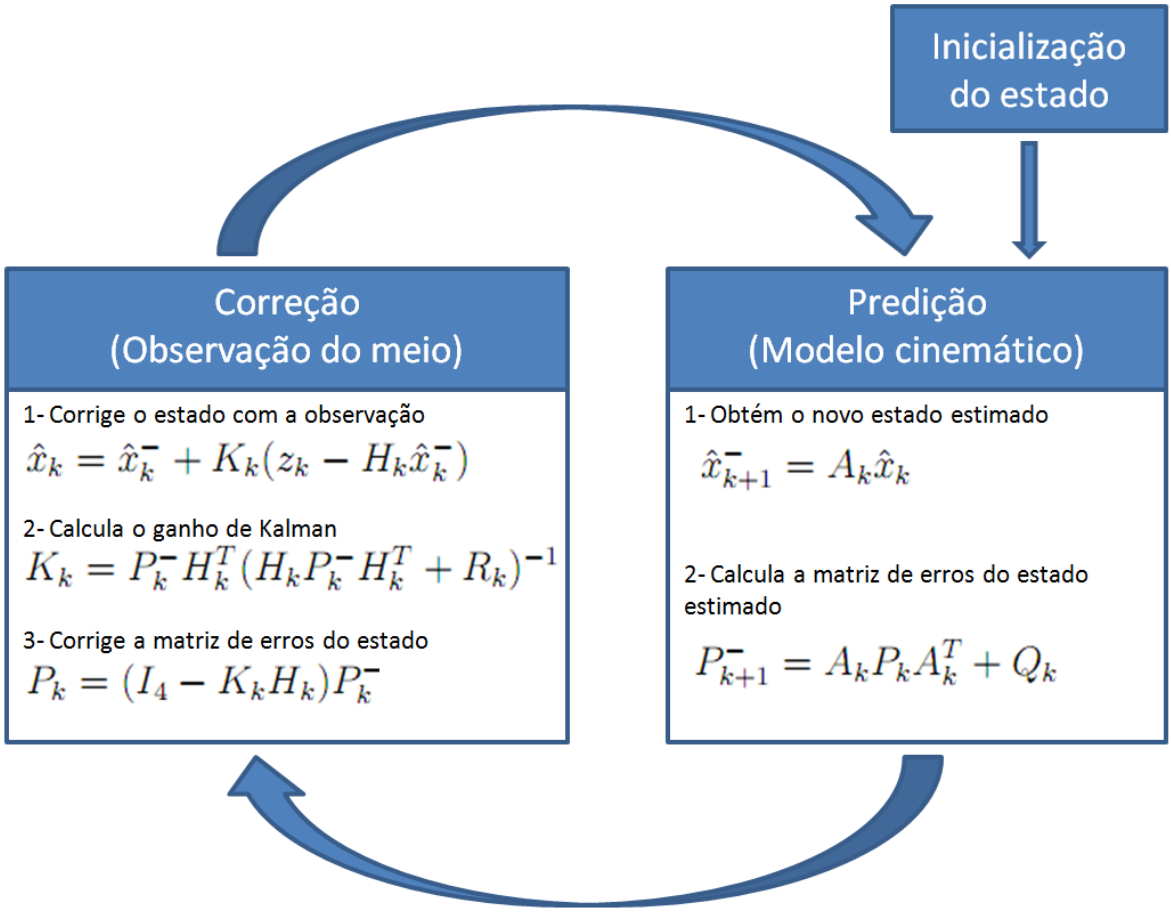

Figura 4.6. O ciclo do filtro de Kalman completo, incluindo as equações demonstradas.

Segundo Fieguth [2011], considerando as multiplicações e inversões de matrizes realizadas durante a execução do algoritmo, pode-se calcular o seu custo computacional. Este custo computacional é definido como $O(N^3)$, sendo N o número de dimensões do vetor de estado x.

4.3 Instanciação

Para utilizar o filtro de Kalman no problema de rastreamento de marcadores, é necessário definir quais os valores dos vetores e matrizes que compõem o filtro, ou seja, instanciar o filtro para o problema em questão.

O primeiro passo é definir as variáveis que irão compor o vetor de estado x_k. Neste problema precisa-se, em última análise, prever a posição dos marcadores, logo, o estado precisa conter a informação sobre a posição. Esta informação é representada por duas variáveis, a coordenada do eixo X e a coordenada do eixo Y.

Além da posição, para aplicar o modelo dinâmico e prever a posição ao longo do tempo, é necessário também armazenar a velocidade do marcador no vetor de estado. Esta velocidade é representada pelas variáveis de velocidade no eixo X e Y. Deste modo, o vetor de estado passa a ser composto por quatro variáveis: duas variáveis de posição (x e y) e duas de velocidade ($velX$ e $velY$), tomando a seguinte forma:

$$\mathbf{x_k} = \begin{bmatrix} x \\ y \\ velX \\ velY \end{bmatrix}$$

Para a aplicação do modelo dinâmico, que é baseado em um movimento retilíneo uniforme, utiliza-se a matriz A_k, definida a seguir. As entradas Δt representam o intervalo de tempo entre os quadros de imagem da câmera, sendo o tempo de deslocamento da posição do marcador de uma captura até a próxima.

$$\mathbf{A_k} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A matriz de covariância Q_k armazena os erros inerentes ao modelo, representados pela variância de cada uma das velocidades.

$$\mathbf{Q_k} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{Q_{velx}} & 0 \\ 0 & 0 & 0 & \sigma^2_{Q_{vely}} \end{bmatrix}$$

Ao rastrear uma imagem à procura de pontos de interesse, no caso, os marcadores, obtêm-se as coordenadas x e y destes pontos. Estas leituras correspondem à fase de

observação ou medição do filtro de Kalman e são representadas pelo vetor z_k, que contém as coordenadas x e y da posição do ponto de interesse.

$$\mathbf{z_k} = \begin{bmatrix} x \\ y \end{bmatrix}$$

Para as observações também é preciso definir seus erros, representados pela variância de cada uma das coordenadas e armazenados na matriz R_k.

$$\mathbf{R_k} = \begin{bmatrix} \sigma^2_{R_x} & 0 \\ 0 & \sigma^2_{R_y} \end{bmatrix}$$

Como o vetor de estado possui quatro dimensões e o vetor de observações apenas duas, para aplicar algumas equações do filtro é necessário uma matriz de transição H_k. Esta matriz torna possível operações entre estes dois vetores de dimensões diferentes e entre as matrizes empregadas no filtro.

$$\mathbf{H_k} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Por fim, tem-se a matriz de covariância de estimação dos valores P_k. Esta matriz terá seus valores definidos durante a execução do filtro, convergindo para um conjunto fixo de valores, desde que as matrizes de covariância de erros (Q e R) sejam mantidas constantes. Para inicializá-la, como se tem certeza da posição inicial do marcador, mas não da sua velocidade, defini-se o erro da posição como zero e atribuí-se um erro inicial alto para as velocidades.

$$\mathbf{P_k} = \begin{bmatrix} \sigma^2_{P_x} & 0 & 0 & 0 \\ 0 & \sigma^2_{P_y} & 0 & 0 \\ 0 & 0 & \sigma^2_{P_{velx}} & 0 \\ 0 & 0 & 0 & \sigma^2_{P_{vely}} \end{bmatrix}$$

4.4 Considerações

Para aumentar a robustez do filtro, uma implementação que varia os valores de Q foi utilizada [Welch & Bishop, 1995]. Esta variação é baseada na velocidade do marcador, de modo que quanto maior a velocidade, maior será o erro do modelo de transição armazenado na matriz Q. Desta forma, a matriz P_k não convergirá para valores fixos durante a execução, e estes valores variáveis serão utilizados para definir a área de busca, que também será variável e dependente da velocidade do marcador.

O capítulo seguinte descreve o segundo método utilizado neste trabalho, o algoritmo CONDENSATION.

Capítulo 5

Algoritmo CONDENSATION

Neste capítulo, é descrito o funcionamento do algoritmo CONDENSATION. Inicialmente, suas principais características são relacionadas e a técnica sobre a qual o algoritmo se baseia é explicada, a *factored sampling*. Posteriormente, demonstra-se como o algoritmo se utiliza desta técnica em seu funcionamento e os passos empregados. Finalmente, na última seção, relacionam-se as modificações de instanciação necessárias para que se aplique o algoritmo ao problema de rastreamento de marcadores, sendo que, para o caso do algoritmo CONDENSATION, esta tarefa foi empregada de duas maneiras diferentes.

5.1 Descrição do método

O algoritmo CONDENSATION é um estimador de estados não linear, bayesiano e que utiliza o princípio de Markov; sendo também conhecido como filtro de partículas ou Monte Carlo. Este algoritmo utiliza conceitos probabilísticos e o conceito de partículas para rastrear objetos, tendo sido desenvolvido para detectar, inclusive, o contorno destes objetos em cenários de difícil detecção [Isard & Blake, 1998].

Sendo baseado no princípio de Markov, este algoritmo utiliza apenas o estado anterior como informação prévia. Assim, não é necessário guardar informações de estados mais antigos. Como este é um filtro bayesiano, os dados do estado anterior são combinados com a observação atual para definir o próximo estado. Devido à dificuldade e os custos de se gerar todas as hipóteses possíveis para os próximos estados, o conceito de partículas é utilizado, sendo explicado mais adiante.

De modo semelhante ao filtro de Kalman, o algoritmo CONDENSATION precisa propagar uma curva de densidade de probabilidade condicional no tempo. Contudo, diferentemente do filtro de Kalman, que trabalha apenas com curvas gaussianas, este

algoritmo não impõe restrições à forma desta distribuição. Sendo assim, esta curva pode ter múltiplos picos, sendo multi modal e não linear. Desta forma, a propagação dessa curva de densidade no tempo pode ser muito custosa (Figura 5.1), comprometendo seriamente o desempenho do previsor.

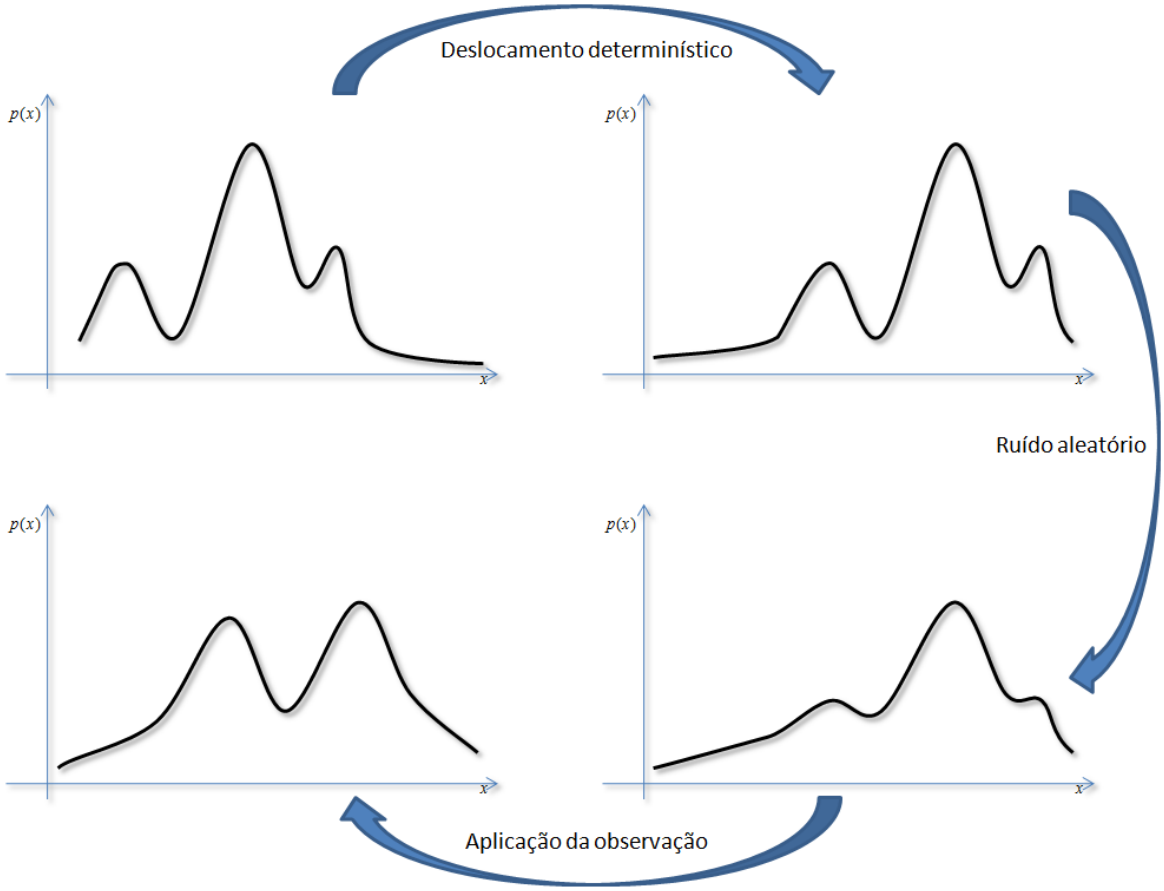

Figura 5.1. Propagação da curva de densidade de probabilidade condicional no tempo. Com o algoritmo CONDENSATION este procedimento é feito em três etapas: deslocamento determinístico utilizando o modelo dinâmico; dispersão das partículas com um ruído aleatório; aplicação da obsevação para correção das probabilidades.

Uma forma de realizar uma operação muito custosa computacionalmente, em tempo hábil, é fazer uma aproximação. Esta é a finalidade das partículas neste tipo de filtro. No algoritmo CONDENSATION, esta técnica é chamada de *factored sampling* e é descrita a seguir.

5.2 *Factored Sampling*

Com esta técnica, a função de densidade de probabilidade (fdp) passa a ser representada por um conjunto de N partículas ou amostras, representadas por $\{s_1, \ldots, s_N\}$. Inicialmente, estas partículas representam o estado *a priori* do sistema, ou seja, o estado antes de sofrer a influência da observação ou da dinâmica do sistema. Desta forma, quanto maior o número N de partículas, mais fiel será a aproximação do modelo.

Cada uma dessas partículas armazena uma possibilidade de estado do sistema. No caso do estado ser definido pela posição e velocidade do marcador, esta também é a informação armazenada nelas. Além disso, associado a cada uma, existe um valor de confiança na partícula, ou um termo que quantifica as chances desta partícula representar o estado real do sistema. Este valor é representado por π_n, com $n \in \{1, \ldots, N\}$. A Figura 5.2 demonstra esta amostragem.

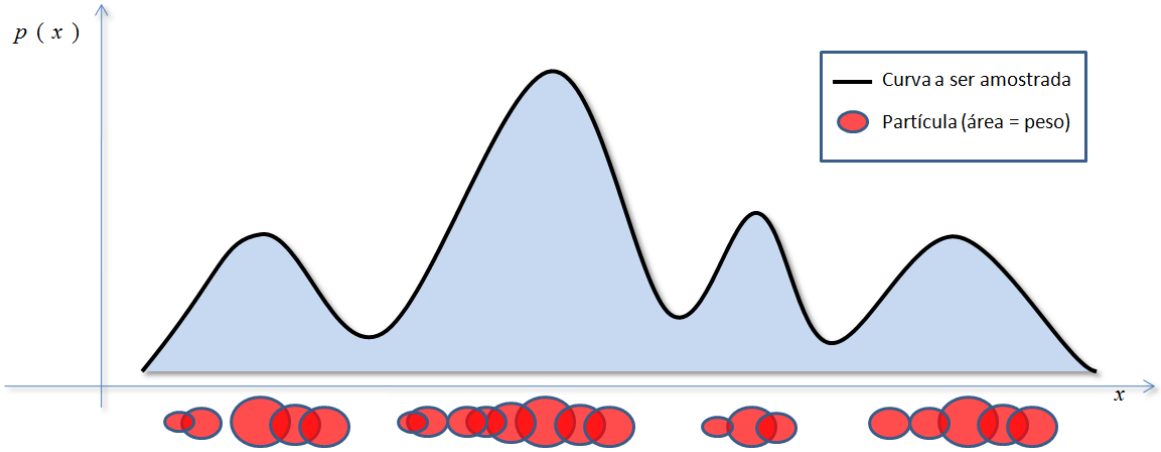

Figura 5.2. Demonstração do *factored sampling* para a amostragem de uma curva, para um problema de uma dimensão. As elipses correspondem às partículas e sua área corresponde ao peso π_n das mesmas.

Esta técnica pode ser modelada através de um problema estocástico de reconhecimento de padrões, no qual se busca o melhor valor estimado para $\hat{x}$. Assim, o estado passa a ser parametrizado por x, sendo a probabilidade do estado anterior (o estado *a priori*) representada por $p(x)$, e a observação dos dados de cada quadro da imagem por z. O estado *a posteriori* $p(x|z)$ (leia-se probabilidade do estado x dado z), representa todo o conhecimento sobre x que é obtido através da observação z. Este valor pode ser obtido aplicando-se a regra de Bayes, como mostra a Equação (5.1).

$$p(x|z) = \frac{p(z|x)p(x)}{p(z)} \tag{5.1}$$

Embora tenha-se a fórmula para calcular esta probabilidade, a dificuldade consiste no cálculo de $p(z|x)$. Por isso, o conceito de partículas é utilizado, e seu valor de confiança é uma estimativa desta probabilidade de avaliação custosa. Assim:

$$\pi_n = \frac{p(z|x = s_n)}{\sum_{i=1}^{N} p(z|x = s_i)} \tag{5.2}$$

Pode-se perceber que a Equação (5.2) gera um valor normalizado de confiança para cada partícula. Deste modo, pode-se utilizar este valor de confiança para calcular uma média ponderada das partículas (5.3), obtendo o estado *a posteriori* $\hat{x}$ que mais se aproxima do valor real, que seria obtido sem o uso de aproximações. Este valor será mais exato, quanto maior for o número de partículas, pois com isso a aproximação terá mais amostras.

$$\hat{x} = \sum_{i=1}^{N} \pi_i s_i \tag{5.3}$$

5.3 O algoritmo CONDENSATION

O algoritmo CONDENSATION, basicamente, aplica a técnica de *factored sampling* a cada instante de tempo. Dessa forma, a cada iteração ou instante de tempo k tem-se um conjunto de partículas $\{s_k^n, n = 1, \ldots, N\}$, ponderadas pelos pesos π_k^n que quantificam a confiança na partícula e representam uma aproximação para a probabilidade condicional $p(z_k|x_k)$.

No início da execução, as partículas são espalhadas de forma aleatória por todo o espaço de busca. No problema de rastreamento de marcadores, este espaço é determinado pelos limites da imagem, definidos por sua resolução. Como o estado do sistema é composto pela posição do marcador (x e y) e as velocidades nessas dimensões (velocidade em x e velocidade em y), então, cada partícula recebe inicialmente os valores de posição de forma aleatória, através de uma distribuição uniforme, e as velocidades são definidas como zero.

Ainda, resta definir os valores iniciais para o peso das partículas. Como este valor é normalizado e no início não se tem informações que permitam diferenciar nossa confiança entre cada partícula, todas recebem o mesmo valor de confiança, dado por $\frac{1}{N}$.

Tendo definido o passo inicial do algoritmo, as interações presentes nos passos seguintes serão explicadas. Resumidamente, pode-se dizer que os passos seguintes consistem em replicar estas partículas em cada instante de tempo. Embora haja uma replicação, o número N de partículas sempre é mantido constante, de forma que o

algoritmo possa, garantidamente, rodar com recursos computacionais pré-definidos, já que o custo depende diretamente do número de partículas.

A replicação das partículas pode ser dividida em três etapas: **seleção**, **predição** e **observação** (ou correção). Pode-se perceber que as duas últimas etapas estão presentes no filtro de Kalman, e até são executadas de maneira semelhante. Embora exista a semelhança, no passo de seleção e de predição existe um processo não determinístico, que gera uma aleatoriedade no algoritmo e define o seu caráter não linear.

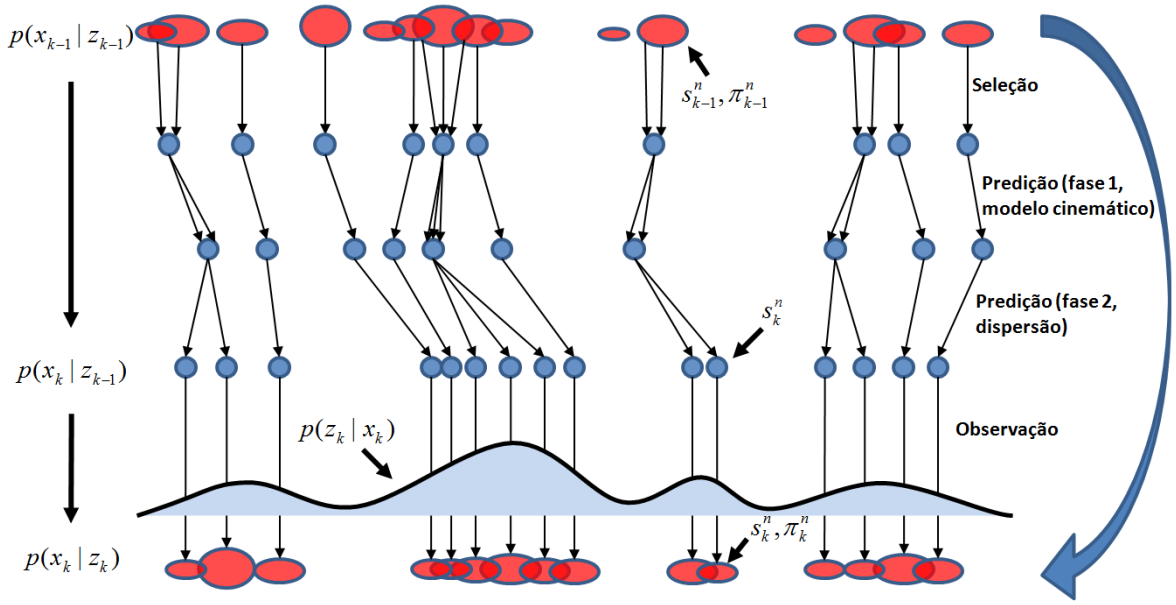

Figura 5.3. Uma iteração do algoritmo CONDENSATION, com todos os seus passos: **seleção**, **predição** (fases 1 e 2) e **observação**.

No primeiro passo, a **seleção**, as partículas a serem replicadas devem ser escolhidas. Esta seleção é feita de forma aleatória, mas baseada no valor de confiança de cada partícula. Assim, partículas com um peso maior terão uma chance maior de serem escolhidas no sorteio. Este sorteio é feito da seguinte forma: por N vezes, um número aleatório r, entre 0 e 1, é escolhido, a partícula que possuir a menor probabilidade π^n_{k-1} maior que r será escolhida para ser replicada no instante k. Com isso, partículas com uma confiança maior terão mais réplicas, enquanto partículas com baixa confiança serão penalizadas, não sendo replicadas.

Após a seleção das partículas, deve-se realizar o passo de **predição** sobre as partículas escolhidas. Este passo pode ainda ser dividido em duas fases. Na primeira, as partículas sofrem um deslocamento determinístico, provocado pela aplicação do modelo dinâmico sobre elas. Dessa forma, partículas replicadas a partir de uma mesma fonte, e que, portanto, ocupavam uma mesma posição inicial, irão se manter em uma mesma posição, mas agora deslocadas pela aplicação da cinemática.

Na segunda fase da predição, as partículas são dispersadas. Para isso, todas as partículas receberão o acréscimo de um ruído aleatório em suas posições. Assim, as partículas que foram replicadas de uma mesma fonte e se mantiveram juntas após o deslocamento, nesta fase, serão espalhadas de forma aleatória. Este passo fornece uma característica não determinística ao algoritmo, que permite gerar novas possibilidades de configuração. Estas possibilidades serão mantidas ou não, em uma próxima seleção, sempre verificando a qualidade destas previsões aleatórias, como em um processo evolutivo.

Nesse momento, todas as partículas já foram geradas, mas sem os pesos. Durante a etapa da **observação** estes pesos serão definidos. Nesta etapa, a posição medida do objeto rastreado é utilizada para calcular a confiança de cada partícula. De forma simplificada, a confiança é maior quanto mais próxima a partícula estiver do ponto real observado. A fórmula utilizada para se calcular estes valores depende de cada tipo de problema, a utilizada neste será demonstrada na seção seguinte, de instanciação do algoritmo para este caso específico (Seção 5.4).

Tendo definido os pesos das partículas em um instante de tempo, concluí-se a execução do algoritmo para este instante. No próximo instante, estes pesos serão utilizados para se calcular uma média ponderada da posição das partícula e, assim, obter uma posição estimada do objeto rastreado. A repetição destes ciclos permite o contínuo rastreamento do objeto. O Algoritmo 1 demonstra estes passos. Segundo Isard & Blake [1998] o custo computacional assintótico do algoritmo é $O(NlogN)$, sendo N o número de partículas.

5.4 Instanciação

Como ocorreu com o filtro de Kalman, o algoritmo CONDENSATION precisa de algumas definições específicas do problema de rastreamento de marcadores. Também de forma similar, este algoritmo necessita de uma definição do estado do objeto no instante k, definido por x_k. Inicialmente, a definição deste estado e a maneira de atualizá-lo foram feitas de uma forma, que apesar de utilizada em alguns trabalhos [Morais, 2005], não apresentou bons resultados no presente trabalho. Para que o conhecimento sobre a primeira forma de implementação não seja perdido, as duas formas empregadas são relatadas a seguir.

5.4.1 Primeira implementação

O estado desta implementação é igual ao estado do filtro de Kalman, definido por:

$$\mathbf{x_k} = \begin{bmatrix} x \\ y \\ velX \\ velY \end{bmatrix}$$

Pode-se perceber que o modelo dinâmico empregado é igual ao do filtro de Kalman, que prevê movimentos lineares e uniformes. Sendo o estado composto por duas variáveis de posição x e y e duas variáveis de velocidade $velX$ e $velY$. Apesar do passo de predição empregar um modelo linear uniforme, a aleatoriedade presente no algoritmo faz com que, na prática, um movimento não linear possa ser modelado utilizando uma cinemática linear.

Algoritmo 1 Iterações algoritmo CONDENSATION

1: A partir do conjunto "antigo"de partículas $\{s^n_{k-1}, \pi^n_{k-1}, n = 1, \ldots, N\}$
2: **for** $0 \leq n < N$ **do**
3: Realize a **seleção** do novo conjunto de partículas.

1. Gere um número aleatório $r \in [0, 1]$, uniformemente distribuido
2. Encontre, para $0 \leq j < N$, o menor π^j_{k-1} que satisfaça $\pi^j_{k-1} \geq r$
3. Defina $s'^n_k = s^j_{k-1}$

4: Realize a **predição** da posição das novas partículas, a partir da aplicação do modelo dinâmico.

1. Na primeira etapa aplique a velocidade à posição das partículas $s'^n_k = s'^n_k + vel^n_{k-1}$
2. Na segunda etapa aplique um ruído aleatório à posição das partículas $s^n_k = s'^n_k + w_p$

5: Utilize a **observação** z_k da posição real do marcador para gerar os valores de confiança das partículas.

$$\pi^n_k \approx p(z_k | x_k = s^n_k)$$

6: Nesta etapa já é possível **estimar** a posição prevista do marcador, que será utilizada na próxima iteração. Esta estimativa é feita através da média ponderada das partículas.

$$\hat{x} = \hat{x} + \pi^n_k s^n_k$$

7: **end for**

Como cada partícula, por si só, representa um estado do sistema, os valores que definem o estado também serão os valores armazenados em cada partícula, além do

seu peso, ou valor de confiança π_n. Com esta definição, pode-se especificar melhor a atualização dos valores de cada variável da partícula, durante a etapa de predição. Sendo s'_k o estado previsto de uma partícula durante o instante k, composto pelas variáveis previstas x', y', $velX'$ e $velY'$, o cálculo destas variáveis previstas é dado por:

$$\begin{aligned} s'_k &= (x', y', velX', velY') \\ x' &= x + velX + w_p \\ y' &= y + velY + w_p \\ velX' &= velX + w_v \\ velY' &= velY + w_v \end{aligned}$$

onde w_p é uma variável aleatória gerada através de uma distribuição uniforme, cujos limites são definidos pelas dimensões da imagem. De forma semelhante, w_v é uma variável aleatória gerada através de uma distribuição uniforme, cujos limites determinam o quanto a velocidade pode variar (em *pixels*) a cada instante. Atualizando a velocidade apenas com esse ruído aleatório, permite que a velocidade seja não linear, pois qualquer valor de velocidade - dentro dos limites estabelecidos - pode ser atribuído às partículas, mas somente as que descreverem melhor o movimento serão favorecidas durante a etapa de observação.

O valor limite para a função de distribuição uniforme, que determina os valores de w_v, foi definido como 30 por este valor demonstrar uma boa relação entre a área de busca e a velocidade de atualização da velocidade. Valores mais altos permitem alterações mais bruscas de velocidade, enquanto valores mais baixos deixam o algoritmo com um retardo na atualização da velocidade. Entretanto, valores mais altos fazem com que a área de busca aumente, devido ao aumento da incerteza gerado pelo aumento do ruído (aleatoriedade da velocidade). Uma área de busca muito grande pode ser prejudicial por aumentar o custo da busca e aumentar as chances de troca de semântica entre os marcadores.

5.4.2 Segunda implementação

Esta segunda forma de implementação é sugerida em Maskell & Gordon [2001], Moghaddam & Pentland [1995], Isard & Blake [2011] e Nicola Martorana, Iacopo Masi and Marco Meoni [2011]. O estado desta implementação é mais simples que o anterior, pois considera apenas as posições e não mais as velocidades:

$$\mathbf{x_k} = \begin{bmatrix} x \\ y \end{bmatrix}$$

Por conter apenas coordenadas de posições, um modelo dinâmico não é empregado nesta implementação. Assim, a previsão da nova posição fica a cargo apenas da análise da confiança das partículas. Desta forma, as previsões são completamente não determinísticas, compreendendo toda a não linearidade dos movimentos, já que somente um ruído aleatório define as novas posições das partículas.

Sendo s'_k o estado previsto de uma partícula durante o instante k, composto pelas variáveis previstas x' e y', o cálculo destas variáveis é dado por:

$$\begin{aligned} s'_k &= (x', y') \\ x' &= x + w_p \\ y' &= y + w_p \end{aligned}$$

onde w_p é uma variável aleatória gerada através de uma distribuição uniforme, cujos limites são definidos pelas dimensões da imagem (em pixels). Apesar de uma dinâmica de movimento não ser definida, somente as partículas que descreverem melhor o movimento serão favorecidas durante a etapa de observação, e isto fará com que elas sejam mais replicadas no próximo passo, deslocando as partículas para a vizinhança da posição atual e definindo uma nova posição prevista.

5.4.3 Implementação em comum

Outro componente do algoritmo dependente do objeto a ser rastreado é a equação do cálculo dos pesos das partículas. Segundo Isard & Blake [1998], a equação que melhor aproxima o valor estimado $p(z|x)$, neste caso de imagens de duas dimensões, é a equação de Poisson para duas dimensões, definida pela Equação (5.4).

$$p(z|x) \propto exp(-\frac{1}{2\sigma^2}(z - s_n)^2) \tag{5.4}$$

onde σ^2 representa a variância das partículas e $z - s_n$ quantifica a diferença entre a posição observada e a posição da partícula.

Observando a Equação (5.4), pode-se perceber que ela é composta de uma função exponencial que recebe como entrada um valor negativo. Devido a isto, suas saídas estão compreendidas entre 0 e 1 (sendo 1 se a entrada for 0), o que está de acordo com a definição das probabilidades, que também devem estar compreendidas neste intervalo.

Outra característica da fórmula é que quando a distância entre a partícula e a posição observada aumenta, a confiabilidade da partícula diminui, como também era esperado.

5.5 Considerações

Tendo definido os componentes do algoritmo, notam-se algumas algumas características também verificadas durante os testes.

Intervalos grandes para os limites das funções de distribuição, tanto para a posição quanto para a velocidade (todas as dimensões do estado), fazem com que as partículas se dispersem mais, deixando a incerteza maior e, consequentemente, aumentando a área de busca. Poder-se-ia deixar o intervalo de variação para a posição como o tamanho de toda a imagem, mas isto implicaria na possibilidade de uma partícula poder se movimentar de um extremo ao outro do quadro. Como essa possibilidade passa a ser contemplada, o ruído, ou a gama de possibilidades gerada no passo aleatório do algoritmo, faz com que as partículas fiquem muito dispersas. Por isso, o valor limite desta variação foi definido como uma fração de cada dimensão da imagem.

Executando o algoritmo, notou-se que é melhor gerar intervalos mais restritos e mais condizentes com o modelo do sistema. Desta forma, as partículas de concentram (ou condensam) mais, diminuindo assim o desvio padrão de suas posições. Este desvio padrão é utilizado para calcular a área de busca, restringindo esta área e evitando a captura de outros marcadores dentro deste espaço e também diminuindo o custo da busca.

Comparando com os outros métodos, percebeu-se que no filtro de Kalman é possível variar os valores da matriz Q. De forma que essa matriz tenha erros maiores, caso a velocidade seja maior, mas isto deve ser feito de forma independente da implementação do filtro convencional. Este aumento dos valores de Q aumentaria a incerteza e geraria um aumento da área de busca. No algoritmo CONDENSATION isto já acontece, pois caso a velocidade seja maior, o marcador irá se afastar mais rapidamente das partículas, elas ficarão mais distantes do marcador e terão uma confiabilidade menor, aumentando o desvio padrão e fazendo com que a área de busca aumente.

O capítulo seguinte descreve o terceiro método utilizado neste trabalho, o filtro Alfa-Beta.

Capítulo 6

Filtro Alfa-Beta

Neste capítulo, descreve-se o funcionamento do filtro Alfa-Beta, o primeiro método utilizado no sistema *OpenMoCap*. Inicialmente, suas principais características são relacionadas. Logo após, na segunda seção, detalham-se suas definições matemáticas. Finalmente, na última seção, relacionam-se as modificações de instanciação necessárias para a aplicação do filtro ao problema de rastreamento de marcadores deste trabalho.

6.1 Descrição do método

O filtro Alfa-Beta é um estimador linear de estados [Yoo & Kim, 2003]. Diferentemente dos dois estimadores apresentados anteriormente, este não é citado na bibliografia pesquisada sobre sistemas de captura de movimento. Apesar de ser um filtro criado na década de 50 e bastante conhecido, ele é mais empregado em outras áreas, como em sistemas de radares ou de rastreamento de alvos [Navarro, 1977; Lawton et al., 1998].

A característica que mais diferencia este filtro dos anteriores é sua simplicidade. Ao contrário do filtro de Kalman e do CONDENSATION, este algoritmo pode ser implementado com poucas linhas de código e também possui uma complexidade computacional bem menor que os outros dois. Devido a isto, espera-se um tempo de execução diferenciado deste algoritmo, sendo algumas vezes menor que o tempo de execução dos algoritmos mais complexos.

Apesar de sua simplicidade, este filtro possui semelhanças com o filtro de Kalman. Além de ambos serem filtros lineares, eles também podem ser separados em dois estágios: predição e correção. Ambos utilizam um valor de ganho para empregar a sua correção, sendo que no filtro de Kalman este ganho é variável e no Alfa-Beta é estático. Devido às semelhanças, pode-se inclusive entender o Alfa-Beta como uma simplifica-

ção do filtro de Kalman, sendo o Kalman um filtro ótimo (no cálculo do menor erro possível, dada a modelagem de um sistema) e o Alfa-Beta sub-ótimo.

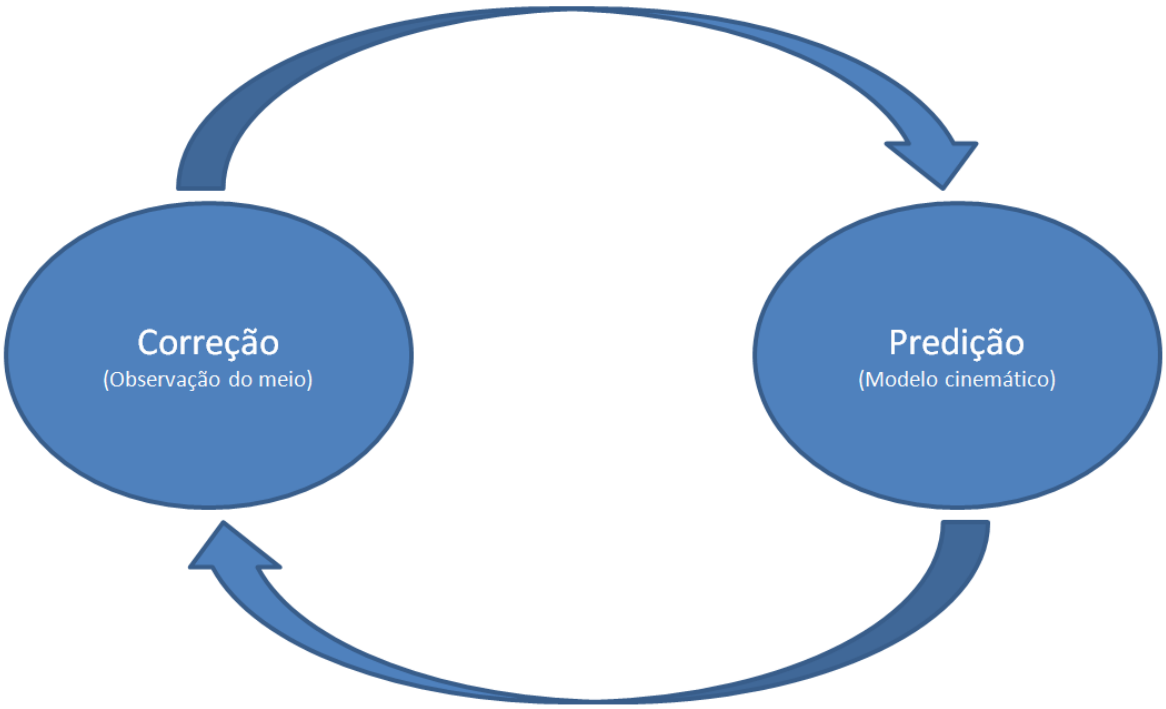

Figura 6.1. Assim como o ciclo do filtro de Kalman, o Alfa-Beta possui um ciclo de duas fases: Correção, quando a previsão anterior é corrigida utilizando a observação; Previsão, uma nova previsão é gerada utilizando o modelo dinâmico. Com algumas diferenças entre os métodos, dentre elas, os valores de ganho estáticos do filtro Alfa-Beta.

Durante a fase de correção, o erro entre a posição prevista e a posição real é calculado. Este erro é obtido subtraindo-se a posição medida da posição prevista, mas ele não é empregado diretamente na correção da posição. Antes de somar este erro à posição e à velocidade previstas, ele sofre uma atenuação (ou ganho) definida pelos parâmetros α e β, que dão nome ao filtro. Somente após o erro sofrer a interferência destes parâmetros, é que ele é acrescido à posição prevista para se obter a posição corrigida.

Na fase de predição, já se possui a posição e velocidade corrigidas, e com elas pode-se realizar a nova previsão. Para prever a nova posição do marcador, basta utilizar a posição corrigida e empregar sobre ela a velocidade corrigida, considerando a diferença de tempo Δt entre os intervalos. Desta forma, tem-se uma posição prevista do marcador para o próximo instante de tempo.

6.2 Definições matemáticas

Ainda considerando a divisão do filtro em duas fases, **correção** e **predição**, agora serão definidas as equações empregadas em cada uma delas. As equações que compõem a fase de **correção** são:

$$r_k = x_k^m - x_k^p \tag{6.1}$$

$$x_k^s = x_k^p + \alpha r_k \tag{6.2}$$

$$v_k^s = v_k^p + \beta \frac{r_k}{\Delta t} \tag{6.3}$$

A Equação (6.1) calcula o erro da predição r_k, a partir da subtração da posição medida x_k^m pela prevista x_k^p. A Equação (6.2) utiliza o erro calculado para corrigir a posição prevista e obter a posição corrigida x_k^s, mas antes de utilizá-lo, o parâmetro α, também conhecido como fator de ganho Alfa, é multiplicado pelo erro. Por fim, na Equação (6.3) a velocidade corrigida v_k^s é calculada, aplicando-se o parâmetro β, ou o fator de ganho Beta, sobre o erro da posição dividido pelo intervalo de tempo Δt.

As equações da segunda fase, **predição**, demonstram como o filtro utiliza a posição e a velocidade corrigidas para calcular a posição prevista. Dado que este filtro considera um modelo com movimentos retilíneos e uniformemente acelerados, basta multiplicar a velocidade pelo instante de tempo para obter a variação da posição:

$$x_{k+1}^p = x_k^s + \Delta t v_k^s \tag{6.4}$$

$$v_{k+1}^p = v_k^s \tag{6.5}$$

A Equação (6.4) calcula a variação da posição a partir da velocidade corrigida multiplicada por Δt. Isto retorna a variação da posição, que é somada à posição corrigida para se obter a posição prevista x_{k+1}^p do instante seguinte $(k+1)$. A Equação (6.5) simplesmente demonstra que a velocidade prevista v_{k+1}^p é igual à velocidade corrigida do instante anterior.

Considerando as equações envolvidas na execução do algoritmo do filtro Alfa-Beta, pode-se calcular o seu custo computacional. O número de operações depende do tamanho do vetor de estado x, além disso, para cada operação em uma entrada desse vetor, também temos uma operação executada em v. Desta forma, o custo

computacional deste algoritmo pode ser definido como $O(2N)$, sendo N o número de dimensões do vetor x.

6.3 Instanciação

Para a utilização do filtro, é necessário definir quais os valores presentes nos vetores utilizados. No caso do Alfa-Beta estes valores são bem simples: o vetor de posição no instante k, que contém as coordenadas x e y da posição:

$$\mathbf{x_k} = \begin{bmatrix} x \\ y \end{bmatrix}$$

E o vetor v_k, que contêm as velocidades em cada uma das coordenadas:

$$\mathbf{v_k} = \begin{bmatrix} velX \\ velY \end{bmatrix}$$

Além da especificação destes vetores, deve-se definir os valores dos parâmetros α e β. Estes valores são definidos empiricamente e são dependentes do problema e por isso serão especificados na Seção 7 de Experimentos. A relação destes valores com o rastreamento dos marcadores, pode ser definida da seguinte forma: quanto maior o valor dos parâmetros, mais rápida é a resposta do filtro à variações de trajetória ou velocidade, e mais lenta é a resposta para valores mais baixos. Entretanto, valores altos são mais suscetíveis a ruído e mudanças na trajetória ou velocidade, que não irão se manter nos próximos instantes e que podem trazer uma instabilidade à previsão, fazendo o rastreador se perder.

6.4 Considerações

Como foi descrito, este filtro trabalha apenas com um modelo linear de movimento. Embora os movimentos dos marcadores possam ser não lineares, assim como ocorre com o filtro de Kalman, espera-se que este filtro possa prever estes movimentos. Esta expectativa se baseia no fato do instante de tempo ser muito pequeno e que, portanto, pode-se considerar um pequeno movimento retilíneo dentro deste espaço de tempo, sem grandes prejuízos para a previsão.

No próximo capítulo, são expostos os experimentos realizados com cada um dos métodos descritos até aqui.

Capítulo 7

Resultados Experimentais

Neste capítulo são apresentados os experimentos realizados para avaliar os métodos de rastreamento implementados. Inicialmente, descreve-se o hardware utilizado. Logo após, os valores dos parâmetros configuráveis dos algoritmos são definidos. Por fim, relata-se como os experimentos foram feitos, seus resultados e comentários relacionados.

7.1 Hardware

Como foram realizados testes de carga de processamento, é importante relacionar os recursos computacionais utilizados. O principal hardware utilizado foi o próprio computador, cuja configuração é exibida na Tabela 7.1.

Tabela 7.1. Configuração do computador utilizado nos experimentos.

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

7.2 Valores dos Parâmetros

Inicialmente, é necessário definir os valores dados aos parâmetros dos métodos. Estes parâmetros e configurações foram escolhidos por apresentarem melhores resultados, ou por apresentarem uma boa relação entre tempo de execução e qualidade do rastreamento. Para o filtro Alfa-Beta, o parâmetro α recebeu o valor de 1,0 e o β de 0,8.

O algoritmo CONDENSATION foi utilizado com 100 partículas. O número de partículas é diretamente proporcional ao tempo de execução do método, e este valor faz com que o método tenha um tempo de execução condizente com os outros métodos, sem prejudicar muito a precisão das previsões. O vetor de estado armazena apenas as variáveis de posição x e y e não é empregado um modelo dinâmico, deixando o método completamente probabilístico e não linear. Alguns resultados utilizando o vetor de estados com posições e velocidades, mais o modelo dinâmico linear uniforme, são apresentados e comparados com a primeira configuração.

No filtro de Kalman, é importante lembrar que a matriz Q_k (que contem os erros da velocidade) é atualizada de forma que os erros são maiores caso a velocidade seja maior [Welch & Bishop, 1995], de forma que a área de busca aumente no caso de velocidades maiores. O valor inicial para $\sigma^2_{Q_{velx}}$ e $\sigma^2_{Q_{vely}}$ foi de 0.2 *pixels*. O valor inicial para $\sigma^2_{R_x}$ e $\sigma^2_{R_y}$ foi de 0.3 *pixels*. Estes valores geraram uma área de busca com uma dimensão não muito grande, para não causar troca de marcadores, e nem muito pequena, para não haver perda do marcador. A matriz P_k foi inicializada considerando-se que tem-se certeza da posição inicial do marcador, assim $\sigma^2_{P_x} = \sigma^2_{P_y} = 0$, já para as velocidades, $\sigma^2_{P_{velx}} = \sigma^2_{P_{vely}} = 100$, pois se tem uma grande incerteza inicial quanto a seus valores. O vetor de estado armazena as variáveis de posição x e y e suas respectivas velocidades.

7.3 Vídeos Empregados

Os testes foram feitos utilizando vídeos sintéticos. Estes vídeos são compostos por um ou mais círculos brancos que representam os marcadores. O uso de vídeos sintéticos agiliza o processo de criação dos vídeos e permite um maior controle sobre a trajetória e a obtenção dos valores ideais da perseguição. Como o resultado de vídeos reais, após passar pelo detector de POIs, também se resume em coordenadas passadas para o rastreador, o uso destes vídeos não trás nenhum prejuízo em relação aos reais.

Os vídeos foram gerados utilizando softwares de animação e efeitos especiais, como o Adobe Flash e o Adobe After Effects [Adobe, 2011]. Foram gerados vídeos

de 30 quadros por segundo, com resolução de 640 por 480 *pixels*. Como cada câmera possui sua própria instância de rastreador, só é necessário realizar os testes com apenas uma câmera simulada através da vídeo câmera implementada no sistema *OpenMoCap* para este trabalho.

Alguns exemplos dos vídeos durante um rastreamento são exibidos nas figuras a seguir. A Figura 7.1 exibe um quadro de um vídeo empregado no teste de robustez do rastreamento, especificamente, no teste de movimento rápido em zigue zague. A Figura 7.2 exemplifica um vídeo de uma situação de oclusão do marcador em uma trajetória retilínea e a Figura 7.3 exibe o quadro de um vídeo utilizado no teste de troca de semântica do marcador.

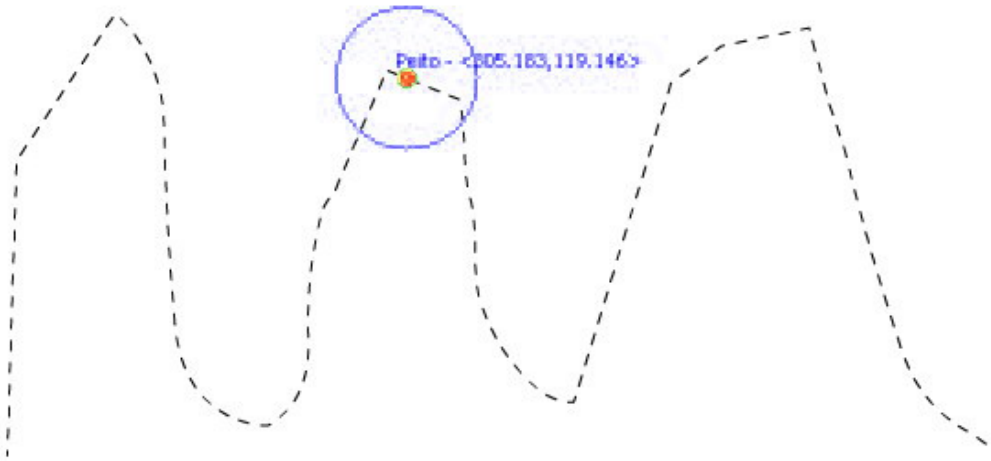

Figura 7.1. Vídeo para o teste de robustez do rastreamento, movimento rápido em zigue zague. A linha tracejada representa a trajetória do marcador, o ponto vermelho representa a posição estimada (sobre o marcador) e o círculo azul representa a área de busca.

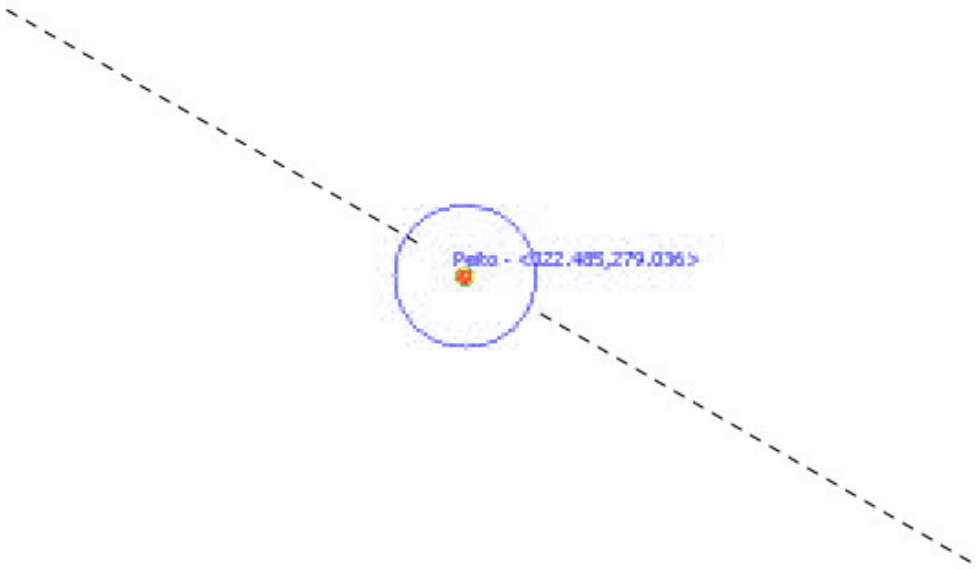

Figura 7.2. Vídeo para o teste de oclusão do marcador. Neste momento ele está passando pela região de oclusão, então somente a posição prevista é exibida.

7.4 Experimentos e Resultados

Foram realizados experimentos para medir a robustez do rastreamento e o tempo de processamento para cada um dos métodos. Para medir a robustez, foram gerados vídeos que simulam situações reais propensas a falhas. Estas situações foram dividas em três categorias: movimento não linear, oclusão e troca de semântica dos marcadores (troca dos nomes de dois marcadores). Em todas as categorias foi registrado o número de vídeos em que houve perda do marcador durante o rastreamento. Além disso, nos testes de movimentos não lineares, nos casos em que não houve perda, foi computada a distância do ponto previsto para o ponto real.

7.4.1 Movimento Não Linear

Nos testes de movimento não linear, um marcador realizava "ziguezagues"diferentes em várias direções. Estes movimentos foram separados em rápidos, lentos e com aceleração. O movimentos rápidos e lentos possuem velocidade constante, mas são não lineares

Figura 7.3. Vídeo para o teste de troca de semântica dos marcadores. O ponto preto está com a semântica Peito (encoberto pelo ponto previsto, vermelho) e o amarelo com a semântica NULL.

quanto à direção (não linearidade na trajetória). Já os movimentos com aceleração têm velocidade variável (não linearidade na velocidade), mas podem apresentar trajetórias retilíneas. Foram feitos vinte vídeos de cada situação e a Tabela 7.2 mostra em quais vídeos houve perda do marcador, por cada um dos métodos, para os movimentos lentos:

Embora se perceba que o filtro de Kalman e o algoritmo CONDENSATION tiveram um desempenho excelente, com nenhuma perda contra três perdas do Alfa-Beta, isto só foi alcançado com a modificação dos métodos após a fase de testes. Inicialmente, a área de busca do filtro de Kalman era fixa, o que gerou um desempenho para o filtro de Kalman bem semelhante ao Alfa-Beta, como pode ser demonstrado na Tabela 7.3.

Já a implementação do algoritmo CONDENSATION no qual, inicialmente, as velocidades compunham seu vetor de estado, o desempenho era bem inferior aos outros métodos. Na tentativa de melhorar seu desempenho, o limite da distribuição uniforme das velocidades, que inicialmente era de 30 *pixels* foi aumentado para 100, o que gerava

Tabela 7.2. Movimentos não lineares lentos, perda do marcador por vídeo e método.

	Alfa-Beta	*Kalman*	*CONDENSATION*
ZigueZagueLento 1			
ZigueZagueLento 2			
ZigueZagueLento 3			
ZigueZagueLento 4			
ZigueZagueLento 5	PERDA		
ZigueZagueLento 6			
ZigueZagueLento 7			
ZigueZagueLento 8	PERDA		
ZigueZagueLento 9			
ZigueZagueLento 10			
ZigueZagueLento 11			
ZigueZagueLento 12			
ZigueZagueLento 13			
ZigueZagueLento 14			
ZigueZagueLento 15			
ZigueZagueLento 16			
ZigueZagueLento 17			
ZigueZagueLento 18	PERDA		
ZigueZagueLento 19			
ZigueZagueLento 20			
Total	3	0	0

um prejudicial aumento da área de busca, mas melhorava o rastreamento. Um aumento do número de partículas não trouxe melhorias relevantes e prejudicava imensamente o tempo de execução do algoritmo. Portanto, são comparadas as implementações com limites de 30 e 100 *pixels*, ambas com 100 partículas. Estes resultados, comparados com a nova implementação do algoritmo CONDENSATION, são mostrados na Tabela 7.4.

Como se pode observar pelos resultados, houve uma melhora muito grande da implementação inicial do algoritmo CONDENSATION para a final. A primeira forma de implementação sofreu 16 perdas do marcador, contra nenhuma da implementação final. Além de um menor número de perdas, a forma final apresenta uma área de busca tão pequena quanto ou menor que a primeira forma. Ainda que a área seja menor, esta forma de implementação também apresenta menos perdas que a segunda forma, cuja área de busca é muito maior.

Como nos testes com movimentos lentos, nos testes com movimentos não lineares rápidos, as implementações do filtro de Kalman com área fixa e do algoritmo

Tabela 7.3. Movimentos não lineares lentos, comparação entre as duas implementações do filtro de Kalman: com área de busca fixa e com área de busca variável.

	Alfa-Beta	*Kalman (fixo)*	*Kalman (variável)*
ZigueZagueLento 1			
ZigueZagueLento 2		PERDA	
ZigueZagueLento 3			
ZigueZagueLento 4			
ZigueZagueLento 5	PERDA	PERDA	
ZigueZagueLento 6			
ZigueZagueLento 7			
ZigueZagueLento 8	PERDA	PERDA	
ZigueZagueLento 9			
ZigueZagueLento 10			
ZigueZagueLento 11			
ZigueZagueLento 12			
ZigueZagueLento 13			
ZigueZagueLento 14			
ZigueZagueLento 15			
ZigueZagueLento 16			
ZigueZagueLento 17			
ZigueZagueLento 18	PERDA	PERDA	
ZigueZagueLento 19			
ZigueZagueLento 20			
Total	3	4	0

CONDENSATION com as velocidades em seu vetor de estado, se mostraram sempre aquém de suas novas implementações. Portanto, essas implementações não serão mais consideradas nos testes. A partir deste ponto, serão somente comparadas as melhores implementações de cada método, de forma que as comparações seguintes serão entre o Alfa-Beta, o filtro de Kalman com área de busca variável e o algoritmo CONDENSATION com o vetor de estado composto apenas pelas coordenadas do marcador.

A Tabela 7.5 exibe os resultados obtidos nos vídeos com movimentos não lineares rápidos.

Neste teste, nota-se que o Alfa-Beta aumenta o número de erros, passando de 3 para 9. O algoritmo CONDENSATION começa a apresentar erros, contabilizando 4 vídeos com perda do marcador. O filtro de Kalman continua a apresentar um ótimo resultado, não contabilizando nenhuma perda do marcador.

Na Tabela 7.6, são exibidos os resultados obtidos nos vídeos de movimentos com aceleração, que podem ser em ziguezagues ou retilíneos.

Tabela 7.4. Movimentos não lineares lentos, comparação entre a nova implementação do algoritmo CONDENSATION, que possui apenas as coordenadas em seu vetor de estado e as implementações com as velocidades em seu vetor de estado, executados com limites para as velocidades de 30 e 100 *pixels*.

	COND. (30)	*COND.(100)*	*COND. (novo)*
ZZL 1	PERDA		
ZZL 2	PERDA		
ZZL 3	PERDA		
ZZL 4	PERDA	PERDA	
ZZL 5	PERDA		
ZZL 6			
ZZL 7			
ZZL 8	PERDA		
ZZL 9	PERDA		
ZZL 10			
ZZL 11			
ZZL 12	PERDA		
ZZL 13	PERDA		
ZZL 14	PERDA		
ZZL 15	PERDA		
ZZL 16	PERDA		
ZZL 17	PERDA	PERDA	
ZZL 18	PERDA	PERDA	
ZZL 19	PERDA		
ZZL 20	PERDA		
Total	16	3	0

Os resultados obtidos com aceleração, apresentaram um número de erros intermediário entre os movimentos lentos e rápidos, com exceção do filtro de Kalman. Este foi o único experimento em que o filtro de Kalman apresentou uma perda do marcador, que ocorreu devido a uma forte aceleração em uma curva. O erro do filtro de Kalman e a maioria dos erros do Alfa-Beta ocorreram em curvas da trajetória, principalmente curvas muito acentuadas e com aceleração brusca. O algoritmo CONDENSATION apresentou melhores resultados nas curvas, mas se perdeu com acelerações muito bruscas, mesmo em trechos retilíneos. Vídeos com estas situações podem ser vistos em [de Queiroz, 2010], na pasta Aceleração.

Tendo apresentado, em números absolutos, os vídeos em que houve perda do marcador para cada método, agora é apresentada uma análise estatística dos vídeos em que não houve perda. A análise foi feita separadamente para os vídeos com movimentos lentos, rápidos e com aceleração, para que se possa comparar as consequências de

Tabela 7.5. Movimentos não lineares rápidos, perda do marcador por vídeo e método.

	Alfa-Beta	*Kalman*	*CONDENSATION*
ZigueZagueRapido 1	PERDA		
ZigueZagueRapido 2			
ZigueZagueRapido 3	PERDA		
ZigueZagueRapido 4	PERDA		
ZigueZagueRapido 5			
ZigueZagueRapido 6	PERDA		
ZigueZagueRapido 7			
ZigueZagueRapido 8			
ZigueZagueRapido 9			
ZigueZagueRapido 10			
ZigueZagueRapido 11			
ZigueZagueRapido 12			
ZigueZagueRapido 13			
ZigueZagueRapido 14	PERDA		PERDA
ZigueZagueRapido 15	PERDA		PERDA
ZigueZagueRapido 16	PERDA		PERDA
ZigueZagueRapido 17			
ZigueZagueRapido 18			
ZigueZagueRapido 19	PERDA		
ZigueZagueRapido 20	PERDA		PERDA
Total	9	0	4

movimentos mais rápidos no erro da previsão. Nestes vídeos, para cada quadro, foi computada a diferença entre a posição prevista do marcador e a realmente medida, obtendo assim o erro da previsão. A análise deve ser feita apenas para os vídeos de acerto, porque a diferença entre a posição estimada e a real é mensurada. Considerando também os vídeos com perda, essa medida seria poluída por valores de erro enormes.

Como cada método acertou em pelo menos 11 vídeos, tendo aproximadamente 80 quadros por vídeo, foi obtido um número próximo ou superior a 1000 valores de erro por método. Esta informação é importante, pois ao se aplicar o teste de normalidade de Shapiro-Wilk [Boslaugh & Watters, 2008], o resultado foi negativo, ou seja, o valores de erro não seguem uma distribuição normal. Apesar disto, pelo Teorema Central do Limite [Ivo D. Dinov, 2008], devido ao grande número de valores, pode-se aplicar análises estatísticas de momento.

Os resultados são apresentados separadamente para os vídeos lentos, rápidos e com aceleração. Para os vídeos lentos, as médias dos erros para cada método e seus limites calculados para 99% de confiança são apresentados na Tabela 7.7. O gráfico da

Tabela 7.6. Movimentos com aceleração, perda do marcador por vídeo e método.

	Alfa-Beta	*Kalman*	*CONDENSATION*
ZigueZague 1			
ZigueZague 2	PERDA		
ZigueZague 3	PERDA		PERDA
ZigueZague 4	PERDA		PERDA
ZigueZague 5			
ZigueZague 6			
ZigueZague 7	PERDA	PERDA	
ZigueZague 8			
ZigueZague 9			
ZigueZague 10			
ZigueZague 11			
ZigueZague 12			
ZigueZague 13			
ZigueZague 14			
ZigueZague 15			
Retilíneo 16			
Retilíneo 17			
Retilíneo 18			
Retilíneo 19			
Retilíneo 20			
Total	4	1	2

Figura 7.4 também expõe estes resultados.

Tabela 7.7. Movimentos lentos. Para um intervalo de confiança de 99%, a tabela exibe para cada método o número de amostras obtido, o desvio padrão dos erros, o limite inferior, a média e o limite superior dos valores em *pixels*

Método	*Amostras*	*D. Padrão*	*L. Inf.*	*Média*	*L. Sup.*
Alfa-Beta	1445	7,95	6,67	7,21	7,75
Kalman	1700	9,62	9,71	10,31	10,91
CONDENSATION	1700	10,77	38,51	39,18	39,83

Para os vídeos rápidos, as médias dos erros para cada método e seus limites calculados para 99% de confiança, são apresentados na Tabela 7.8. O gráfico da Figura 7.5 também expõe estes resultados.

Por fim, são apresentados os resultados obtidos com os vídeos contendo aceleração, as médias dos erros para cada método e seus limites calculados para 99% de confiança, são apresentados na Tabela 7.9. O gráfico da Figura 7.6 também expõe

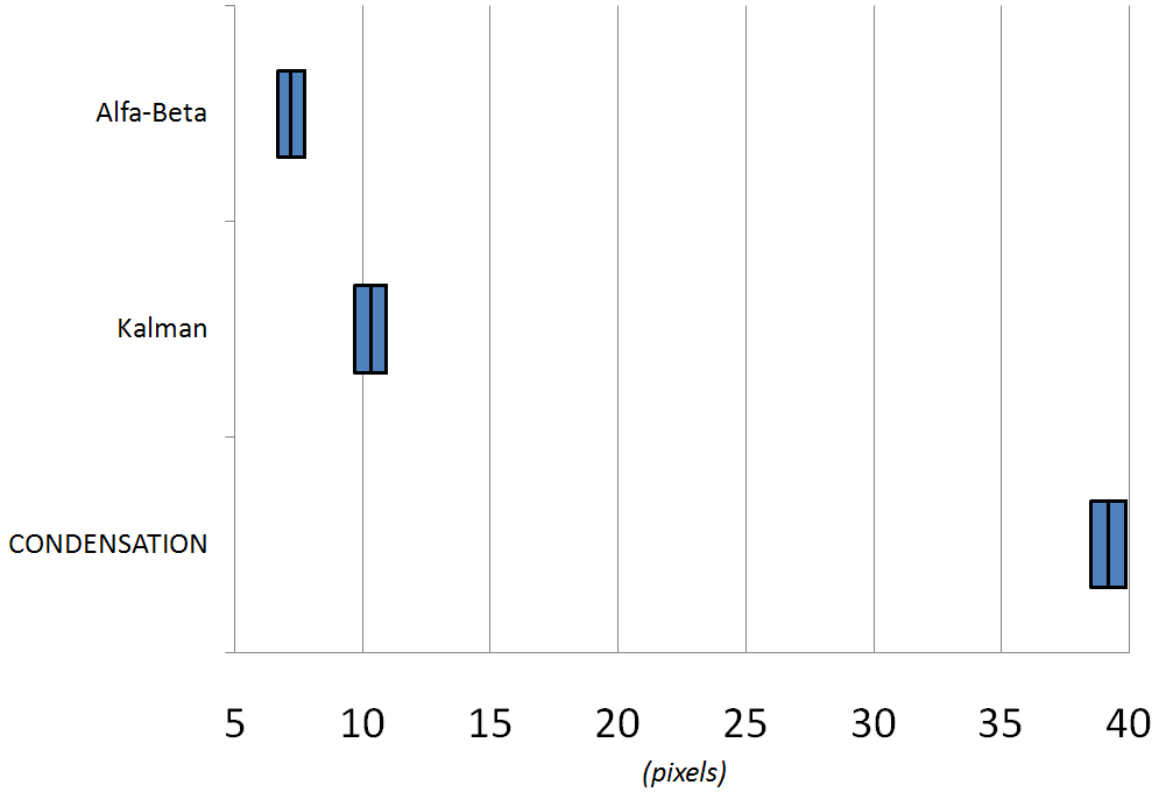

Figura 7.4. Movimentos lentos. O gráfico de erro representa as três últimas colunas da tabela, ou seja, os limites inferior, superior e a média (traço ao centro da barra) para um intervalo de confiança de 99%.

Tabela 7.8. Movimentos rápidos. Para um intervalo de confiança de 99%, a tabela exibe para cada método o número de amostras obtido, o desvio padrão dos erros, o limite inferior, a média e o limite superior dos valores em *pixels*.

Método	*Amostras*	*D. Padrão*	*L. Inf.*	*Média*	*L. Sup.*
Alfa-Beta	594	9,77	9,77	10,81	11,84
Kalman	1080	12,04	14,03	14,98	15,93
CONDENSATION	864	14,20	38,77	40,02	41,27

estes resultados.

Analisando as tabelas com os valores de erro de previsão dos métodos e os gráficos gerados com estes valores, percebe-se uma grande diferença entre os valores de cada método. Os que mais se assemelham são o filtro Alfa-Beta e o filtro de Kalman, apresentando os menores valores de erro, mas ainda assim os valores de suas médias não entram no intervalo, definido pelos limites inferior e superior, do outro. Já o algoritmo CONDENSATION, para todos os casos, apresenta valores de erro bem superiores. Com as outras implementações testadas, ele apresentava valores ainda maiores, chegando a

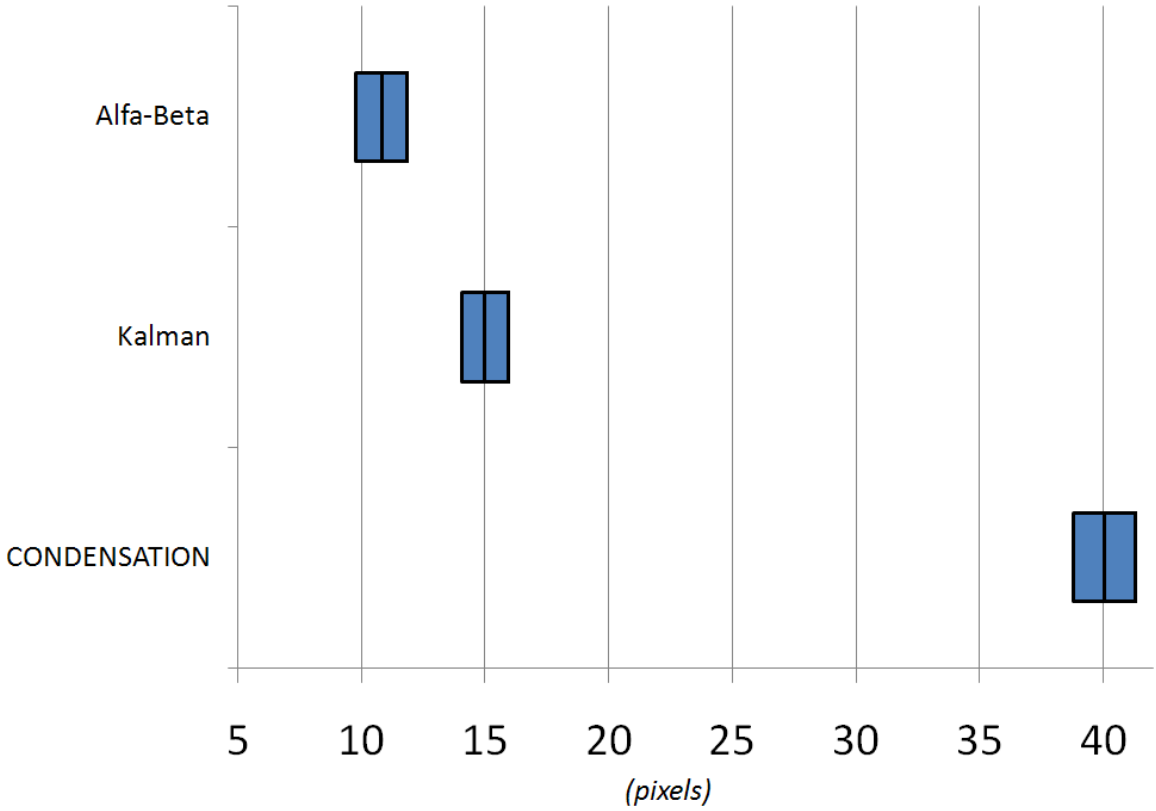

Figura 7.5. Movimentos rápidos. O gráfico de erro representa as três últimas colunas da tabela, ou seja, os limites inferior, superior e a média (traço ao centro da barra) para um intervalo de confiança de 99%.

Tabela 7.9. Movimentos com aceleração. Para um intervalo de confiança de 99%, a tabela exibe para cada método o número de amostras obtido, o desvio padrão dos erros, o limite inferior, a média e o limite superior dos valores em *pixels*.

Método	*Amostras*	*D. Padrão*	*L. Inf.*	*Média*	*L. Sup.*
Alfa-Beta	1360	4,48	2,86	3,17	3,48
Kalman	1615	7,59	5,50	5,98	6,47
CONDENSATION	1530	12,78	21,82	22,67	23,51

ser discrepantes com os resultados dos outros métodos.

Uma observação importante sobre estes valores, é que erros menores de previsão não condizem com um número de acertos maior nos vídeos. O filtro Alfa-Beta, que apresentou os menores valores de erro de previsão, foi sempre o método que mais errou nos vídeos. A partir desta informação, pode-se concluir que uma previsão muito precisa, ou muito confiante no estado anterior do sistema, não é a melhor solução. Ao que parece, uma dose de desconfiança na previsão, utilizando previsões com uma certa

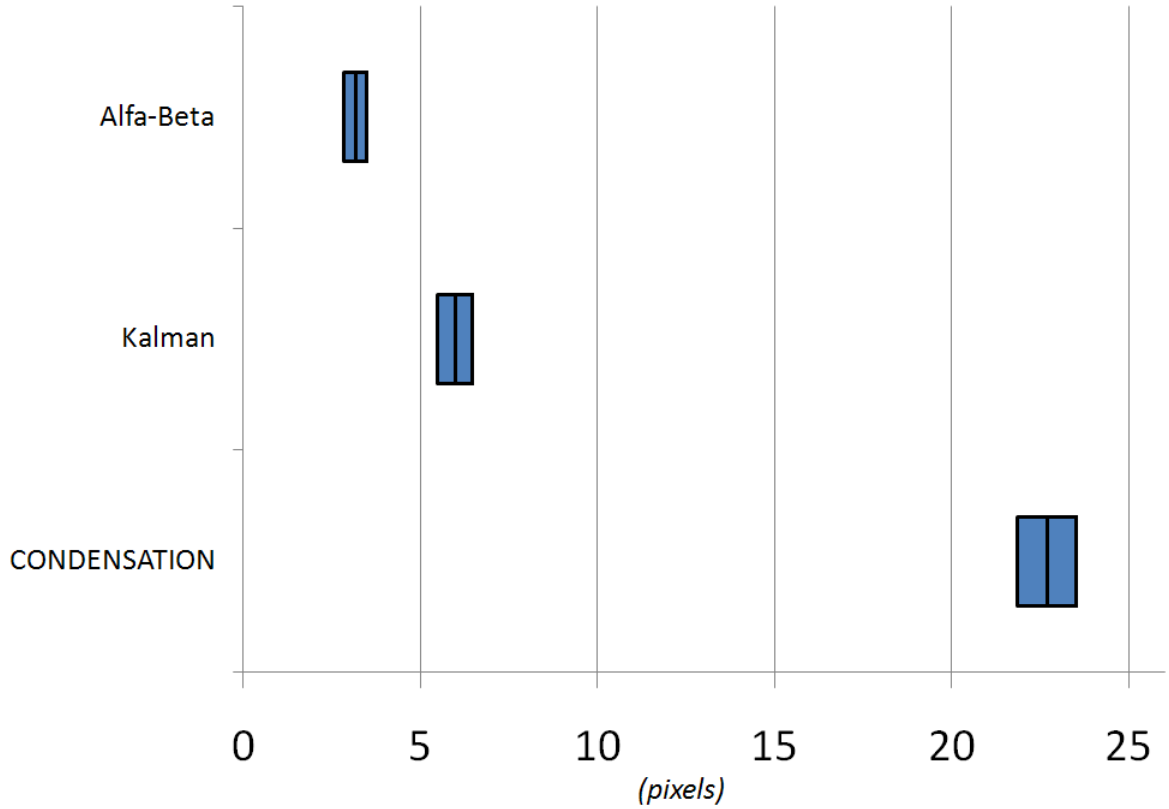

Figura 7.6. Movimentos com aceleração. O gráfico de erro representa as três últimas colunas da tabela, ou seja, os limites inferior, superior e a média (traço ao centro da barra) para um intervalo de confiança de 99%.

dose de incerteza (calculada pelas matrizes de erro no filtro de Kalman e pela confiança das partículas no algoritmo CONDENSATION) obtêm-se menos perdas do marcador, apresentando um resultado global melhor.

Outro fator que melhora os resultados do filtro de Kalman e do algoritmo CONDENSATION é a área de busca de tamanho variável. Como a área de busca nestes métodos aumenta com o aumento da incerteza (ou velocidade), em situações mais propensas a erro, como em altas velocidades, eles se saem melhor. A partir desta observação, talvez uma boa solução de rastreador, seria combinar a simplicidade do Alfa-Beta com uma área de busca variável, a princípio, baseada na velocidade. Esta ideia será listada como trabalhos futuros e pretende-se analisar a sua viabilidade.

7.4.2 Oclusão

Antes de apresentar o resultados dos testes, cabe lembrar qual seria o comportamento esperado de cada método sob uma situação de oclusão. O Alfa-Beta e o filtro de Kalman empregam um modelo dinâmico preditivo linear e uniforme, ou seja, em uma situação

de perda momentânea (ou oclusão) do marcador, eles tendem a continuar prevendo uma trajetória linear para o marcador. Nestes casos, durante a situação de perda em uma curva, por exemplo, o ponto previsto sairá tangente à curva, já em um seguimento reto, as chances de se recuperar a semântica do marcador são grandes, pois a trajetória real é condizente com o modelo previsto. Alguns vídeos com estas situações podem ser visualizados em [de Queiroz, 2010], na pasta Oclusão.

O comportamento do algoritmo CONDENSATION sob uma situação de oclusão, difere dos demais. Da maneira como foi implementado, ele não utiliza um modelo dinâmico, apenas utiliza a confiança nas partículas para deslocar o seu ponto previsto. Dessa forma, em uma situação de perda momentânea (ou oclusão) do marcador, o ponto previsto tende a ficar praticamente parado, pois não há novas leituras de posições reais, que provoquem o deslocamento da posição prevista. Apesar deste comportamento sugerir um desempenho muito inferior aos outros algoritmos, que deslocam continuamente o ponto previsto, isto não foi o constatado no teste. Isto porque, em algumas situações, permanecer parado em um meandro da trajetória, faz com que ele recupere o marcador ao passar pelo outro lado da curva. Estas situações também podem ser vistas em [de Queiroz, 2010], na pasta Oclusão.

Nos experimentos, um obstáculo de tamanho variável (em cada vídeo) foi posicionado sobre a trajetória do marcador, que podia ser linear ou não. Também foram feitos 20 vídeos de teste, sendo feita a contagem do número de perdas do marcador para cada método. A Tabela 7.10 demonstra os resultados obtidos nos testes de oclusão do marcador.

O desempenho geral dos métodos foi muito semelhante. O filtro de Kalman foi ligeiramente melhor que os outros, com 6 perdas, e o algoritmo CONDENSATION pouco pior, com 8 perdas. Apesar de o resultado absoluto ser praticamente o mesmo, todos os métodos tiveram um acerto exclusivo e uma perda exclusiva (exceto o filtro de Kalman) em pelo menos um vídeo. Isto demonstra que os métodos possuem características únicas que os beneficiam em algumas situações e os prejudicam em outras, sugerindo que uma combinação de suas qualidades pode trazer melhores resultados que seu uso individual.

Neste teste específico, percebe-se que o algoritmo CONDENSATION possui um desempenho melhor em oclusões que ocorrem em seções curvas da trajetória, mas sofre perda do marcador em trajetórias completamente retas com oclusão (vídeos 13, 14, 15 e 16). Já os outros métodos, que utilizam um modelo preditivo linear, têm um desempenho muito bom em seções retas da trajetória, com exceção do erro do filtro de Kalman no vídeo 16, que apesar de ter uma trajetória retilínea, continha um obstáculo muito grande. Como o filtro de Kalman nestes casos fica um longo período sem sua

Tabela 7.10. Oclusão do marcador, perda do marcador por vídeo e método.

	Alfa-Beta	*Kalman*	*CONDENSATION*
Oclusão 1	PERDA	PERDA	PERDA
Oclusão 2	PERDA		PERDA
Oclusão 3	PERDA	PERDA	PERDA
Oclusão 4	PERDA	PERDA	PERDA
Oclusão 5			PERDA
Oclusão 6			
Oclusão 7			
Oclusão 8			
Oclusão 9	PERDA	PERDA	
Oclusão 10	PERDA	PERDA	
Oclusão 11			
Oclusão 12			
Oclusão 13			
Oclusão 14			
Oclusão 15			PERDA
Oclusão 16		PERDA	PERDA
Oclusão 17			
Oclusão 18			PERDA
Oclusão 19	PERDA		
Oclusão 20			
Total	7	6	8

fase de correção, sua incerteza aumenta muito, causando a perda do marcador.

7.4.3 Troca de Semântica

Nos testes de troca de semântica, dois marcadores percorriam trajetórias que se interceptavam, criando situações propícias à troca. Para que se possa distinguir entre os dois marcadores, um deles foi criado com a cor amarela. Embora haja a diferença, isto não interfere nos testes, pois os métodos de rastreamento não têm acesso a esta informação de diferenciação, já que eles trabalham apenas com as coordenadas obtidas pelo detector de POIs. Nos 20 vídeos, os casos em que houve troca foram registrados, sendo mostrados na Tabela 7.11.

Novamente, percebe-se um desempenho diferenciado entre os filtros de Kalman e Alfa Beta e o algoritmo CONDENSATION. Pela tabela, verifica-se que em vários vídeos houve um acerto (7 e 19) ou um erro (1, 2, 4, 9, 16 e 20) exclusivo do algoritmo CONDENSATION, demonstrando seu caráter diferenciado. Uma situação muito interessante foi o acerto do método CONDENSATION no vídeo 19, que apresenta uma

Tabela 7.11. Troca de semântica do marcador, troca da semântica por vídeo e método.

	Alfa-Beta	*Kalman*	*CONDENSATION*
ZigueZague 1			TROCA
ZigueZague 2			TROCA
ZigueZague 3			
ZigueZague 4			TROCA
ZigueZague 5			
ZigueZague 6		TROCA	TROCA
ZigueZague 7	TROCA	TROCA	
ZigueZague 8	TROCA		TROCA
ZigueZague 9			TROCA
ZigueZague 10	TROCA	TROCA	TROCA
ZigueZague 11	TROCA	TROCA	TROCA
ZigueZague 12	TROCA	TROCA	TROCA
ZigueZague 13	TROCA	TROCA	TROCA
ZigueZague 14		TROCA	TROCA
ZigueZague 15	TROCA	TROCA	TROCA
ZigueZague 16			TROCA
ZigueZague 17	TROCA	TROCA	TROCA
ZigueZague 18		TROCA	TROCA
ZigueZague 19	TROCA	TROCA	
ZigueZague 20			TROCA
Total	9	11	16

situação de previsão muito difícil para métodos que utilizam um modelo de previsão retilíneo e uniforme, difícil até mesmo para seres humanos (com os marcadores de cores iguais). Apesar deste acerto nesta situação inesperada, também inesperadamente ele errou no vídeo 20, o qual parecia bastante simples. Estes exemplos estão expostos em [de Queiroz, 2010], na pasta Troca.

O número de erros neste teste, foi bem superior ao dos outros. Este resultado demonstra a dificuldade de se definir semânticas diferentes a dois marcadores que na verdade são idênticos. A única diferenciação entre eles é o seu movimento, que neste teste, propositalmente, também foi feito para se assemelhar ao do outro, provocando uma grande taxa de erros.

Apesar de nestes testes os marcadores serem idênticos, em situações reais isso nem sempre é verdade. Uma diferenciação entre dois marcadores, que no mundo real são iguais, surge na imagem devido às diferentes distâncias as quais cada marcador está da câmera. Devido à geometria projetiva, marcadores mais distantes da câmera terão áreas menores que marcadores mais próximos. A possibilidade de se utilizar esta

informação é discutida nas conclusões e trabalhos futuros, Capítulo 8.

7.4.4 Custo Computacional

O teste para se mensurar o custo computacional de cada algoritmo foi feito medindo-se o tempo necessário para se processar cada quadro do vídeo. Este tempo considera exclusivamente a tarefa de previsão da posição do marcador, excluindo-se tempos como o de processamento da imagem, detecção do POI, etc. Apesar de este tempo ser uma medida extremamente dependente do hardware empregado, o objetivo era fazer um comparativo entre os custos de cada método. Portanto, as comparações são válidas, desde que as mesmas configurações sejam mantidas para todos os algoritmos, como foi feito neste trabalho.

Neste experimento foram feitas trajetórias simples para os marcadores. Eles apenas oscilam lentamente de um ponto a outro repetitivamente, pois o objetivo não era testar a robustez do rastreamento, já avaliada anteriormente. Para medir o desempenho dos algoritmos com diferentes números de marcadores, foram feitos vídeos com 1, 5, 10, 15, 20, 25 e 30 marcadores. Os tempos foram mensurados durante 200 quadros e os resultados são exibidos a seguir. As Figuras 7.7 e 7.8 exibem quadros dos vídeos com 5 e 30 marcadores, utilizados nestes testes.

Nas análises de tempo de processamento, uma análise estatística de ordem se torna mais proveitosa que a de momento. Pode-se observar este fato avaliando o valor obtido para, por exemplo, o limite superior do tempo de processamento com o filtro de Kalman para 20 marcadores. Com 99% de confiança esse valor corresponde a 0,30 ms, já o máximo valor obtido nesta situação corresponde a 0,39 ms. Caso o limite de tempo para um processamento em tempo real estivesse entre 0,30 ms e 0,39 ms, podería-se acreditar que seria possível o processamento em tempo real apenas observando o limite superior, mas o pico de tempo apresentado pelo valor máximo demonstra que haveria a perda de um quadro nesta situação. Portanto, para a comparação entre os métodos, as medianas dos valores são consideradas, mas também foram analisados os valores máximos para verificar a viabilidade de execução dos algoritmos em tempo real.

O gráfico da Figura 7.9 exibe uma comparação entre o aumento dos valores da mediana de cada método. A maioria dos métodos apresenta um crescimento que pode ser representado por uma função linear com um erro mínimo. Somente o algoritmo CONDENSATION apresenta uma pequena variação no crescimento para os últimos valores. Os valores máximos, na maioria dos casos, também seguem um crescimento linear, que não se afasta muito dos valores da mediana. A exceção é o Alfa-Beta, cujos máximos nem sempre aumentaram, como pode ser observado na Figura 7.10.

Figura 7.7. Vídeo para o teste de custo computacional com 5 marcadores.

Analisando os resultados, percebe-se uma grande diferença entre os tempos de processamento dos métodos. Como era esperado, o filtro Alfa-Beta apresenta os menores tempos, chegando a ser 2 vezes mais rápido que o filtro de Kalman e 12 vezes mais rápido que o algoritmo CONDENSATION, demonstrando uma enorme diferença de custo computacional entre os métodos. Mesmo na comparação entre os métodos esperadamente mais custosos, o filtro de Kalman e o algoritmo CONDENSATION, nota-se um tempo de processamento do algoritmo CONDENSATION até 6 vezes maior que o do filtro de Kalman.

Os maiores tempos de processamento obtidos foram sempre com o algoritmo CONDENSATION, que atingiu o maior pico com 30 marcadores, obtendo o tempo de 2,812 ms. O custo computacional deste algoritmo é diretamente proporcional ao número de partículas, que foi definido como 100 nos testes. Com apenas 10 partículas, os tempos deveriam ser reduzidos a uma ordem 10 vezes menor, mas isto implicaria em uma piora significativa da qualidade do rastreamento.

Apesar do maior tempo de 2,812 ms obtido pelo algoritmo CONDENSATION pa-

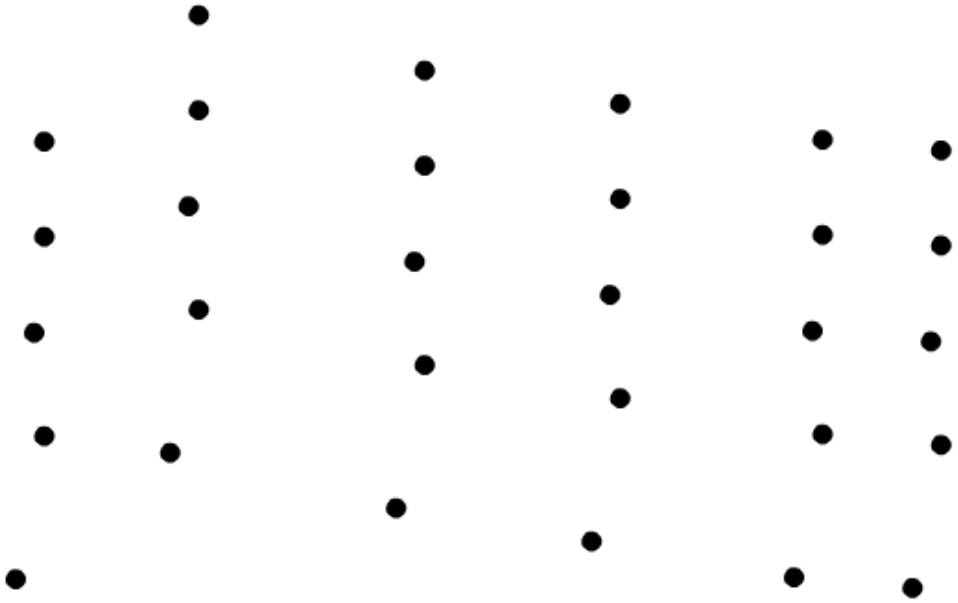

Figura 7.8. Vídeo para o teste de custo computacional com 30 marcadores.

Tabela 7.12. Alfa-Beta. Para cada número de marcadores, a tabela exibe o 5º percentil, a mediana, o 95º percentil e o máximo dentre os tempos de processamento obtidos (em ms).

Marcadores	*5º Percentil*	*Mediana*	*95º Percentil*	*Máximo*
1	0,043	0,046	0,053	0,077
5	0,064	0,067	0,084	0,104
10	0,086	0,091	0,110	0,116
15	0,109	0,113	0,126	0,218
20	0,130	0,140	0,155	0,221
25	0,136	0,164	0,184	0,211
30	0,168	0,194	0,214	0,274

recer grande em relação aos outros métodos, este valor corresponde a apenas 8,44% da janela de tempo de 33 ms disponível para o processamento. Muito embora a proporção seja pequena, outras etapas da captura de movimento também devem ser executadas neste tempo. Baseado no trabalho de [Flam, 2009], sabe-se que na média o tempo de processamento das outras etapas não passa de 20 ms, o que permite gastar até 13 ms

Tabela 7.13. Filtro de Kalman. Para cada número de marcadores, a tabela exibe o 5º percentil, a mediana, o 95º percentil e o máximo dentre os tempos de processamento obtidos (em ms).

Marcadores	*5º Percentil*	*Mediana*	*95º Percentil*	*Máximo*
1	0,067	0,077	0,100	0,126
5	0,120	0,130	0,148	0,174
10	0,172	0,175	0,195	0,268
15	0,226	0,234	0,256	0,331
20	0,293	0,296	0,314	0,386
25	0,338	0,356	0,376	0,453
30	0,409	0,418	0,445	0,503

Tabela 7.14. CONDENSATION. Para cada número de marcadores, a tabela exibe o 5º percentil, a mediana, o 95º percentil e o máximo dentre os tempos de processamento obtidos (em ms).

Marcadores	*5º Percentil*	*Mediana*	*95º Percentil*	*Máximo*
1	0,134	0,146	0,162	0,180
5	0,387	0,499	0,529	0,801
10	0,824	0,950	1,017	1,095
15	0,958	1,384	1,423	1,604
20	1,285	1,842	1,920	2,116
25	1,560	1,934	2,333	2,427
30	1,886	2,105	2,791	2,812

com o rastreamento. Logo, mesmo considerando o maior pico de tempo obtido, ainda assim seria possível um rastreamento em tempo real com uma taxa de 30 quadros por segundo.

7.4.5 Vídeos Reais

Além dos testes com vídeos sintéticos, também foram realizados testes com vídeos reais. Para isso, gravou-se vídeos nos quais um ator se movimentava utilizando uma roupa com marcadores reflexivos ao infra vermelho emitido pela câmera. O vídeos foram gravados com a resolução de 640 x 480 *pixels*, a uma taxa de 30 quadros por segundo e utilizando 11 marcadores. Um exemplo de vídeo empregado é exibido na Figura 7.11.

Os vídeos foram criados com objetivos similares aos buscados nos vídeos sintéticos. Desta forma, tentou-se criar vídeos com movimentos lentos, movimentos rápidos, oclusões e trocas de semântica, sendo cinco vídeos para cada situação. Entretanto, como são movimentos reais do ator, em alguns casos ocorrem problemas de oclusão

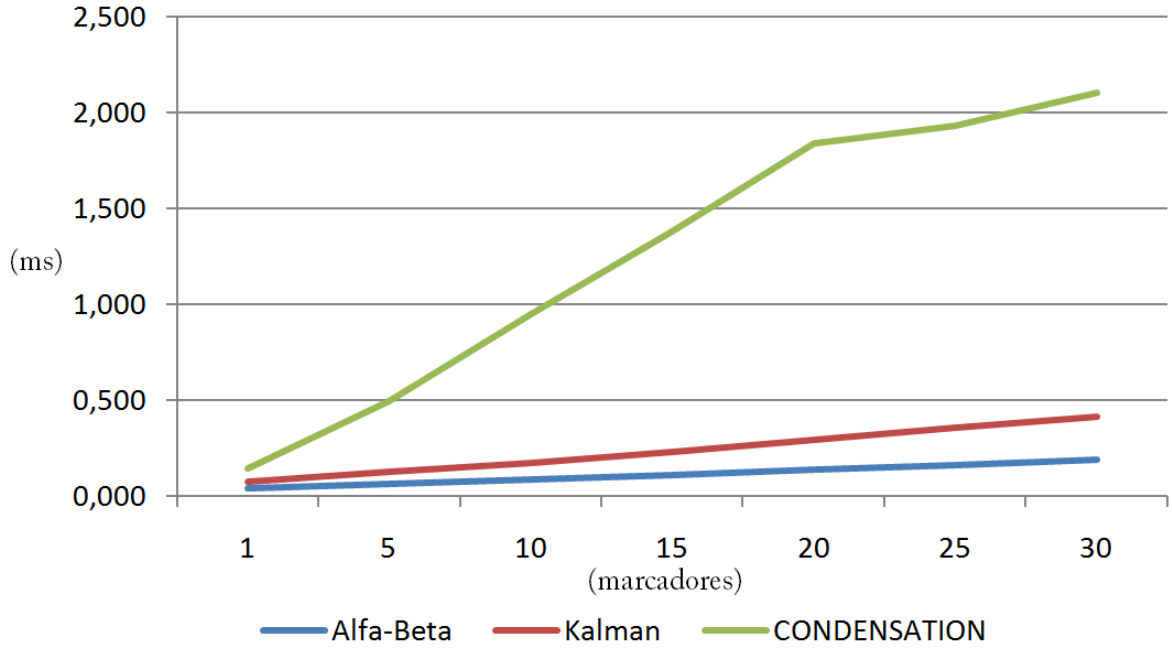

Figura 7.9. Tempos de processamento. Gráfico com os valores das medianas para os três métodos.

ou troca de semântica em vídeos nos quais se pretendia apenas avaliar a robustez do rastreamento quanto a trajetória. Estas situações foram descritas nos resultados e uma análise mais qualitativa é feita ao final da seção.

Tabela 7.15. Testes de movimentos lentos com vídeos reais. Número de perdas de marcador para cada vídeo e método, com 11 marcadores.

	Alfa-Beta	*Kalman*	*CONDENSATION*
Lento 1	1(oclusão)	1(oclusão)	1(oclusão)
Lento 2	0	0	0
Lento 3	0	0	0
Lento 4	0	0	0
Lento 5	2 (oclusão)	2 (oclusão)	2 (oclusão)
Total	3	3	3

Analisando os resultados, percebe-se um resultado semelhante de todos os métodos. Como as condições de vídeos reais não podem ser tão bem controladas como nos vídeos sintéticos, pequenas variações nos métodos, que os tornam melhores que os outros, não são tão bem percebidas nestes testes. Ainda assim, considerando o desempenho geral e observando qualitativamente os resultados, o filtro de Kalman novamente obteve um melhor desempenho que os outros dois métodos.

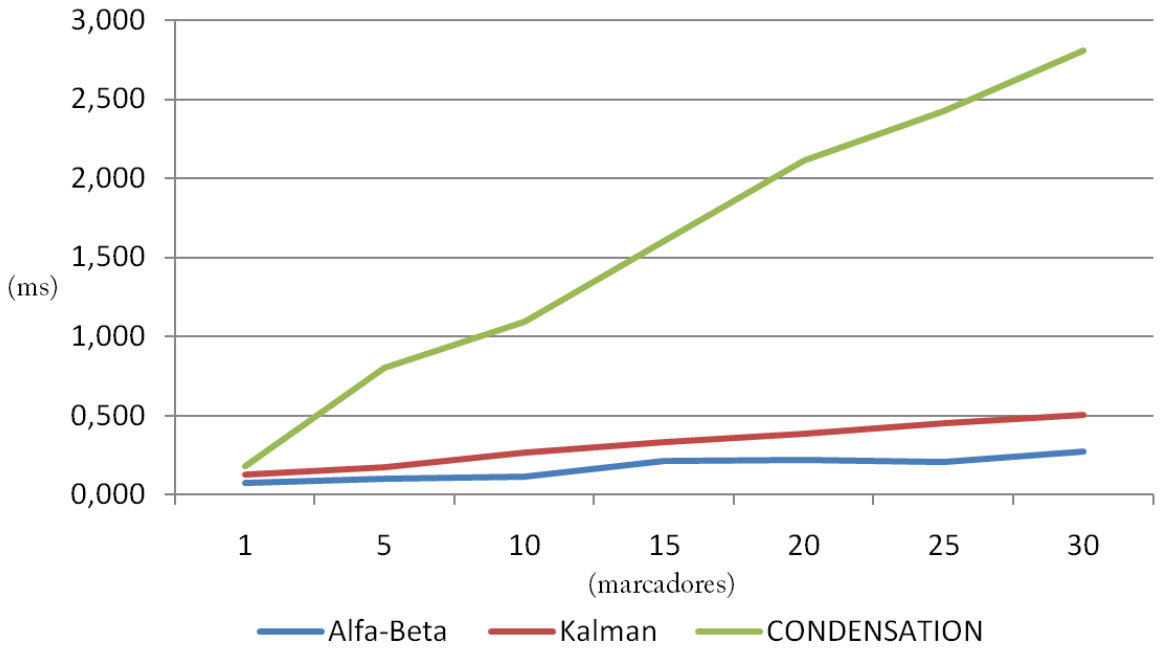

Figura 7.10. Tempos de processamento. Gráfico com os valores máximos para os três métodos.

Tabela 7.16. Testes de movimentos rápidos com vídeos reais. Número de perdas de marcador para cada vídeo e método, com 11 marcadores.

	Alfa-Beta	*Kalman*	*CONDENSATION*
Rápido 1	0)	0	0
Rápido 2	0	0	0
Rápido 3	0	0	1
Rápido 4	0	0	2
Rápido 5	0	0	0
Total	0	0	3

O filtro de Kalman empatou como o melhor com pelo menos um método nos três primeiros testes e foi melhor no teste de troca de semântica. Já o filtro Alfa-Beta foi o pior para os testes de oclusão, principalmente quando um dos marcadores era sobreposto por um objeto e mudava sua trajetória enquanto estava atrás do obstáculo. Como o algoritmo CONDENSATION tende a ficar no mesmo lugar em caso de oclusão, ele conseguia se recuperar nestas situações. Entretanto, nos testes de movimentos rápidos, devido ao retardo na atualização da posição do algoritmo CONDENSATION, ele foi prejudicado por movimentos muito rápidos, como uma intensa movimentação das mãos ao balançar os braços do ator. Exemplos dos vídeos empregados podem ser

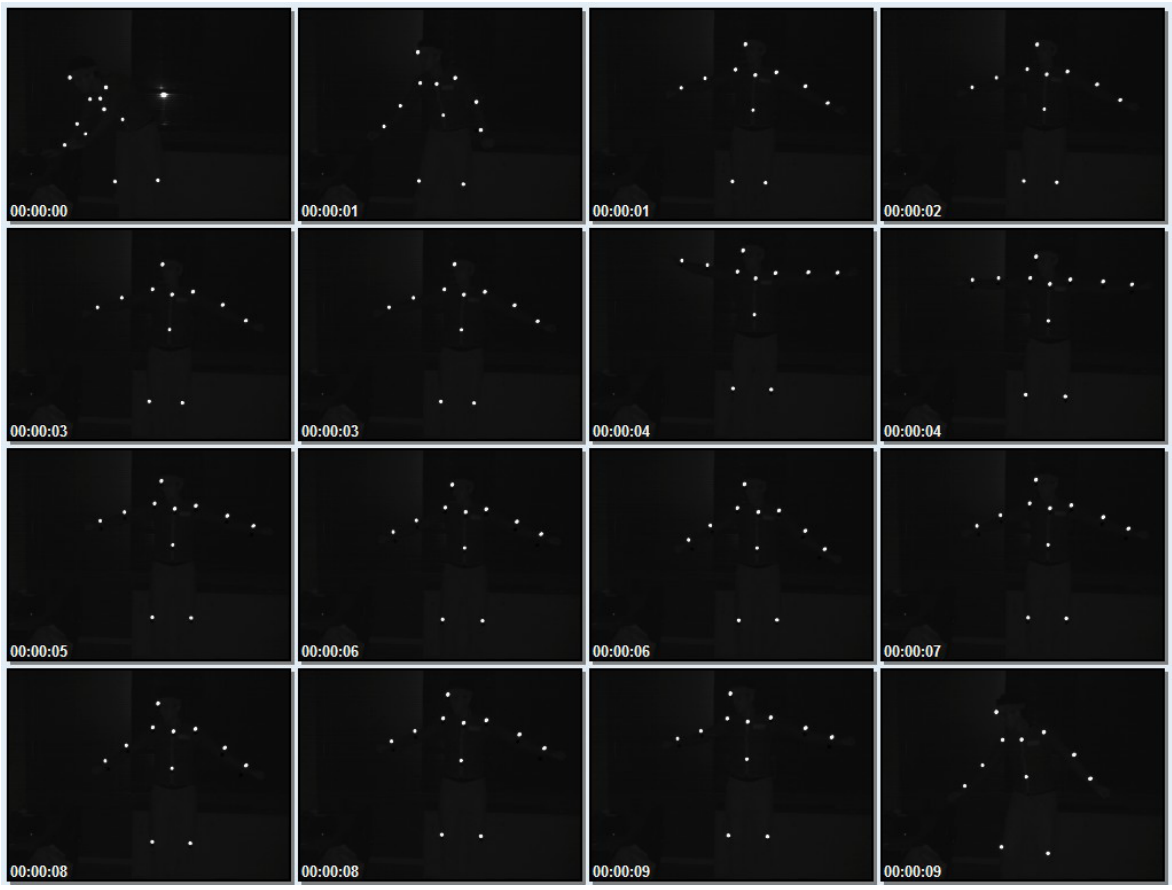

Figura 7.11. Movimentos rápidos. Exemplo de vídeo real para movimentos rápidos no qual o ator promove diversos saltos e agita os braços.

encontrados em [de Queiroz, 2010], na pasta Reais.

7.5 Considerações

Neste capítulo, o hardware utilizado foi relacionado e foram expostos os experimentos realizados com os métodos avaliados neste trabalho. Foram aplicados experimentos para se avaliar a robustez do rastreamento: analisando movimentos não lineares (quanto a trajetória e/ou velocidade), robustez à oclusão e troca de semântica dos marcadores. Além dos testes de robustez, também foram aplicados testes para mensurar a carga computacional necessária para a execução de cada algoritmo. Os testes foram realizados utilizando vídeos que simulam situações reais propensas a falhas.

Observando os resultados, percebe-se que os métodos se agrupam em duas classes, devido ao seu comportamento diferenciado frente aos testes: uma classe compreendendo o filtro de Kalman e o filtro Alfa-Beta e outra o algoritmo CONDENSATION. De maneira geral, o filtro de Kalman apresentou melhores resultados que os outros dois

Tabela 7.17. Testes de oclusão com vídeos reais. Número de perdas de marcador para cada vídeo e método, com 11 marcadores.

	Alfa-Beta	*Kalman*	*CONDENSATION*
Oclusão 1	2	2	2
Oclusão 2	1	1	1
Oclusão 3	1	2	1
Oclusão 4	5	4	4
Oclusão 5	2	2	2
Total	11	10	10

Tabela 7.18. Testes de troca de semântica com vídeos reais. Número de trocas de semântica do marcador para cada vídeo e método, com 11 marcadores.

	Alfa-Beta	*Kalman*	*CONDENSATION*
Troca 1	0	0	0
Troca 2	0	0	1
Troca 3	2	2	3
Troca 4	1	1	1
Troca 5	3	2	3
Total	6	5	8

métodos, tendo uma taxa de perdas do marcador menor na maioria dos testes e um desempenho computacional intermediário.

No próximo capítulo, é concluída a dissertação sobre este trabalho com uma discussão sobre os objetivos alcançados e possíveis trabalhos a serem realizados.

Capítulo 8

Conclusões

8.1 Objetivos Alcançados

O objetivo inicial deste trabalho era implementar na plataforma do *OpenMoCap* os métodos de rastreamento mais utilizados para a captura de movimento e confrontá-los com o método até então utilizado. Tendo o propósito de mensurar a qualidade do algoritmo até então utilizado, e escolher o rastreador que deixaria o sistema mais eficaz e robusto, aproximando-o da qualidade dos sistemas comerciais de captura de movimento.

Fazendo uma análise geral dos resultados de todos os experimentos, pode-se concluir que o objetivo foi alcançado, escolhendo o filtro de Kalman como o melhor rastreador nas situações apresentadas. Pelo fato do filtro de Kalman ser um rastreador linear, este resultado talvez não fosse o esperado observando-se apenas o problema de rastrear marcadores que se movimentam de forma não linear. A princípio, poder-se-ia esperar que o algoritmo CONDENSATION obtivesse um resultado melhor que todos os métodos lineares (desconsiderando tempo de processamento). A Tabela 8.1 resume as qualidades e limitações de cada método, percebidas durante os experimentos.

Este resultado, de alguma maneira, inesperado faz parte dos objetivos secundários alcançados com este trabalho. Pode-se atribuir este resultado ao fato de os movimentos dos marcadores ocorrerem em uma matriz discreta de *pixels* e durante um intervalo de tempo muito pequeno, o que provoca também um deslocamento muito pequeno entre cada quadro. Então, considerando a discretização do movimento e o pequeno *quantum* de deslocamento ou variação da aceleração, estes movimentos não lineares acabam sendo bem modelados por um sistema linear. Esta teoria explicaria o fato de um filtro linear apresentar um desempenho melhor, como também o fato de uma instância do algoritmo CONDENSATION sem um modelo dinâmico, ou seja, que prevê a partícula

parada no próximo quadro, também ter se saído melhor.

Além deste objetivo secundário, ao se observar os testes de cada método, muito se aprendeu sobre o comportamento destes preditores. Agora sabe-se que o filtro Alfa-Beta e o filtro de Kalman realmente pertencem a uma classe diferente do algoritmo CONDENSATION, pois as situações nas quais os primeiros se comportam melhor, são opostas às do último. Isto reforça a citação no Capítulo 6 sobre o filtro Alfa-Beta, de que ele é uma simplificação do filtro de Kalman.

Este comportamento complementar, também estimula a criação de filtros que combinem as qualidades destes métodos. Observa-se que os dois filtros lineares são muito bons em oclusões ou altas velocidades em trajetórias retilíneas, já o algoritmo CONDENSATION tem um desempenho melhor nestas situações, mas em trajetórias curvilíneas. Logo, o melhor dos filtros seria uma combinação das qualidades destas duas classes, talvez através de um chaveamento, ou a troca do método empregado, dependendo da situação presente.

Embora o algoritmo CONDENSATION tenha um desempenho melhor em certas situações, o seu custo computacional chegou a ser uma ordem de grandeza maior que o dos outros (mais de 10 vezes). Essa caraterística, poderia ser um empecilho à implementação deste algoritmo em um sistema de captura que gaste mais tempo de processamento nas outras etapas do processo, deixando uma janela de tempo menor para o rastreamento. Neste caso, a melhor escolha seria o filtro Alfa-Beta, que tem um custo computacional extremamente interessante. Além disso, o uso deste filtro poderia ficar ainda mais atrativo com a implementação de técnica para alterar sua área de busca, relacionando-a com alguma incerteza na previsão, como a velocidade.

Outra modificação, comportada pelo filtro de Kalman e o algoritmo CONDENSATION, seria a inclusão da área do marcador no estado do sistema. Fazendo algumas modificações no *OpenMoCap*, poder-se-ia obter esta área e incluí-la no vetor de estado destes métodos, juntamente com uma velocidade da sua variação. Esta modificação ajudaria em situações de troca da semântica dos marcadores, pois como eles têm uma mesma área no mundo real, e ao se cruzarem eles devem estar em planos de distância Z diferentes, sua área na imagem projetada será tão menor quanto mais longe o marcador estiver da câmera, possibilitando uma diferenciação entre os marcadores que se cruzam.

Ao final dos trabalhos, uma metodologia de testes para rastreadores e um acervo de vídeos foram criados. Espera-se que a metodologia e os vídeos gerados possam ser úteis na avaliação e comparação de outros trabalhos. Assim como o código do sistema que é aberto, todo este material estará disponível para futuros testes com outros rastreadores para marcadores ou, até mesmo, rastreamento de outras entidades

Tabela 8.1. Tabela comparativa relacionando as qualidades e limitações de cada um dos métodos. Esta tabela resume as conclusões observadas sobre o comportamento de cada método durante os experimentos.

Método	Prós	Contras
Kalman	- Melhor desempenho geral - Bom custo computacional - Área de busca variável ou fixa	- Número de trocas de semântica intermediário
CONDENSATION	- Bom em situações não lineares (curvas bruscas e acelerações) - Área de busca variável	- Maior valor de erro - Maior custo computacional - Maior número de trocas de semântica
Alfa-Beta	- Menor valor de erro - Menor custo computacional - Menor número de trocas de semântica	- Maior número de perdas - Área de busca fixa

relacionadas.

Também, foram alcançados os objetivos relacionados à melhoria do sistema através da expansão para múltiplas câmeras e implementação da câmera simulada. Atualmente, o sistema trabalha com um número n de câmeras, só limitado pelos recursos de *hardware* necessários para o processamento da captura. Além disso, as múltiplas câmeras reais podem trabalhar em conjunto com câmeras simuladas por vídeo ou separadamente.

A implementação dessas duas últimas funcionalidades também trouxeram o benefício da aprendizagem aprofundada de todo o processo de captura, e não apenas do módulo de rastreamento. Para incluir o suporte a múltiplas câmeras, foi necessário realizar modificações na calibração das câmeras (inicialização) e triangulação dos pontos (reconstrução), compreendendo assim todo o processo de captura de movimento.

Este trabalho focou na construção, avaliação e melhoria do sistema de código livre para a captura óptica de movimento, *OpenMoCap*. Ele está inserido em um projeto em desenvolvimento no Núcleo de Processamento Digital de Imagens (NPDI), com apoio da Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG), Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e Conselho Nacional de Desenvolvimento Científico e Tecnológico (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.

8.2 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.

- Desenvolver métodos que combinem as qualidades dos filtros de partículas com as qualidades dos filtros lineares ótimos.
- Desenvolver maneiras de variar a área de busca do filtro Alfa-Beta e avaliar seu desempenho.
- Incluir a área do marcador no estado do filtro de Kalman e do algoritmo CONDENSATION.

8.3 Publicações Oriundas Deste Trabalho

1. Flam, D. L.; de Queiroz, D. P.; de Souza Ramos, T. L. A.; de Albuquerque Araujo, A. & Gomide, J. V. B. (2009). Openmocap: An open source software for optical motion capture. Games and Digital Entertainment, Brazilian Symposium on, 0:151-161. IEEE Computer Society. http://doi.ieeecomputersociety.org/10.1109/SBGAMES.2009.25

2. Gomide, J.V.B., Flam, D.L., Queiroz, D.P. & Araújo, A. de A. Motion capture and character animation in games. Proceedings of the VIII Brazilian Symposium on Digital Games and Entertainment, SBGames, Rio de Janeiro, RJ, Brazil, 2009, pp 1-15.

3. Gomide, J. V. B.; Flam, D. L.; de Queiroz, D. P. & de Albuquerque Araujo, A. (2010). An open source motion capture system and its applications in arts and communication. In Proceedings, WCCA 2010 World Congress on Communication and Arts, Guimarães - Portugal. WCCA 2010 World Congress on Communication and Arts.

Apêndice A

Glossário

Como algumas seções desta dissertação contêm alguns termos de estatística e probabilidade, foi criado este glossário para que uma consulta rápida possa esclarecer suas dúvidas quanto a estes termos.

A.1 Covariância

A covariância é uma medida que relaciona duas variáveis aleatórias. Assim, a covariância entre duas variáveis aleatórias reais x e y é definida por uma medida de como duas variáveis aleatórias variam conjuntamente, sendo calculada por:

$$Cov(x, y) = E[(x - E[x])(y - E[y])]$$

Onde E é o operador do valor esperado. Uma maneira equivalente de realizar o cálculo é expressa por:

$$Cov(x, y) = E[xy] - E[x]E[y]$$

Pode-se entender a covariância como uma medida da relação linear entre duas variáveis aleatórias. Em outras palavras, é a medida de quanto o valor de uma variável está relacionada ao valor de outra.

Se duas variáveis têm covariância positiva, isso significa que quanto mais uma cresce mais a outra tende a crescer também; se a covariância for negativa, tem-se a relação inversa; e se a covariância for zero, então o valor de uma não interfere no valor da outra e elas são ditas independentes.

A covariância entre a mesma variável é igual à variância, de modo que:

$$Cov(x, x) = var(x)$$

Neste trabalho, muitas vezes são citadas as matrizes de covariância. Elas são matrizes simétricas que sumarizam esta relação de covariância entre N variáveis. Assim, é uma matriz em que o elemento da *i-ésima* linha da *j-ésima* coluna registra a covariância entre a variável i e a variável j, com $i \in \{1, \ldots, N\}$ e $j \in \{1, \ldots, N\}$. Seguindo a regra, percebe-se que a diagonal da matriz irá relacionar cada variável com ela mesma e, portanto, irá conter as variâncias.

A.2 Função de Densidade de Probabilidade (fdp)

Vide Variável Aleatória A.5.

A.3 Valor Esperado

O valor esperado, também conhecido como esperança matemática ou expectância de uma variável aleatória, é a soma dos possíveis valores de uma variável aleatória, multiplicados por suas respectivas probabilidades. Isto é, representa o valor médio esperado de uma experiência, se ela for repetida muitas vezes. Note-se que o valor pode ser improvável ou impossível, por não ser um dos valores do conjunto de valores possíveis da variável aleatória. Se todos os eventos tiverem igual probabilidade o valor esperado é a média aritmética. Assim, o valor esperado da variável aleatória x, representado por $E[x]$, pode ser calculado pelo seguinte somatório:

$$E[x] = \sum_{i=1}^{\infty} x_i p(x_i)$$

Sendo x_i a representação de cada um dos possíveis valores de x.

A.4 Variância

A variância de uma variável aleatória é uma medida da sua dispersão estatística, indicando quão longe em geral os seus valores se encontram do valor esperado. No escopo deste trabalho, utilizando aspectos práticos, para uma variável aleatória x pode-se defini-la como o quadrado do desvio padrão, σ_x^2.

Desta forma, a variância mede o quanto os valores de uma variável aleatória tendem a se desviar da média. Uma variância alta indica valores mais dispersos em relação à média, já uma variância baixa, indica valores mais concentrados nas proximidades da média.

A.5 Variável Aleatória

Uma variável aleatória representa a medição de algum parâmetro que pode gerar um valor diferente a cada medida - por exemplo, o resultado de jogar um dado pode dar qualquer número de 1 a 6. Considerando o exemplo de lançar o dado, a variável aleatória x, que traduz o resultado desta experiência, pode tomar os valores 1, 2, 3, 4, 5 ou 6. E $p(x)$ representa a probabilidade de algum valor - do conjunto de valores possíveis de x - ocorrer, sendo, neste caso, igual a $1/6$ para todos os valores de x.

O exemplo citado demonstra uma variável aleatória discreta, para variáveis aleatórias contínuas, os termos função de densidade de probabilidade ou função de distribuição de probabilidade são mais utilizados. Assim, ao invés do conjunto de valores x, temos a função $fdp(x)$. Neste caso, para encontrar a probabilidade de um dado intervalo, deve-se integrar a função neste intervalo e para encontrar a probabilidade de um valor específico, deve-se utilizar a derivada deste valor.

Referências Bibliográficas

A. Kamalvand, P. M. & Tran, T. (2004). Factored sampling tracking: Comparison of the kalman and the condensation algorithms for missile tracking in a defense target environment. Technical Report DAAD 19-02-1-0357, U.S. Army Research Office. 15

Adobe (2011). Adobe. Este é um documento eletrônico disponível em: http://www.adobe.com. Acessado em: 24 de Janeiro de 2011. 50

Animazoo (2010). Motion Capture Comparison Table. Este é um documento eletrônico disponível em: http://www.animazoo.com. Acessado em: 14 de Dezembro de 2010. xv, 3, 4

Ascension Technology Corporation (2010). ReActor 2. Este é um documento eletrônico disponível em: http://www.ascension-tech.com. Acessado em: 14 de Dezembro de 2010. xv, 5

Bazzani, L.; Bloisi, D. & Murino, V. (2009). A comparison of multi-hypothesis Kalman filter and particle filter for multi-target tracking. In *Performance Evaluation of Tracking and Surveillance workshop at CVPR 2009*, pp. 47--54, Miami, Florida. 15

Boslaugh, S. & Watters, D. P. A. (2008). *Statistics in a nutshell*. O'Reilly & Associates, Inc., Sebastopol, CA, USA. 57

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. 14

da Silva, F. W. (1997). Motion capture - introdução à tecnologia. Technical report, LCG - COPPE/SISTEMAS, UFRJ. 2

de Queiroz, D. P. (2010). Tracking sample videos. Este é um documento eletrônico disponível em: http://www.npdi.dcc.ufmg.br/mocap/. Acessado em: 15 de Dezembro de 2010. 56, 62, 64, 71

Fieguth, P. (2011). *Statistical Image Processing and Multidimensional Modeling*. Springer, New York, NY, USA. 32

Figueroa, P. J. (1998). Perseguição de marcadores para análise de movimentos humanos. Dissertação de Mestrado, UNICAMP. 13

Figueroa, P. J.; Leite, N. J.; Barros, R. L. & Brenzikofer, R. (1998). Tracking markers for human motion analysis. In *Proc. of IX European Signal Processing Conf*, pp. 941--944. 13

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. 13, 14, 16

Flam, D. L. (2009). Openmocap: Uma aplicação de código livre para a captura Óptica de movimento. Dissertação de Mestrado, UFMG. xv, 17, 18, 20, 22, 67

Flam, D. L.; de Queiroz, D. P.; de Souza Ramos, T. L. A.; de Albuquerque Araujo, A. & Gomide, J. V. B. (2009). Openmocap: An open source software for optical motion capture. *Games and Digital Entertainment, Brazilian Symposium on*, 0:151–161. IEEE Computer Society. http://doi.ieeecomputersociety.org/10.1109/SBGAMES.2009.25. xv, 2, 21, 24

Fraga, L. G. & Silva, I. V. (2008). Direct 3d metric reconstruction from two views using differential evolution. In *IEEE Congress on Evolutionary Computation*, pp. 3266–3273. IEEE. 19

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. 12

Gomide, J. V. B. (2006). Captura Digital de Movimento no Cinema de Animação. Dissertação de Mestrado, EBA/UFMG. 2

Gomide, J. V. B.; Flam, D. L.; de Queiroz, D. P. & de Albuquerque Araujo, A. (2010). An open source motion capture system and its applications in arts and communication. In *Proceedings, WCCA 2010 World Congress on Communication and Arts*, Guimarães - Portugal. WCCA 2010 World Congress on Communication and Arts. 2, 5

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. 21

Intel Corporation (2010). Laptop, notebook, desktop, server and embedded processor technology - intel. Este é um documento eletrônico disponível em: http://www.intel.com. Acessado em: 9 de Decembro de 2010. 24

Isard, M. & Blake, A. (1998). Condensation-conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5--28. 13, 35, 40, 43

Isard, M. & Blake, A. (2011). The condensation algorithm home page. Este é um documento eletrônico disponível em: http://code.google.com/p/video-tracker. Acessado em: 18 de Fevereiro de 2011. 42

Isard, M. & MacCormick, J. (2001). Bramble: A bayesian multiple-blob tracker. In *ICCV*, pp. 34–41. 14

Ivo D. Dinov, Nicholas Christou, J. S. (2008). Central limit theorem: New socr applet and demonstration activity. *Journal of Statistics Education*, 16(2). 57

Julier, S.; Uhlmann, J. & Durrant-Whyte, H. (1995). A new approach for filtering nonlinear systems. In *American Control Conference, 1995. Proceedings of the*, volume 3, pp. 1628 –1632 vol.3. 13

Julier, S.; Uhlmann, J. & Durrant-Whyte, H. F. (2002). A new method for the nonlinear transformation of means and covariances in filters and estimators. *Automatic Control, IEEE Transactions on*, 45(3):477--482. 13

Julier, S. & Uhlmann, J. K. (1996). A general method for approximating nonlinear transformations of probability distributions. Technical report, Robotics Research Group, Department of Engineering Science - University of Oxford. 13

Julier, S. J. & Uhlmann, J. K. (1997). A new extension of the kalman filter to nonlinear systems. In *Proc. SPIE Vol. 3068, Signal Processing, Sensor Fusion, and Target Recognition VI, Ivan Kadar; Ed.*, pp. 182--193. 13

Kalman; Rudolph & Emil (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35--45. 13, 25

Lawton, J. A.; Jesionowski, R. J. & Zarchan, P. (1998). Comparison of four filtering options for a radar tracking problem. *AIAA Journal of Guidance, Control and Dynamics*, 21(4):618–623. 45

Marron, M.; Garcia, J.; Sotelo, M.; Cabello, M.; Pizarro, D.; Huerta, F. & Cerro, J. (2007). Comparing a kalman filter and a particle filter in a multiple objects tracking application. In *Intelligent Signal Processing, 2007. WISP 2007. IEEE International Symposium on*, pp. 1 –6. 14

Maskell, S. & Gordon, N. (2001). A tutorial on particle filters for on-line nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50:174--188. 42

Maybeck, P. S. (1979). *Stochastic models, estimation, and control*, volume 141 of *Mathematics in Science and Engineering*. Academic Press, Inc. (London). 26

Menache, A. (2000). *Understanding Motion Capture for Computer Animation and Video Games*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 1

Moeslund, T. B. (2000). Interacting with a virtual world through motion capture. In *Interaction in Virtual Inhabited 3D Worlds, chapter 11*. Springer-Verlag. 1, 12

Moeslund, T. B. & Granum, E. (2001). A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231--268. 1, 4, 12, 18

Moeslund, T. B.; Hilton, A. & Krüger, V. (2006). A survey of advances in vision-based human motion capture and analysis. *Comput. Vis. Image Underst.*, 104(2):90--126. 12

Moghaddam, B. & Pentland, A. (1995). Probabilistic visual learning for object detection. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pp. 786--793. 42

Morais, E. F. (2005). Rastreamento e contagem de peixes utilizando filtro preditivo. Dissertação de Mestrado, DCC - UFMG. 14, 40

Morais, E. F.; Campos, M. F. M.; Padua, F. L. C. & Carceroni, R. L. (2005). Particle filter-based predictive tracking for robust fish counting. In *Proceedings of the XVIII Brazilian Symposium on Computer Graphics and Image Processing*, pp. 367--, Washington, DC, USA. IEEE Computer Society. 14, 16

Motion Analysis Corporation (2010). The industry leader for 3d passive optical motion capture. Este é um documento eletrônico disponível em: http://www.motionanalysis.com. Acessado em: 29 de Dezembro de 2010. xv, 6, 22

Navarro, A. M. (1977). General properties of alpha beta, and alpha beta gamma tracking filters. *NASA STI/Recon Technical Report N*, 77:24347–+. 45

Nicola Martorana, Iacopo Masi and Marco Meoni (2011). Video-tracker. Este é um documento eletrônico disponível em: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCALCOPIES/ISARD1. Acessado em: 18 de Fevereiro de 2011. 42

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. 22

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. 13

University of Wisconsin-Madison (2010). Biovision bvh. Este é um documento eletrônico disponível em: http://www.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/BVH.html. Acessado em: 29 de Dezembro de 2010. 22

Welch, G. & Bishop, G. (1995). An introduction to the kalman filter. Technical report, Department of Computer Science - University of North Carolina at Chapel Hill, Chapel Hill, NC, USA. 13, 34, 50

Willow Garage (2010). OpenCV - Wiki. Este é um documento eletrônico disponível em: http://pr.willowgarage.com/wiki/OpenCV. Acessado em: 1 de Dezembro de 2010. 24

Yoo, J.-C. & Kim, Y.-S. (2003). Alpha-beta-tracking index (alpha-beta-lambda) tracking filter. *Signal Processing*, 83(1):169--180. 13, 45

Zeng, W.; Zhu, G. & Li, Y. (2009). Point matching estimation for moving object tracking based on kalman filter. *Computer and Information Science, ACIS International Conference on*, 0:1115–1119. 14