INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

PREDGRAB - Predição de Trajectórias de Alvos Móveis

Aplicação ao controlo de um braço robot

Paulo Jorge Pereira Carreiras

Dissertação para obtenção do grau de Mestre em

Engenharia Electrotécnica e de Computadores

Júri

Presidente:	Doutor Francisco Miguel Prazeres Silva Garcia
Orientador:	Doutor Alexandre José Malheiro Bernardino
Co-Orientador:	Doutor José Alberto Rosado dos Santos Victor
Vogais:	Doutor Pedro Manuel Urbano Almeida Lima

Setembro 2007

Resumo

Neste trabalho criou-se um sistema de controlo em tempo real para aplicação num robot antropomórfico composto por um braço robótico e uma cabeça stereo. O objectivo principal é o de alcançar um objecto móvel com um corpo com inércia, baseado em medidas da posição do objecto no instante presente, o que é uma tarefa complicada, especialmente se estiver envolvida a movimentação de massas consideráveis.

A abordagem seguida baseia-se na modelação matemática do movimento do móvel tendo em conta a trajectória que tem vindo a descrever. O comando de actuação é dado de acordo com a predição da trajectória do móvel, consistindo na estimativa da posição 3D. Esta estimativa é referente a um instante de tempo suficientemente distante para garantir a compensação do tempo associado ao regime dinâmico da movimentação do braço do robot.

Para garantir uma boa predição são utilizados modelos implementados com filtros de *Kalman* a correr em paralelo com diferentes hipóteses relativas ao tipo de movimento. A posição para a qual o robot é comandado é o resultado da combinação das estimativas dos modelos considerados. Esta combinação adaptativa visa minimizar o erro entre a estimativa e a posição real do objecto e resume-se ao cálculo da média ponderada dos vectores de estado pesados pela probabilidade condicional de cada um dos modelo ser válido.

Para melhorar o desempenho do sistema foi ainda considerado um preditor baseado na autocorrelação do sinal. O objectivo é fazer a estimativa da posição de movimentos periódicos com base na observação de um conjunto de períodos. A combinação deste estimador com os restantes baseados nas equações às diferenças do vector de estado, é feito recorrendo a um teste simples de hipóteses admitindo uma função densidade de probabilidade do tipo Chi-quadrado.

Para validação dos resultados é utilizado um protótipo de um robot humanóide disponível e é desenvolvido um simulador do mesmo.

Para que esta implementação seja possível são ainda consideradas questões relacionadas com a cinemática e a estimativa da posição do móvel baseada na reconstrução tridimensional realizada a partir de um par de cameras estéreo.

Palavras Chave: Predição de trajectórias, combinação adaptativa de modelos, filtros de kalman, cinemática, segmentação de imagem, reconstrução 3D

Abstract

This work presents an approach do deal with the positioning of inertial robots within the trajectory of a target.

The displacement of robots introduces delays to achieve a desired position due to the dynamics response of its mass. The suggested strategy predicts the target position, based on the mathematical modulation of its trajectory, exploring the regularity usually present on the motion. The actuation command is issued to the predicted position, with a certain time span, instead of the actual location of the target.

Several kalman models designed for different motion hypothesis run in parallel to grant a good intersection with a generic moving target. A multiple model adaptive estimation technique based on the conditional probability is employed to get together the contribution of all models. This probability is computed to each one of the models and uses the observed accuracy performance.

Additionally a non kalman model based on the autocorrelation is used to deal with periodic trajectories. A basic hypothesis test assuming chi-squared error distribution is applied to select the output between these different methodologies.

The algorithm is tested in a humanoid robot previously assembled which is available in the laboratory. A simulator is also build with the aim of test and validate the algorithm before dealing with the real robot.

The 3D reconstruction based on a pair of cameras disposed in a human like fashion is also discussed, and presented a segmentation algorithm to track the object trough the cameras' images. This is done using the *camshift* algorithm implemented with the high performance open cv functions.

Key-words: Trajectory prediction, multiple model adaptive estimation, kalman filters, kinematics, image segmentation, 3D reconstruction

Conteúdo

Lista de Figuras

Lista de Tabelas

1 Introdução

A robótica tem sido nas últimas décadas uma das áreas de investigação com maior produção de artigos científicos. Na criação de robots são empregues conhecimentos de diversas áreas, nomeadamente da electrónica, controlo, mecânica, física, software, entre outras. A deslocação de corpos é uma temática comum a muitos dos robots desenvolvidos. Para situações simples a hipótese de massa desprezável pode ser aceitável, contudo em situações mais exigentes é necessário fazer outras considerações.

O trabalho que agora se apresenta aborda uma estratégia de actuação que visa facilitar a actuação dos robots no mundo real. O problema que se pretende resolver é o do posicionamento atempado do robot sobre a trajectória de um móvel, a fim de permitir interceptá-lo.

Na maior parte das situações, as trajectórias dos alvos apresentam regularidades que podem ser exploradas para efectuar o seu seguimento com erro em regime permanente suficientemente baixo. A intercepção do móvel é conseguida através de um algoritmo que permite predizer a posição do móvel alguns instante de tempo no futuro. Esta técnica, inspirada na biologia, ganha tempo para que todo o regime dinâmico do robot se manifeste, dando ordem para posicionamento não na posição actual, mas sim na posição prevista para o móvel. Na realidade esta estratégia é também realizada pelos seres humanos, visto que os "pedidos"de actuação enviados pelo cérebro para os diversos elementos do corpo humano são referentes a acontecimentos futuros preditos pelo cérebro.

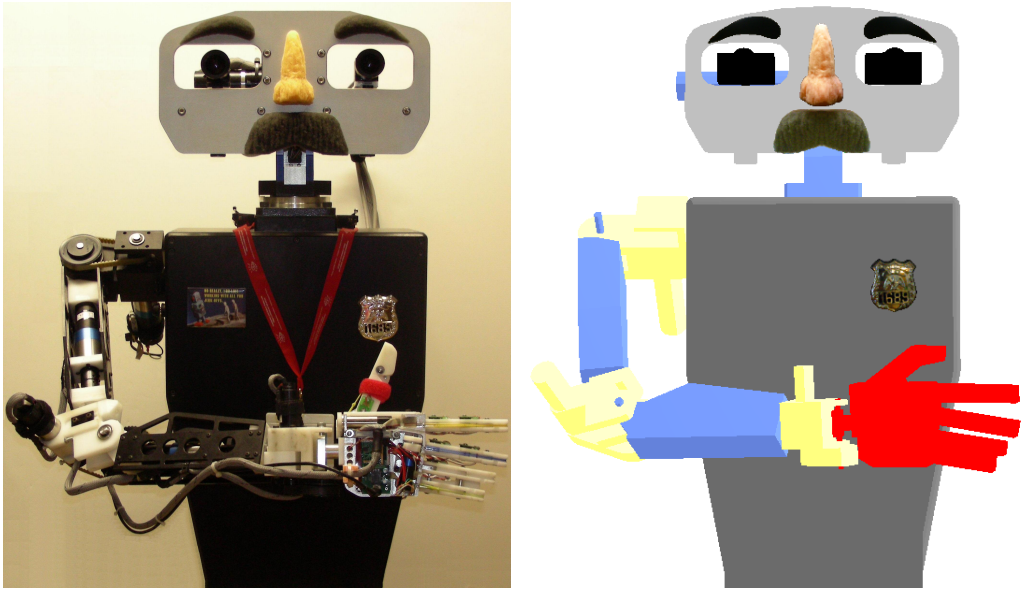
O trabalho foi desenvolvido num robot humanóide já existente - Baltazar [1]. Este robot está equipado com um manipulador série que permite um comportamento semelhante ao braço humano. O robot tem ainda duas cameras cuja posição e orientação é comandada por um conjunto de juntas, equiparando-se ao funcionamento dos olhos humanos.

Nos capítulos seguintes abordam-se em detalhe as questões relacionadas com a concretização do objectivo de posicionamento atempado do braço do robot. Numa primeira fase trata-se do problema do posicionamento do braço num ponto tridimensional conhecido, sem ter em conta qualquer comportamento dinâmico, isto é, tendo em conta apenas a cinemática.

A estimativa da posição do móvel a partir das imagens das duas cameras é também abordada, assim como se resolve o problema do posicionamento das mesmas a fim de conseguir seguir o corpo móvel. Nos capítulos 4 e 5 apresenta-se a estratégia de predição da posição, nomeadamente as técnicas e o desenvolvimento de modelos para o movimento. Resolve-se também aqui o problema da estimação da posição a partir de estimativas produzidas por diferentes modelos de movimento. Finalmente aborda-se a questão relacionada com antecipação realmente necessária para interceptar o móvel, fazendo uma estimativa do horizonte de predição.

No capítulo 7 é feita uma descrição sumária do sistema disponível e é criado um simulador para facilitar o desenvolvimento do trabalho. As técnicas desenvolvidas ao longo dos vários capítulos são validadas por experiências realizadas num simulador [28] e no robot real, apresentando-se no final do trabalho uma secção com resultados experimentais da utilização do algoritmo realizados no robot real.

Figura 1: Modelo CAD do *Baltazar* e robot real

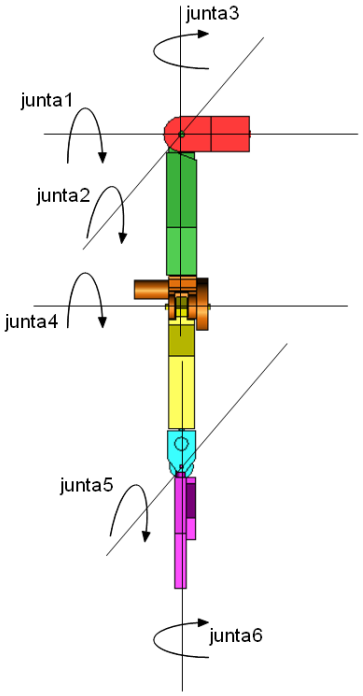


2 Cinemática do Braço Antropomórfico

A cinemática estuda o movimento de corpos tendo apenas em conta as características geométricas destes, isto é, sem considerar quaisquer aspectos relacionados com forças. O modelo cinemático de um robot expressa a forma como os diversos componentes se movem entre si realizando uma transformação entre espaços de configuração. Os espaços em causa são o espaço de trabalho e o espaço directamente associado aos graus de liberdade do robot. Quando esta transformação se dá do espaço associado aos graus de liberdade do robot para o espaço de trabalho, designa-se por *Cinemática Directa*, designando-se por *Cinemática Inversa* caso a transformação se dê em sentido contrário.

O *Baltazar* possui um braço antropomórfico inspirado no braço humano. Dada a complexidade das articulações do braço humano, não é ainda tecnologicamente viável reproduzir este membro. Assim é necessário proceder a algumas simplificações e que consequentemente levam à perda de alguns graus de liberdade e consequentemente da capacidade de manobra do membro. O braço do *Baltazar* apresenta um bom compromisso entre complexidade e capacidade de replicação do braço humano, sendo composto por 6 juntas, estando atribuídos 3 graus de liberdade ao ombro que permitem flectir, rodar e levantar o braço, 2 ao cotovelo que permitem fazer a extensão/flexão e rotação do ante-braço, e 1 ao punho que permite a sua flexão. O *Baltazar* encontra-se na figura 1, podendo-se ver o braço em mais pormenor na figura 2. Nas secções seguintes apresenta-se a cinemática directa e inversa do braço.

Figura 2: Detalhe do braço do *Baltazar*



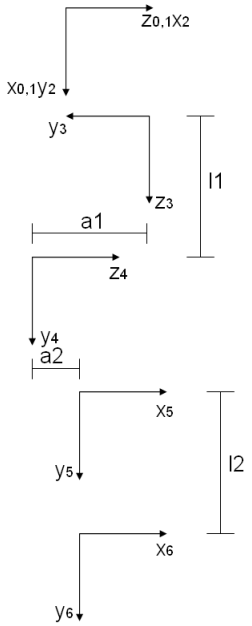
2.1 Cinemática Directa

O cálculo da cinemática directa permite, para uma dada configuração do braço, ter em conta a posição de cada uma das juntas constituintes do braço do *Baltazar*, possibilitando, por exemplo, a operação do braço do robot na proximidade de obstáculos com posições conhecidas. A cinemática directa será utilizada neste projecto como critério adicional na escolha a fazer sobre as múltiplas soluções resultantes da cinemática inversa. O conjunto de soluções fica assim restringido as hipóteses que cumpram os limites impostos às juntas intermédias do braço robótico, nomeadamente à posição do cotovelo.

Esta cinemática será também utilizada na avaliação do desempenho do algoritmo de predição, visto que permite fazer a comparação entre a posição do móvel e do braço. Na realidade esta medida do erro não é exacta uma vez que a determinação das coordenadas do objecto e a leitura da posição das juntas do braço não são síncronas.

Por questões de sistematização do processo de descrição das transformações, utiliza-se os parâmetros de *Denavit-Hartenberg Modificados* [2] para a representação do braço do robot. Nesta abordagem o braço é representado como uma associação de troços, aos quais se associam referenciais cuidadosamente escolhidos. Os referenciais escolhidos para o braço robótico apresentam-se na figura 3. Existem algumas regras para a associação dos referenciais a cada uma das juntas. O eixo dos zz deve estar na direcção do eixo da junta, o eixo dos xx deve ser paralelo à normal comum das duas juntas e o eixo dos yy é o resultante dada a

Figura 3: Referenciais atribuídos a cada uma das juntas do braço do robot



localização dos eixos x e z.

Os parâmetros de *Denavit-Hartenberg* são $a_{i-1}, \alpha_{i-1}, d_i$ e θ_i. O primeiro é a distância entre as origens dos referenciais i e $i+1$ medida ao longo de x_i, α_{i-1} é o ângulo entre z_i e z_{i+1} medido em x_i. O terceiro parâmetro, d_i, é a distância entre as origens dos referenciais $i-1$ e i medida segundo z_i enquanto que θ_i é o ângulo entre x_{i-1} e x_i medido em torno de z_i. Com base nestes parâmetros é possível, recorrendo à matriz apresentada na equação 1 determinar a matriz de rotação e o vector translação entre dois troços consecutivos. Os parâmetros de *Denavit-Hartenberg Modificados* resultantes para o *Baltazar* apresentam-se na tabela 1. A ordem dos referenciais aqui utilizada é a apresentada na figura 2.

$$T_{i+1}^{i} = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i) & 0 & a_{i-1} \\ sin(\theta_i)cos(\alpha_{i-1}) & cos(\theta_i)cos(\alpha_{i-1}) & -sin(\alpha_{i-1}) & -sin(\alpha_{i-1})d_i \\ sin(\theta_i)sin(\alpha_{i-1}) & cos(\theta_i)sin(\alpha_{i-1}) & cos(\alpha_{i-1}) & cos(\alpha i-1)d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

A posição e orientação do troco j relativa ao referencial base é obtida através da multiplicação das várias matrizes transformação de troço, como descrito na expressão seguinte. A matriz de rotação corresponde às três primeiras linhas e colunas, enquanto que o vector de posição corresponde à ultima coluna da matriz, encontrando-se expresso em coordenadas homogéneas.

$$T_j^0 = \prod_{i=0 \ i \leq j} T_{i+1}^{i}$$

Tabela 1: Parâmetros de Denavit-Hartenberg

Junta	a_{i-1}[cm]	α_{i-1}[cm]	d_i[º]	θ_i[º]
1	0	0	0	θ_1
2	0	90	0	$\theta_2 + 90$
3	0	90	29.13	$\theta_3 + 90$
4	2.82	90	0	θ_4
5	-2.18	-90	26.95	$\theta_5 + 90$
6	0	90	0	θ_6

Para um vector expresso em coordenadas homogéneas $x = [x'_1\ x'_2\ x'_3\ \alpha]^T$, o vector em coordenadas cartesianas correspondente $x = [x_1\ x_2\ x_3]^T$ é dado por

$$x_i = \alpha x'_i$$

2.2 Cinemática Inversa

Dado que o algoritmo de predição opera sobre coordenadas 3D é fundamental saber passar pontos deste espaço para o espaço de juntas do robot. Sendo assim, o cálculo da cinemática inversa torna-se imprescindível para a concretização do objectivo proposto. Esta técnica, apresenta no entanto um inconveniente relacionado com a ausência de um mecanismo de *feedback*. Sempre que se pretende posicionar o braço do robot numa posição específica procede-se simplesmente ao cálculo das posições das juntas com base apenas no modelo geométrico introduzido *a priori* nos cálculos, não sendo feito qualquer ajuste no posicionamento eventualmente necessário devido a erros de modelação. Existem na bibliografia outras alternativas baseadas normalmente em redes neuronais para fazerem a aprendizagem desta transformação de espaços [3], não sendo no entanto abordadas neste trabalho.

A cinemática inversa utilizada baseia-se no algoritmo iterativo descrito em [1], sendo realizada em duas fases: posicionamento do punho e orientação da mão.

2.2.1 Posicionamento do Punho

Para o posicionamento do braço na posição P, é necessário determinar $\theta_1, \theta_2, \theta_3$ e θ_4. Dado que muitas das equações associadas a estas variáveis são transcendentes o resultado apresentado na equação 2 será por diversas vezes utilizado.

$$acos\theta + bsin\theta = c \quad \theta = 2arctan\left(\frac{b \pm \sqrt{a^2 + b^2 - c^2}}{a + c}\right) \tag{2}$$

Dada a geometria do braço do robot e tendo em conta as relações trignométricas e o teorema de pitágoras,

Tabela 2: Limites das juntas do braço do *Baltazar*

Junta	1	2	3	4	5	6
Limite [º]	[-45,135]	[-110,10]	[-90,0]	[-90,0]	[-90,90]	[-45,45]

pode-se escrever para θ_4

$$(2(-a_2a_1 + l_2l_1))cos(\theta_4) + (-2(l_2a_1 + a_2l_1))sin(\theta_4) = \rho^2 - (a_2^2 + l_2^2 + l_1^2 + a_1^2) \tag{3}$$

Esta equação encontra-se escrita de acordo com (2), permitindo assim determinar o valor de θ_4.
Fazendo $Z = [0\,0\,0\,1]^T$, a posição $P = [x\,y\,z\,1]^T$ do punho é dada pela equação 4

$$P = \prod_{i=0}^{5} T_{i+1}^{i} Z \tag{4}$$

A solução para θ_2 e uma restrição em θ_3 são obtidas analisando a componente z dada por (4). Para os parâmetros da equação (2) de forma a existir solução para θ_2, tem-se que θ_3 deve ser tal que $a^2 + b^2 - c^2 > 0$, com

$$\begin{aligned} a &= d_2cos(\theta_4) + l_2sin(\theta_4) - d_1sin(\theta_3) \\ b &= -d_1sin(\theta_4) + l_2cos(\theta_4) + l_1 \\ c &= z \end{aligned} \tag{5}$$

O algoritmo de cálculo da cinemática consiste em inicializar θ_3 de forma a cumprir esta restrição e calcular os ângulos das restantes juntas. Caso seja impossível determinar a solução para todas as varáveis, é escolhido outro valor para θ_3 e iniciando novamente o processo de determinação das variáveis, até que que uma solução geral seja descoberta.
A solução para θ_1 é obtida do sistema de 2 equações resultantes da expressão seguinte

$$T_1^2 T_0^1 P = T_3^2 T_4^3 T_5^4 T_6^5 Z \tag{6}$$

Reescrevendo estas equações de acordo com o resultado apresentado em 2 fica

$$\begin{cases} a = -cos(\theta_2)x \\ b = -cos(\theta_2) * y \\ c = sin(\theta_4)d_2 - cos(\theta_4)l_2 - l_1 + sin(\theta_2)z \\ a = -y \\ b = x \\ c = -cos(\theta_3)(cos(\theta_4)d_2 + sin(\theta_4)l_2 - d_1) \end{cases}$$

Para todas as variáveis são sempre testadas as duas soluções de (2) e verificada a sua coerência com os limites físicos de cada um dos motores do braço. A listagem dos limites encontram-se na tabela 2.

2.2.2 Orientação da Mão

No que respeita à orientação, é suficiente especificar as restrições para um plano, sendo deixado livre um grau de liberdade. Ao posicionar o punho paralelo ao referido plano, a estrutura da mão oferece a flexibilidade extra necessária para a manipulação de objectos. Esta questão não será no entanto abordada neste trabalho.

Dado o sistema de eixos escolhido, o eixo dos x, no referencial do punho é perpendicular à mão, pelo que basta igualar a primeira coluna da matriz R_6^0 à normal N do plano pretendido (equação 7). Dado que R_0^4 já é conhecido é simples determinar a solução para θ_5 e θ_6. O sistema de equações para θ_5 e θ_6 é:

$$R_6^0 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix} \tag{7}$$

2.2.3 Escolha das Soluções

A fim de optimizar o funcionamento do robot, é necessário determinar todas as configurações no espaço de junta que conduzem ao ponto pretendido P. Isto significa que a procura em θ_3 deve ser exaustiva, passando por todos os valores possíveis para esta variável.

Sobre o conjunto de soluções possíveis são aplicadas duas condições: uma que limita o posicionamento das juntas intermédias e outra que minimiza a variação na posição angular das juntas do braço do *Baltazar* entre cada 2 pontos 3D preditos. Para cada uma das soluções disponibilizadas, é aplicada a cinemática directa ao espaço de junta e é determinada a posição da junta correspondente ao cotovelo. Todas as soluções que não cumpram a restrição do posicionamento do cotovelo são eliminadas.

Para a minimização dos deslocamentos das juntas é possível enveredar por dois caminhos distintos: a via analítica ou procura exaustiva. Dada a complexidade das equações da cinemática, optou-se por seguir a versão iterativa, conduzindo no entanto a um maior peso computacional. Para a implementação do critério de minimização do deslocamento das juntas é suficiente calcular para cada uma das soluções restantes, a diferença em relação à posição anterior de cada uma as juntas e escolher a solução que minimiza este resultado. Convém ainda notar que esta opção tem o inconveniente de não garantir a convergência para a solução óptima, podendo sempre conduzir a um mínimo local.

3 Modelo de observação do alvo

Do ponto de vista deste trabalho, assume-se a existência de um sensor baseado em imagem stereo que obtém medidas da posição 3D de alvos coloridos num referencial fixo. Apesar de não ter sido efectuado este trabalho, ele resultou essencialmente de um projecto paralelo e apresentam-se aqui os métodos e resultados.
Para obter medidas acerca da posição 3D do alvo, utiliza-se a reconstrução stereo da posição obtida nas duas imagens. Um primeiro passo consiste na segmentação do alvo nas imagens. As coordenadas do alvo nas imagens são obtidas pelo centróide da área segmentada. Estas coordenadas são depois utilizadas num processo de triangulação para estimar a posição do objecto no espaço 3D.

3.1 Segmentação de Imagem

A segmentação de imagem consiste em dividir a imagem em dois ou mais subconjuntos de acordo com um determinado critério. Desta forma, cada subconjunto é composto pelos pixeis da imagem que satisfazem uma mesma condição, que pode ser por exemplo, ter a mesma cor, ter a mesma textura, apresentar transições bruscas, entre muitos outros.
O objectivo pretendido é, a partir da selecção de um objecto, fazer o seu seguimento ao longo do tempo.
Estão disponíveis na literatura inúmeras técnicas de segmentação de imagem que permitem solucionar este problema cada uma das quais com vantagens e desvantagens. Na primeira secção apresenta-se uma breve listagem das técnicas mais comuns e respectivos prós e contras. Seguidamente, apresenta-se em detalhe o algoritmo de segmentação escolhido - o camshift [4], [5],[6], assim como uma breve análise do seu desempenho. Aborda-se também aqui a questão da reconstrução tridimensional e a interligação entre os algoritmos de segmentação e de reconstrução. Finalmente apresentam-se alguns dados referentes ao desempenho do sistema.

3.1.1 Algoritmos de Segmentação

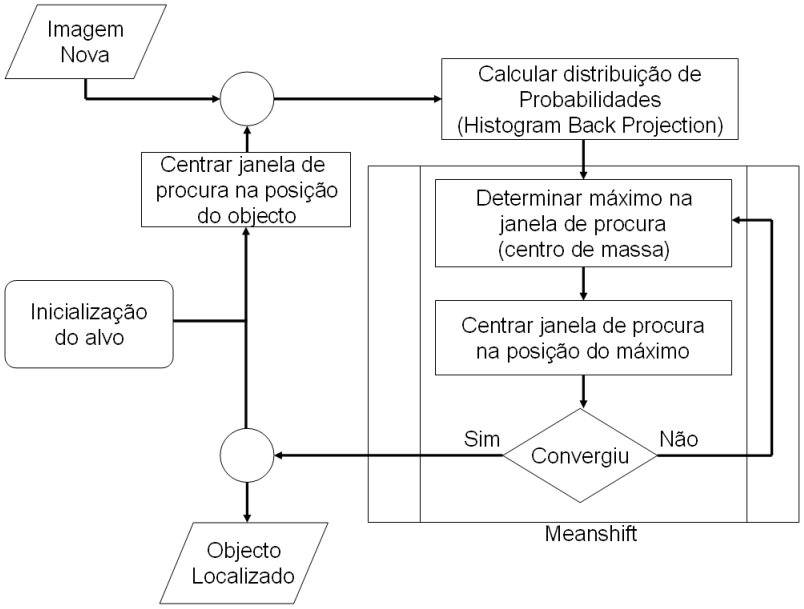
O problema do seguimento de objectos e segmentação de imagens pode ser abordado de diferentes perspectivas. Existem métodos complexos que realizam o seguimento com base na detecção dos contornos do objecto com *snakes* |citesnakes, outros utilizam técnicas de associação baseadas em valores próprios [8], ou ainda outros que calculam a convolução da imagem com padrões pré-definidos [9]. Contudo, muitos destes algoritmos não cumprem a totalidade dos requisitos necessários para a tarefa pretendida: pretende-se um método capaz de fazer seguimento na presença de ruído, de objectos semelhantes e acima de tudo que seja rápido e eficaz, uma vez que vai operar em tempo real em paralelo com um conjunto significativo de tarefas computacionalmente pesadas que permitem a intercepção do braço do robot com o móvel.
Uma outra abordagem possível é a interpretação estatística do movimento. Fazendo uma interpretação Bayseana do problema, a solução para a estimação do movimento pode passar pela inferência e comparação de diferentes funções de verosimilhança geradas por diferentes modelos de movimento [10].

Também é possível optar por utilizar o algoritmo EM - *Expectation Maximization* para a escolha do melhor modelo. Este algoritmo, muitas vezes utilizado em problemas estatísticos, baseia-se em dois passos: um de extrapolação e um de maximização. No primeiro passo associam-se todos os pontos concordantes para um dado modelo escolhido ao acaso e no segundo passo actualiza-se os parâmetros deste modelo com base nos pontos que lhe foram atribuídos. O algoritmo repete-se até que os parâmetros do modelo estabilizem.
Esta classe de soluções apresenta um problema semelhante ao abordado neste trabalho. Por um lado a necessidade de estabelecer a priori o conjunto de modelos de movimento que o objecto pode apresentar, por outro a escolha do melhor modelo de movimento. Neste sentido o trabalho realizado no âmbito dos modelos de movimento 3D poderá também prestar algum contributo na área de processamento de imagem.
Muitas outras soluções e abordagens para o problema são possíveis, mas, uma vez que o desenvolvimento de um algoritmo de *tracking* não é o principal objectivo deste trabalho, optou-se por escolher uma solução disponibilizada no *opencv - Intel Open Source Computer Vision Library*. Esta biblioteca composta por funções de C e C++ reúne um conjunto de funcionalidades normalmente utilizadas no processamento de imagem, assim como implementa alguns algoritmos usuais. Entre as várias hipóteses disponíveis escolheu-se o camshift, que será abordado na secção seguinte.

3.1.2 Camshift

O camshift é um algoritmo robusto, não paramétrico baseado no seguimento do máximo do gradiente da distribuição de probabilidade - *meanshift* [11]. O *meanshift* é um algoritmo iterativo que determina o centro de massa da distribuição de probabilidades na janela de procura e centra neste ponto a nova janela a utilizar na próxima iteração. O processo repete-se até que não haja alteração significativa da posição.
O passo inicial do Camshift é estabelecer à área de interesse. Seguidamente centra a janela do meanshift no objecto que se pretende seguir e determina a distribuição de probabilidades desta região. A partir daqui itera o meanshift até não haver alteração na posição, estabelece neste ponto a nova janela de procura e torna a repetir todo o processo enquanto se pretender fazer seguimento deste objecto.
Para que tudo isto seja possível é necessário determinar a distribuição de probabilidades da imagem. Esta distribuição pode ser determinada por um qualquer método que associe a cada pixel o valor da probabilidade deste pertencer ao alvo. O método utilizado é designado por *Histogram Back-Projection* e é aplicado ao canal *hue* da imagem no espaço de cores HSV.
Esta técnica desenvolvida por Swain e Ballar em 1991, recorre ao rácio de histogramas, ou seja, à relação entre o histograma do objecto seleccionado e o histograma correspondente à zona da janela de cálculo que está a ser analisada dentro da janela de procura. A janela de cálculo é uma área móvel, que se desloca por toda a janela de procura, sobre a qual se calcula o referido histograma. Desta forma pretende-se aumentar a diferença entre o *backgroud* e o objecto, levando a uma localização mais fiável deste. O *Histogram Back-Projection*

Figura 4: Fluxograma do *camshift*



consiste em atribuir a cada pixel da imagem a probabilidade correspondente do rácio dos histogramas. A imagem assim resultante é normalmente composta por zonas em que os pixeis tomam valores mais elevados e que corresponde à região do objecto que se pretende localizar.

Para viabilizar o seguimento com o *camshift* é ainda necessário que o tamanho da janela se adapte ao alvo que está a ser seguido, uma vez que o tamanho ideal da janela varia consoante o objecto se encontra mais perto ou mais afastado das cameras.

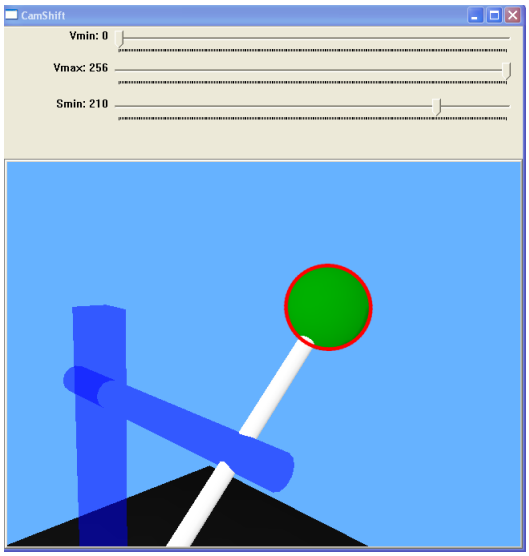
Esta adaptação é feita com base no momento de ordem zero, que pode ser interpretado como a área sobre a janela de procura. Desta forma o comprimento e a altura da janela são determinados em função deste momento que consiste na soma das probabilidades de todos os pixels na região de interesse. Uma análise mais detalhada deste algoritmo pode ser encontrada em [4], [5] e [6]. O fluxograma simplificado deste algoritmo encontra-se na figura 4.

Como produto deste método resultam não só as coordenadas do pixel centróide do alvo, correspondente à localização do meanshift, mas também a informação respeitante à área da janela. Neste caso a área corresponde à superfície de uma elipse que é caracterizada pelos seus eixos maior e menor e orientação.

3.1.3 Implementação do Algoritmo

O algoritmo *camshift* foi implementado na arquitectura utilizada baseada no protocolo de comunicações *yarp*, que será mais detalhado na capítulo 7. Para permitir a selecção do objecto que se pretende seguir

Figura 5: Interface gráfico para configuração do *camshift*



e para realizar o ajuste de alguns parâmetros do camshift associados ao *threshold* da saturação dos pixeis da imagem foi criado um interface gráfico, que permite também visualizar a localização, eixo maior e eixo menor da elipse.

Neste interface é possível seleccionar o valor mínimo para a componente de saturação (S) e configurar o limite inferior e superior para o brilho (V) do sistema HSV.

A selecção dos objectos é feita seleccionando com o botão esquerdo do rato a área correspondente, sendo permitido o seguimento simultâneo de até 10 objectos. A eliminação é feita seleccionando a área a eliminar com o botão direito do rato. O aspecto desta janela pode ser visto na figura 5. Os dados resultantes do algoritmo são enviados num vector, em que a primeira posição indica o número de objectos em seguimento e as restantes posições correspondem a um identificador do objecto, das coordenadas (x,y) do pixel do seu centróide, do eixo maior e menor da elipse e do ângulo que o eixo maior faz com a horizontal.

3.2 Reconstrução 3D

Para que o robot consiga operar no mundo real é fundamental que consiga ter percepção tridimensional. A cabeça do *Baltazar*, possui duas cameras, colocadas de forma semelhante aos olhos dos seres humanos. A noção de profundidade é então obtida por combinação da informação proveniente das duas cameras. Nesta secção aborda-se a forma de determinar as coordenadas de um ponto do espaço visível em ambas as cameras, conhecendo à partida a disposição relativa dos dois sensores. A matriz de parâmetros intrínsecos da camera, que estabelece a relação entre um ponto 3D e os pixeis do sensor, não será determinada explicitamente

neste trabalho, evitando assim o passo de calibração. Os únicos aspectos tidos em conta são a resolução da imagem, a distância focal e o tamanho do pixel.

3.2.1 Controlo da Cabeça Robótica

Para permitir que o braço do robot opere num volume de trabalho mais amplo é necessário movimentar as cameras. Esta opção acarreta duas consequências, uma relacionada com a determinação das coordenadas da localização do móvel num referencial fixo e outra relacionada com a actuação nas juntas da cabeça do robot, que deverão ser actuadas de forma a que esta acompanhe o movimento do móvel. Uma vez que as cameras não estão fixas, para se obter as coordenadas de um ponto 3D num referencial fixo é necessário calcular a cinemática da cabeça. Para a actuação das juntas da cabeça não foi tomada qualquer medida em especial de forma a minimizar o erro de seguimento, uma vez que não é crítico para esta aplicação, já que é possível fazer a reconstrução de um ponto 3D sem que este esteja centrado na imagem - apenas é necessário estar visível em ambas as cameras.

Para evitar o cálculo da cinemática inversa da cabeça, optou-se por desenvolver um controlador proporcional do erro ao centro da imagem. Nesta solução as juntas de *pan* e *tilt* são então actuadas em velocidade de acordo com a localização do centróide do móvel numa das cameras. Caso seja localizado para a esquerda ou direita do centro da imagem, a junta de *pan* é actuada em velocidade de forma a tentar compensar este desvio. Se o móvel se localizar acima ou abaixo actua-se a junta de *tilt*.

Para obter melhores resultados no seguimento, a velocidade com que cada uma das juntas é actuada é função da distância ao centro, fazendo-se corresponder velocidades baixas a perturbações pequenas e velocidades mais elevadas nos casos em que o centróide se encontra mais distante do centro da imagem

Este mecanismo apesar de bastante simples permite alcançar o objectivo de apontar as cameras para a localização do móvel, aumentando desta forma o volume de trabalho.

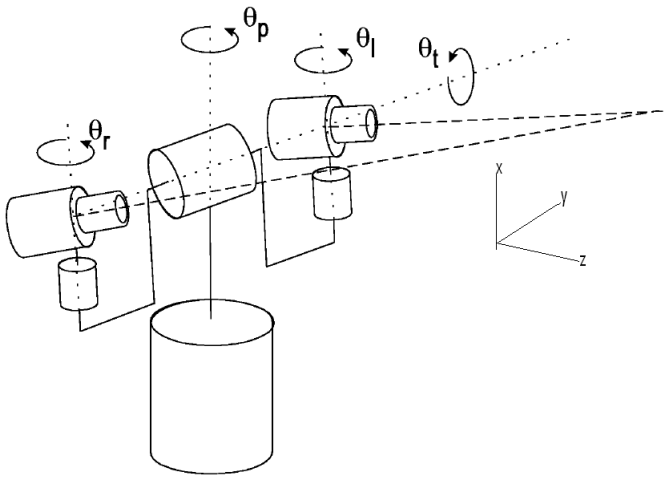
3.2.2 Reconstrução num Referencial Fixo

Afim de poder determinar as coordenadas de um objecto num referencial fixo, torna-se necessário determinar a cinemática directa da cabeça do *Baltazar*. Partindo das coordenadas no plano de imagem pretende-se obter as coordenadas num referencial fixo.

Alguns detalhes sobre a reconstrução 3D e sobre a cinemática directa da cabeça são agora apresentados. O esquema da cabeça encontra-se representado na figura 6. A expressão da equação 8, obtida por análise geométrica, relaciona as coordenadas no plano de imagem duma das cameras com as coordenadas referentes a um referencial fixo.

$$P_l = R_l(P - t_l) \tag{8}$$

Figura 6: Esquema da cabeça do *Baltazar* [extraído de [12]]



onde as matriz R_l e a translação t_l são as da equação 9.

$$R_l = \begin{bmatrix} c_p c_l - c_t s_p s_l & -s_p s_t & c_t c_l s_p + c_p s_l \\ -s_t s_l & c_t & c_l s_t \\ -c_l s_p - c_p c_t s_l & -c_p s_t & c_p c_t c_l - s_p s_l \end{bmatrix}; \quad t_l = \begin{bmatrix} -B' c_l - tY s_t s_l + tZ(-c_l s_p - c_p c_t s_l) \\ tY c_t - tZ c_p s_t \\ tY c_l s_t - B' s_l + tZ(c_p c_t c_l - s_p s_l) \end{bmatrix} \tag{9}$$

em que c_x e s_x representam respectivamente $cos\theta_x$ e $sin\theta_x$ e $B' = B/2$ corresponde a metade da *baseline*.

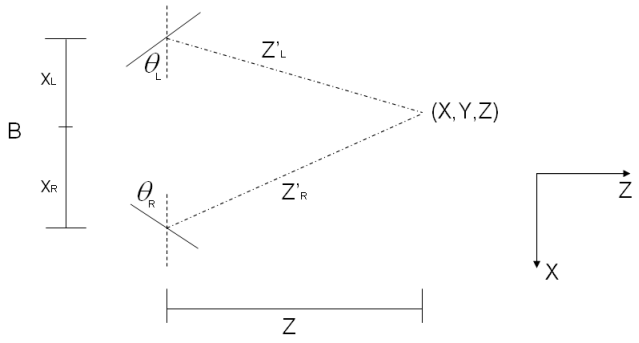
Para a reconstrução tridimensional é necessário considerar situações distintas relacionadas com a orientação relativa das cameras. O ponto 3D é calculado de forma diferente caso as cameras estejam paralelas ou com orientação arbitrária.

Para o caso de cameras paralelas, e tendo em conta o modelo de *buraco de agulha* para as cameras, o ponto 3D, num referencial alinhado com o eixo óptico da camera, é calculado de acordo com as expressões 10.

$$x = \frac{FX}{Z} \quad y = \frac{FY}{Z} \tag{10}$$

onde as coordenadas com letras minúsculas referem-se à posição na imagem e as coordenadas com letra maiúscula referem-se à posição expressa em metros. A variável F representa a distância focal e está expressa em metros. Considerando a existência de duas cameras e designando por B a distância entre o centro óptico das cameras (*baseline*), pode-se obter a coordenada Z através das equações apresentadas em (11). As

Figura 7: Esquema da reconstrução 3D



coordenadas X e Y são obtidas substituindo este valor nas expressões (10)

$$\begin{gathered} y_L = \frac{FY_L}{Z_L} \\ Y_R = Y_L - B \quad Z_L = Z_R = Z \\ y_R = \frac{FY_R}{Z_R} = F\frac{Y_L - B}{Z} \\ y_L - y_R = F\frac{B}{Z} \leftrightarrow Z = \frac{FB}{y_L - y_R} \end{gathered} \tag{11}$$

Seguidamente apresenta-se o caso em que as cameras se encontram com uma orientação arbitrária. Partindo do esquema apresentado na figura 7 pode-se escrever para a coordenada Z

$$\begin{gathered} tan\left(\theta_L + x_L/F\right) = \frac{B/2}{Z} \\ tan\left(\theta_R + x_R/F\right) = \frac{B/2}{Z} \\ Z = \frac{B}{tan(x_L/F + \theta_L) - tan(x_R/F + \theta_R)} \end{gathered} \tag{12}$$

Para o caso da coordenada X tem-se

$$\begin{gathered} tan\left(\theta_L + x_L/F\right) = \frac{X_L}{Z} \\ tan\left(\theta_R + x_R/F\right) = \frac{X_R}{Z} \\ X = -\frac{Z}{2}\left(tan\left(x_L/F + \theta_L\right) + tan\left(x_R/F + \theta_R\right)\right) \end{gathered} \tag{13}$$

Finalmente para a coordenada Y fica

$$\begin{gathered} Z'_R = \frac{Z}{cos\theta_R} \\ Z'_L = \frac{Z}{cos\theta_L} \\ Y = \frac{Zy_R}{2Fcos\theta_R} + \frac{Zy_L}{2Fcos\theta_L} \end{gathered} \tag{14}$$

Uma abordagem mais detalhada sobre a reconstrução 3D a partir de cameras não paralelas pode ser encontrada em [12]

3.3 Análise de Resultados

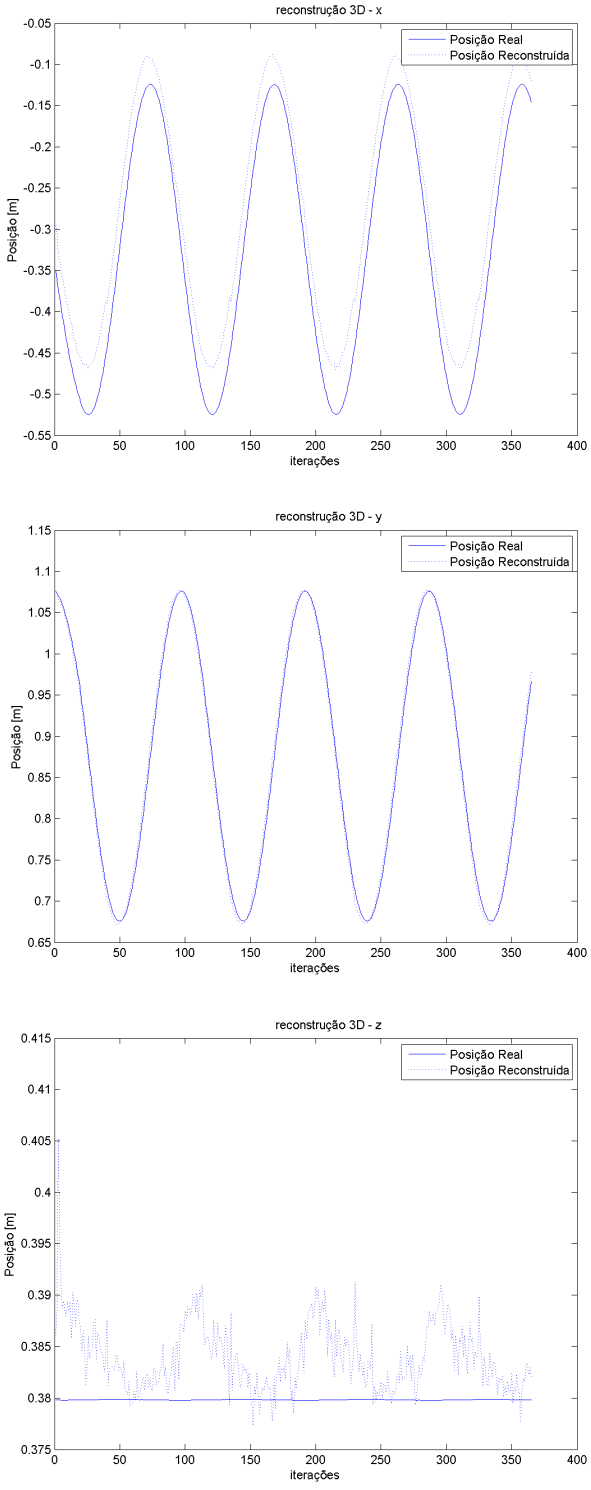
Nesta secção apresentam-se alguns resultados obtidos no simulador utilizando os algoritmos de *tracking* e de reconstrução tridimensional apresentados anteriormente. Foram feitos testes com diversas cores e saturações afim de testar a capacidade do *camshift*, tendo-se verificado que os melhores resultados correspondem a cores mais saturadas, uma vez que o histograma esta a ser calculado apenas para a componente *hue* do espaço de cores HSV. Verificou-se também que o desempenho do algoritmo piora com utilização de cores mais escuras. Isto acontece porque ao diminuir a saturação, a área do cone diminui, tornando a componente *hue* menos imune ao ruído.

Apresentam-se agora alguns resultados obtidos em simulação que permitem avaliar o erro de *tracking*. O movimento simulado é suficientemente lento para garantir que o objecto não sai da área de procura do *camshift* na transição entre duas imagens consecutivas.

Os gráficos com a evolução da posição real do móvel e a posição reconstruída a partir das cameras apresentam-se na figura 8.

O erro quadrático médio em cada uma das coordenadas x, y, z é respectivamente $4cm$, $0.7cm$ e $0.7cm$. O erro segundo x é maior devido essencialmente a dois factores: erros cometidos na transformação de coordenadas e falta de sincronismo entre a leitura da posição angular das juntas da cabeça e a captação da imagem.

Figura 8: Erros de seguimento segundo as coordenadas (x, y, z)



4 Modelos de Movimento

O objectivo primordial deste projecto consiste em predizer a posição de um corpo móvel com base na informação adquirida até um dado instante, a fim de permitir posicionar o braço de um robot atempadamente na trajectória do móvel. Uma hipótese para realizar a predição passa por criar um modelo para o movimento do móvel e utiliza-lo para determinação da posição do corpo num instante futuro. O factor decisivo para o seguimento de objectos móveis é fazer uma boa estimativa das trajectórias do alvo. Apesar de a maioria dos alvos não serem pontuais e sem contornos estas hipóteses são normalmente assumidas pela quase totalidade dos modelos de movimento. Quase todos os sistemas de seguimento baseiam-se no principio de existir uma relação matemática entre o movimento do alvo e as observações realizadas.

Existem várias hipóteses para a criação de modelos. Pode-se optar pela construção de um modelo 2D, baseado na evolução do móvel no plano de imagem ou noutro qualquer espaço, ou optar por criar modelos 3D baseados na evolução da posição do móvel no volume de trabalho. A criação de modelos 3D é de longe mais simples e intuitiva, uma vez que não comporta todos os problemas associados à representação do espaço 3D num plano. Dentro do campo dos modelos 3D existe um conjunto de possibilidades para a sua construção, assentando normalmente em hipóteses para o tipo de movimento do corpo, nomeadamente movimentos a velocidade constante, acelerados, circulares, periódicos entre outros. Como não se pretende impor qualquer tipo de restrição ao tipo de movimentos a operar, é necessário implementar modelos para diferentes tipos de movimento, sendo a predição realizada com base no desempenho que cada um dos modelos revelou em relação às posições por onde o móvel tem passado.

Nas secções seguintes apresentam-se os detalhes relacionados com a predição da posição, sendo apresentado o diagrama do sistema de predição, assim como todos os modelos utilizados. São ainda apresentados alguns resultados obtidos por simulação.

4.1 Formulação do Problema

De uma forma geral, um modelo discreto linear com período de amostragem T pode ser descrito pelo sistema de equações 15, onde $u(k)$,$y(k)$ e $x(k)$ representam, respectivamente a entrada, a saída e o vector de estado do sistema.

$$\begin{aligned} x(k) &= Ax(k-1) + Bu(k-1) \\ y(k) &= Cx(k-1) + Du(k-1) \end{aligned} \tag{15}$$

No problema presente o vector de estado x representa a posição do alvo e suas derivadas, e u representa possíveis manobras do seu movimento. Uma vez que não é possível saber a posição exacta do móvel, nem é possível criar um modelo que represente fielmente a evolução do estado, é necessário acrescentar as componentes de ruído às equações.

As variáveis aleatórias $w(k)$ e $\mu(k)$ representam o ruído do processo e do sensor respectivamente e assumem-

se independentes, de média nula e variância Q e R.

Desta forma as equações que descrevem um processo discreto, linear e invariante no tempo são:

$$\begin{aligned} x(k) &= Ax(k-1) + Bu(k-1) + Gw(k-1) \\ y(k) &= Cx(k-1) + Du(k-1) + H\mu(k) \\ f(w) &\sim N(0, Q) \\ f(\mu) &\sim N(0, R) \end{aligned} \tag{16}$$

Para um processo de dimensão n, com l entradas e m saídas, a matriz A $n \times n$ da equação às diferenças, relaciona o estado no instante $k-1$ com o estado actual. A matriz B $n \times l$ permite integrar no modelo uma entrada no sistema, enquanto que a matriz C $m \times n$ relaciona o estado com a saída do sistema. A matriz D estabelece a forma como a entrada afecta directamente a saída, mas para todos os modelos desenvolvidos será nula.

As matrizes G e H são responsáveis pela introdução do ruído nas equações de estado. Na ausência de ruído, bastaria utilizar as equações(15) para a propagação do vector de estado. No entanto, assumindo a presença de ruído tal já não é possível, sendo necessário utilizar outras ferramentas.

4.1.1 Estimação por filtragem de Kalman

O filtro de Kalman é uma estratégia muito utilizada neste domínio, não só por conduzir à solução óptima, mas também devido ao seu reduzido peso computacional.

O objectivo deste filtro é obter uma estimativa *a posteriori* com base numa combinação linear da estimativa *a priori* realizada com base nos modelos expressos pela equação 15. O erro *a priori* e *a posteriori* são definidos de acordo com a equação 17, onde $\hat{x}^-(k)$ representa a estimativa *a priori* e $\hat{x}(k)$ representa a estimativa dada a leitura $z(k)$.

$$\begin{aligned} e^-(k) &= x(k) - \hat{x}^-(k) \\ e(k) &= x(k) - \hat{x}(k) \end{aligned} \tag{17}$$

A nova equação para a evolução do estado passa a ser

$$\hat{x}(k) = \hat{x}^-(k) + K\left(z(k) - C\hat{x}^-(k)\right) \tag{18}$$

O ganho de kalman K é responsável por introduzir, de forma devidamente pesada, a discrepância entre a estimação e a medida actual na equação de propagação de estado. Este ganho é determinado de forma a minimizar a covariância do erro *a posteriori*, podendo ser obtido substituindo (18) em (17), derivando e igualando a zero.

Seguindo este procedimento, chega-se a conclusão que o ganho de Kalman é dado por

$$K(k) = P^-(k)C^T\left(CP^-(k)C^T + R\right)^{-1} \tag{19}$$

onde $P^-(k)$ representa a covariância do erro *a priori*.
Como se pode verificar, pela expressão do ganho de Kalman, sempre que a covariância do erro de medida se aproxima de 0, o filtro atribui mais peso à medida, desvalorizando a estimativa predita do estado. Por outro lado, se a matriz de covariância do erro *a priori* tender para 0, o filtro passa a atribuir mais peso ao estado predito em vez de englobar as medidas realizadas.

O filtro de Kalman utiliza intrinsecamente um esquema de *feedback*. Para um dado instante faz a estimativa do estado do processo e depois obtém *feedback* desta estimativa através da medida obtida.
Desta forma é possível separar as equações do filtro em dois grupos: um que realiza a actualização temporal do estado e da covariância do erro (predição) e outro responsável pelo *feedback*, que incorpora uma nova medida na estimativa *a priori* para obter uma estimativa *a posteriori* mais realista (correcção).

A implementação do filtro de Kalman é bastante simples. Partindo das estimativas iniciais do estado e da covariância do erro de estimação entra-se num ciclo em que se realiza o passo de predição de acordo com 20 e o passo de correcção como expresso em 21.

$$\begin{gathered}\text{Equações de actualização temporal (Predição)}\\ \hat{x}^-(k) = A\hat{x}(k-1) + Bu(k-1)\\ P^-(k) = AP(k-1) + Q\end{gathered} \tag{20}$$

$$\begin{gathered}\text{Equações de actualização da medição (Correcção)}\\ K(k) = P^-(k)C^T\left(CP^-(k)C^T + R\right)^{-1}\\ \hat{x}(k) = \hat{x}^-(k) + K(k)\left(z(k) - C\hat{x}^-(k)\right)\\ P(k) = \left(I - K(k)C\right)P^-(k)\end{gathered} \tag{21}$$

Como se pode verificar nas equações (20) e (21), se as covariâncias do ruído do processo e da medida R e Q forem constantes, o ganho de kalman vai acabar por estabilizar depois de realizadas algumas iterações, pelo que é possível determiná-lo, desde que sejam conhecidas as covariâncias. Um texto introdutório ao filtro de kalman pode ser encontrado em [13], enquanto que em [14], [15] e [16] se pode ter uma abordagem mais profunda a esta ferramenta.

4.2 Seguimento com múltiplos Modelos

Os diferentes modelos 3D que são utilizados para a predição da posição do móvel consistem então em difrentes matrizes A, B, C, D, Q e R. A saída de cada modelo corresponde às posições do vector de estado que representam as coordenadas (x, y, z) de um ponto num referencial 3D.

Para a estimativa do vector de estado dos modelos utiliza-se o filtro de kalman, a fim de conseguir ter um melhor desempenho do sistema na presença de ruído.

Para a realização da predição foram desenvolvidos sete modelos que operam em paralelo, nomeadamente um modelo de velocidade constante, considerando que a aceleração é ruído branco, três modelos de aceleração constante: dois deles baseados no modelo de Wiener e um outro no modelo de Singer, um modelo para movimentos circulares, um modelo para movimentos curvilíneos e um modelo concebido para movimentos balísticos incorporando a possibilidade de colisão com uma superfície horizontal conhecida à partida. Cada um destes modelos será detalhadamente descrito nas secções seguintes.

Para cada um deles serão apresentadas as matrizes A, C, Q e R necessárias para a estimativa da posição utilizando o filtro de kalman. O vector de estado $x = [x\ \dot{x}\ \ddot{x}\ y\ \dot{y}\ \ddot{y}\ z\ \dot{z}\ \ddot{z}]^T$ é utilizado em todos os modelos, à excepção do modelo de velocidade constante e do modelo balístico. Para estes casos não existe a componente da aceleração, sendo utilizado o vector $x = [x\ \dot{x}\ y\ \dot{y}\ z\ \dot{z}]^T$.

Em todos os modelos apresentados, a excepção do modelo balístico, a matriz B é nula de dimensão $[9\ \times\ 1]$ ou $[6\ \times\ 1]$, consoante o modelo tenha ou não a componente da aceleração no vector de estado.

A estratégia seguida para o desenvolvimento dos modelos é escrever as equações do espaço de estados em tempo contínuo e fazer a sua discretização. Esta discretização é feita assumindo a existência de retentores de ordem zero (ZOH) nas entradas do sistema, isto é, assumindo que a entrada do sistema é constante durante o período de amostragem.

Partindo da integração das equações do modelo de estado contínuas,

$$x(t) = e^{A_c(t-t_0)}x(t_0) + \int_{t_0}^{t} e^{A_c(t-\tau)}B_c u(\tau)d\tau \tag{22}$$

Fazendo $t_0 = kT$ e $t = t_0 + T$, resulta da equação 22

$$x((k+1)T) = e^{A_c T}x(kT) + \int_{kT}^{(k+1)T} e^{A_c((k+1)T-\tau)}B_c u(\tau)d\tau \tag{23}$$

Assumindo agora a condição do ZOH $u(\tau)$ é constante e a equação 23 pode-se escrever na forma

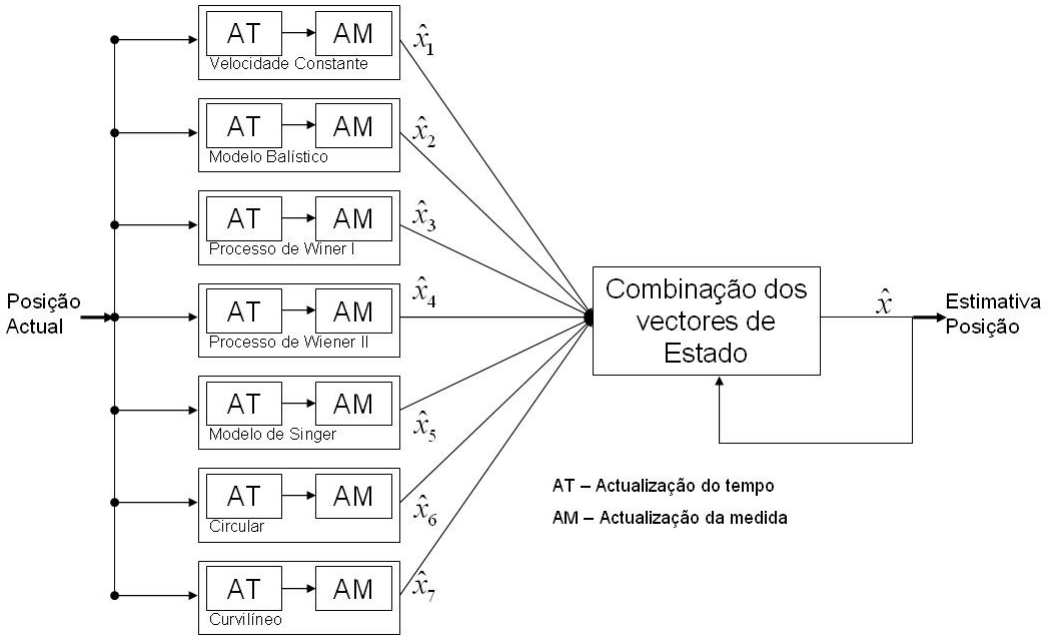
$$\begin{gathered} x(k+1) = Ax(k) + Bu(k) \\ A = e^{A_c T} \quad B = \left(\int_0^T e^{A\tau}d\tau\right) B_c \end{gathered} \tag{24}$$

Esta operação pode ser realizada através do comando *c2d* do *Matlab*.

Na figura 9 apresenta-se o esquema da arquitectura utilizada que será também mais aprofundada nas secções seguintes.

Para o caso de movimentos periódicos foi ainda desenvolvido um outro modelo baseado na técnica da correlação, que será descrito mais à frente uma vez que não utiliza às técnicas do espaço de estados apresentadas aqui.

Figura 9: Arquitectura do Preditor de Posição 3D



4.2.1 Movimentos de Velocidade Constante - Acelerações Mínimas

Este modelo considera um corpo a deslocar-se num referencial inercial a velocidade constante, também designado por movimento uniforme. Como referido anteriormente, o vector de estado utilizado neste modelo é $x = [x\ \dot{x}\ y\ \dot{y}\ z\ \dot{z}]^T$. As equações discretas da dinâmica deste modelo, considerando a presença de uma pequena aceleração aleatória em cada coordenada são:

$$\begin{aligned} p^{(i)}(k+1) &= p^{(i)}(k) + Tv^{(i)}(k) + \tfrac{T^2}{2}w^{(i)}(k) \\ v^{(i)}(k+1) &= v^{(i)}(k) + Tw^{(i)}(k) \end{aligned} \tag{25}$$

onde p denota a posição, v a velocidade e T o período de amostragem. O índice (i) representa a coordenada i, enquanto que a variável $w^{(i)}(k)$ representa a aceleração aleatória no instante k, segundo a coordenada i. Utilizando a notação matricial, o sistema 25 pode ser reescrito como

$$\begin{aligned} x^{(i)}(k+1) &= A^{(i)}x^{(i)}(k) + G^{(i)}w^{(i)}(k) \\ z^{(i)}(k) &= C^{(i)}x^{(i)}(k) + H^{(i)}\mu^{(i)}(k) \end{aligned} \tag{26}$$

em que

$$A^{(i)} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \quad G^{(i)} = \begin{bmatrix} \frac{T^2}{2} \\ T \end{bmatrix} \quad C^{(i)} = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad H^{(i)} = [1] \tag{27}$$

O modelo englobando as três coordenadas, assumindo que o ruído é independente entre elas pode ser escrito como:

$$\begin{aligned} x(k+1) &= Ax(k) + Gw(k) \\ z(k) &= Cx(k) + H\mu(k) \end{aligned} \tag{28}$$

com

$$\begin{aligned} A &= diag(A^{(1)}, A^{(2)}, A^{(3)}) \\ G &= diag(G^{(1)}, G^{(2)}, G^{(3)}) \\ C &= diag(C^{(1)}, C^{(2)}, C^{(3)}) \end{aligned} \tag{29}$$

$$R = diag(\sigma_{\mu 1}^2 H^{(1)}, \sigma_{\mu 2}^2 H^{(2)}, \sigma_{\mu 3}^2 H^{(3)})$$

Onde R representa a matriz de covariância do ruído da medida e $\sigma_{\mu 1}^2$, $\sigma_{\mu 2}^2$ e $\sigma_{\mu 3}^2$ correspondem ao ruído da medida em cada uma das direcções do referencial cartesiano, comportando assim a possibilidade de ter erros diferentes segundo as várias direcções. Esta funcionalidade poderá ser útil por exemplo, para incorporar maior ruído para a profundidade de um ponto obtido a partir da informação proveniente das duas cameras do robot.

Para poder utilizar o filtro de kalman falta apenas determinar a matriz de covariância do ruído do modelo Q, que é dada por 30.

$$\begin{aligned} Q^{(i)} &= cov\left(G^{(i)} w^{(i)}(k)\right) = var\ w^{(i)}(k) \times \begin{bmatrix} \frac{T^4}{4} & \frac{T^3}{2} \\ \frac{T^3}{2} & T^2 \end{bmatrix} \\ Q &= diag(\sigma_{w1}^2 Q^{(1)}, \sigma_{w2}^2 Q^{(2)}, \sigma_{w3}^2 Q^{(3)}) \end{aligned} \tag{30}$$

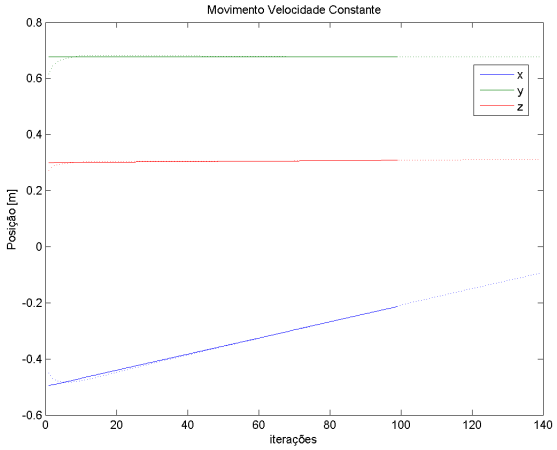
No gráfico 10 apresentam-se os resultados realizados em simulação no *Webots*, após ter calibrado devidamente os ruídos do modelo e da medida.

Neste gráfico apresenta-se a evolução de cada uma das coordenadas do móvel lidas directamente do simulador (linha a cheio) e resultantes do modelo (linha ponteada).

Como se pode verificar, nos primeiros instantes observa-se uma fase de adaptação do filtro, em que a saída deste converge rapidamente para a posição real (cerca de 5 iterações). Nos instantes seguintes verifica-se que o modelo segue bastante bem a posição do objecto, tendo um erro quadrático médio de $0.89cm$. Nos últimos instantes da simulação apresenta-se o resultado da iteração sucessiva da equação de actualização do estado. Este último troço corresponde, desta forma, à predição da posição do móvel até 40 instantes de tempo à frente.

Uma vez que a hipótese de movimento a velocidade constante é de facto verificada no deslocamento do móvel, este modelo é capaz de produzir uma predição bastante plausível para a posição do móvel. Fica também

Figura 10: Predição com modelo de velocidade constante A contínuo apresenta-se a posição real do objecto e a pontilhado apresenta-se a posição predicta



aqui presente que, caso a condição de velocidade constante não seja de facto verdadeira, o modelo falha na predição, apresentando, inclusive, alguma dificuldade em acompanhar o movimento do alvo, como se pode ver na imagem da figura 11.

4.2.2 Movimentos de Aceleração Constante - Processo de Wiener

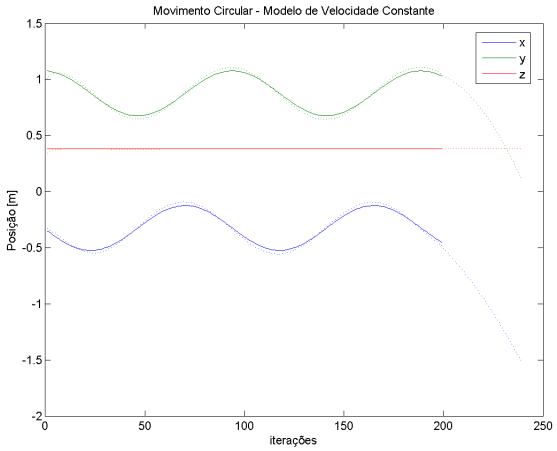
Este modelo de aceleração constante admite que a aceleração é um processo de Wiener, ou movimento Brawniano. Este processo é utilizado em muitas situações reais, nas mais diversas áreas do conhecimento, como a física a matemática ou a economia e consiste num processo estocástico de *càdlàg* com incrementos estacionários e independentes. Este modelo assume portanto que a aceleração do alvo é um processo com incrementos independentes.

É frequente encontrar duas versões diferentes para o processo de Wiener. Uma assume que a derivada da aceleração é ruído branco, conhecido como modelo de aceleração do tipo impulsiva, enquanto que a outra versão designada por Modelo de aceleração em sequência de Wiener admite que os incrementos na aceleração constituem um processo independente de ruído branco. O vector de estado utilizado em ambas as versões deste modelo é $x = [x\ \dot{x}\ \ddot{x}\ y\ \dot{y}\ \ddot{y}\ z\ \dot{z}\ \ddot{z}]^T$.

Para o primeiro caso a versão contínua do modelo é, para a coordenada i:

$$\dot{x}^{(i)}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x^{(i)}(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} w^{(i)}(t) \tag{31}$$

Figura 11: Predição de trajectória circular com modelo de velocidade constante



cujo equivalente discreto é:

$$x^{(i)}(k+1) = A^{(i)}x^{(i)}(k) + G^{(i)}w^{(i)}(k)$$

$$z^{(i)}(k) = C^{(i)}x^{(i)}(k) + H^{(i)}\mu^{(i)}(k) \tag{32}$$

$$A(i) = \begin{bmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \quad G(i) = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad C^{(i)} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \quad H^{(i)} = [1]$$

com $x^{(i)} = \left[x^{(i)}\ \dot{x}^{(i)}\right]$ representa o vector de estado correspondente à coordenada i.

A matriz de covariância do ruído do modelo é:

$$Q^{(i)} = cov\left(w^{(i)}(k)\right) = S_w^{(i)} \times \begin{bmatrix} \frac{T^5}{20} & \frac{T^4}{8} & \frac{T^3}{6} \\ \frac{T^4}{8} & \frac{T^3}{3} & \frac{T^2}{2} \\ \frac{T^3}{6} & \frac{T^2}{2} & T \end{bmatrix} \tag{33}$$

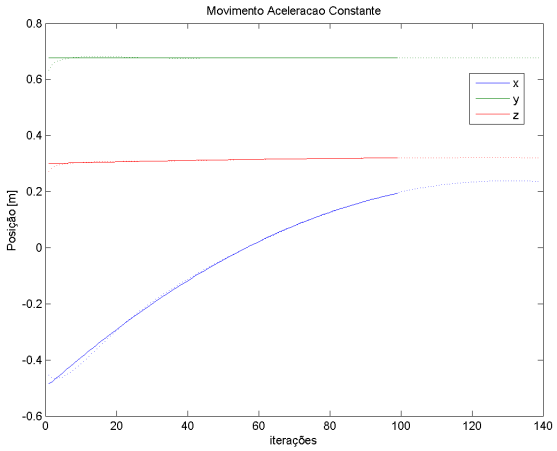
$$Q = diag(S_w^1 Q^{(1)}, S_w^2 Q^{(2)}, S_w^3 Q^{(3)})$$

onde $S_w^{(i)}$ representa a densidade espectral de potência na coordenada i.

A densidade espectral de potência pode ser facilmente obtida, utilizando o teorema de Wiener-Khinchin, através da transformada de Fourier da função de autocorrelação do sinal.

Uma vez que se assume que o ruído é branco, a densidade espectral de potência é constante para todas as frequências. Este valor foi escolhido de forma a optimizar o desempenho do modelo.

Figura 12: Predição da posição com modelo de aceleração constante: Processo de Wiener



As matrizes A e R englobando as três coordenas são obtidas por:

$$A = diag(A^{(1)}, A^{(2)}, A^{(3)})$$

$$C = diag(C^{(1)}, C^{(2)}, C^{(3)}) \tag{34}$$

$$R = diag(\sigma^2_{\mu 1} H^{(1)}, \sigma^2_{\mu 2} H^{(2)}, \sigma^2_{\mu 3} H^{(3)})$$

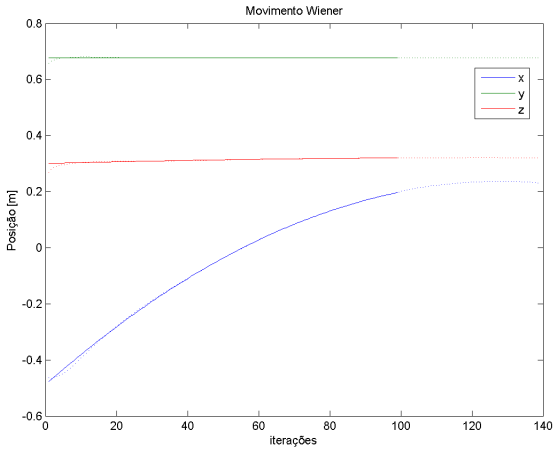
Para este modelo a matriz B é nula de dimensão $[9 \times 1]$.

Seguidamente apresentam-se alguns ensaios realizados para móveis com aceleração constante. Como já ficou evidente na secção anterior é inútil tentar utilizar exclusivamente um modelo para realizar predição a longo horizonte da posição se este não corresponder ao tipo de movimento efectivamente presente no móvel. Contudo, incorporar um modelo cuja hipótese não se verifique num esquema de modelos paralelos continua a trazer vantagens, na medida em que, na realidade o movimento do móvel apresenta sempre algum ruído, que pode sempre ser diminuído juntando a informação proveniente de diversas fontes, como se verá mais à frente.

No gráfico da figura 12 é possível verificar a fase inicial de convergência do filtro, apresentando a partir de então uma boa resposta no que respeita ao seguimento da posição do móvel, com um erro quadrático médio de $0.47cm$. Quanto à predição da posição, verifica-se, também aqui um bom desempenho do modelo, apresentando uma evolução parabólica típica dos movimentos uniformemente acelerados.

Para a versão em que os incrementos na aceleração são considerados independentes, as únicas alterações reflectem-se nas matrizes G e Q, que para este caso são:

Figura 13: Predição da posição com modelo de aceleração com incrementos independentes



$$G^{(i)} = \begin{bmatrix} \frac{T^2}{2} \\ T \\ 1 \end{bmatrix} \quad G = diag\left(G^{(1)}, G^{(2)}, G^{(3)}\right)$$

$$Q^{(i)} = cov(G^{(i)} w^{(i)}(k)) = var(w^{(i)}(k)) \times \begin{bmatrix} \frac{T^4}{4} & \frac{T^3}{2} & \frac{T^2}{2} \\ \frac{T^3}{2} & \frac{T^2}{2} & T \\ \frac{T^2}{2} & T & 1 \end{bmatrix} \tag{35}$$

$$Q = diag(\sigma_{w1}^2 Q^{(1)}, \sigma_{w2}^2 Q^{(2)}, \sigma_{w3}^2 Q^{(3)})$$

Os resultados para esta versão apresentam-se na figura 13. Aqui é possível verificar, em comparação com a situação anterior, não só uma convergência mais rápida, como também um melhor ajustamento entre a posição resultante do modelo e a posição real do móvel, tendo-se alcançado um erro quadrático médio de $0.43cm$.

4.2.3 Movimentos de Aceleração Constante - Modelo de Singer

Enquanto que no ruído branco cada amostra está "isolada"no tempo, uma vez que o valor num dados instante está completamente desacoplado do valor num qualquer outro instante de tempo, num processo de Markov existe uma ligação entre o valor do processo num dado instante e o valor em instantes vizinhos. Um processo diz-se de Markov estacionário se para $t > t_1 > 0$ a distribuição da probabilidade condicionada no instante t dada a história do processo até ao instante t_1 inclusive, depende apenas do estado do processo no instante t_1. Isto é, o processo no instante t é independente do passado até ao instante t_1, dado o estado do processo em t_1. Um processo de Markov de ordem N, em vez de depender apenas do estado de um único instante

depende do estado de N instantes consecutivos.

O modelo que agora se descreve foi o primeiro modelo a caracterizar a aceleração do alvo como uma grandeza correlacionada temporalmente, isto é, sem a considerar ruído branco. O modelo de Singer assume que a derivada da aceleração do alvo é um processo de Markov de média nula e ordem 1. A autocorrelação e potência espectral da aceleração são

$$R_a(\tau) = E\left[a(t+\tau)a(t)\right] = \sigma^2 e^{-\alpha|\tau|} \tag{36}$$

Uma vez que um processo de Markov com uma potência espectral racional é equivalente ao SLIT correspondente excitado apenas por ruído branco, pode-se escrever que

$$\dot{a}(t) = -\alpha(t) + w(t), \alpha > 0 \tag{37}$$

Considerando apenas a coordenada i e representando o vector de estado correspondente por $x^{(i)} = \left[x^{(i)}\ \dot{x}^{(i)}\right]$ pode-se escrever, utilizando a notação matricial:

$$\dot{x}^{(i)}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -\alpha \end{bmatrix} x^{(i)}(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} w^{(i)}(t) \tag{38}$$

O seu equivalente discreto é:

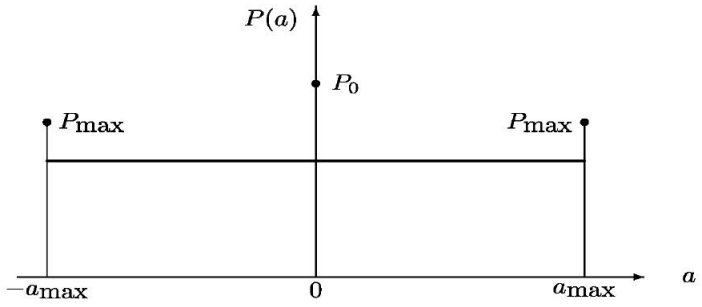
$$\begin{gathered} x^{(i)}(k+1) = A(i)x(i)(k) + w(i)(k) \\ z^{(i)}(k) = C^{(i)}x^{(i)}(k) + H^{(i)}\mu^{(i)}(k) \\ A^{(i)} = \begin{bmatrix} 1 & T & \frac{\alpha T - 1 + e^{-\alpha T}}{\alpha^2} \\ 0 & 1 & \frac{1-e^{-\alpha T}}{\alpha} \\ 0 & 0 & e^{-\alpha T} \end{bmatrix} \quad C^{(i)} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \quad H^{(i)} = [1] \end{gathered} \tag{39}$$

resultando para as matrizes A, C, R

$$\begin{gathered} A = diag(A^{(1)}, A^{(2)}, A^{(3)}) \\ C = diag(C^{(1)}, C^{(2)}, C^{(3)}) \\ R = diag(\sigma_{\mu 1}^2 H^{(1)}, \sigma_{\mu 2}^2 H^{(2)}, \sigma_{\mu 3}^2 H^{(3)}) \end{gathered} \tag{40}$$

Como se pode verificar à medida que α diminui, a matriz A do modelo de Singer, reduz-se à primeira versão do modelo de Wiener, onde a aceleração é considerada impulsiva. Por outro lado quando α aumenta,

Figura 14: Função densidade de probabilidade da aceleração



o modelo de Singer tende a tornar-se um modelo de velocidade constante, em que a aceleração é apenas ruído. Desta forma a escolha do parâmetro α perimte posicionar o modelo de Singer entre um modelo de velocidade constante e um modelo de aceleração constante.

A matriz de covariância do ruído do modelo Q que se pode encontrar em [17] é demasiado trabalhosa para introduzir neste texto. Para garantir a sua inteligibilidade basta referir que, à semelhança da matriz A, a matriz de covariância do ruído do modelo é função do parâmetro α e do período de amostragem T. De facto o sucesso deste modelo está criticamente dependente da escolha conveniente dos parâmetros α e σ^2. O parâmetro α é o inverso da constante de tempo τ, que representa a duração da dependência entre instantes de tempo. Por sua vez σ^2 representa a variância instantânea da aceleração. De acordo com o proposto por Singer, a função densidade de probabilidade da aceleração do alvo deve ser função de três parâmetros: a probabilidade do alvo se mover sem aceleração, P_0, o valor máximo possível para a aceleração, a_{max} e a probabilidade de a_{max} acontecer, P_{max}. Para os restantes valores possíveis de a a sua função densidade de probabilidade é constante. A representação desta função encontra-se na figura 14

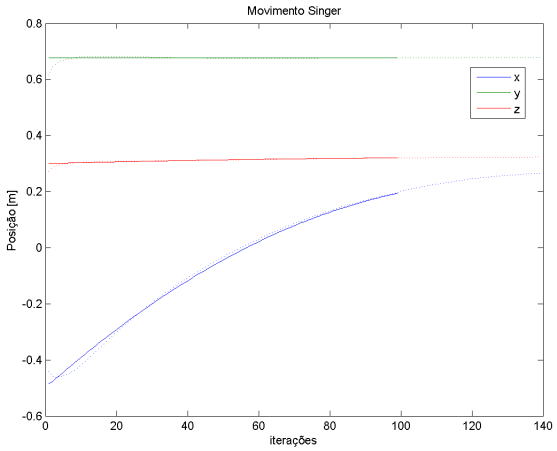
Considerando esta distribuição de probabilidades, pode-se obter

$$\sigma^2 = \frac{a_{max}^2}{3}\left(1 + 4P_{max} - P_0\right) \tag{41}$$

Este modelo apesar de tudo é um modelo *a priori* uma vez que não utiliza a informação da posição do alvo para obter uma estimativa dos seus parâmetros de desenho, podendo-se no entanto alterá-lo de forma a torná-lo adaptativo.

Seguidamente apresentam-se alguns resultados para este modelo simulando no *webots* alvos com movimentos de aceleração constante. No gráfico da figura 15 é possível ver, comparando com o modelo de Wiener, um pior desempenho por parte do modelo de Singer, apesar da sua maior complexidade. Esta aparente contradição deve-se ao facto deste modelo estar dependente de alguns parâmetros que não são estimados em função do movimento do alvo, tornando a sua resposta menos ajustável. O erro quadrático médio foi $0.49cm$ Desta forma, o modelo de Wiener torna-se mais proveitoso, a menos que se opte por incorporar um esquema

Figura 15: Predição da posição com modelo de aceleração constante: modelo de Singer



para estimação dos referidos parâmetros no modelo de Singer.

4.2.4 Movimentos Circulares

Este modelo é concebido para alvos que se desloquem a velocidade angular constante numa trajectória circular no espaço 3D. Para estes casos em que $\dot{v} = 0$, sabe-se que a aceleração é perpendicular à velocidade ($a \perp v$). Designando a velocidade angular do alvo por Ω, esta condição é equivalente a

$$a = \Omega \times v \tag{42}$$

Uma vez que Ω é em geral desconhecido, uma forma simples de ultrapassar o problema passa por acrescentar esta variável ao vector de estado do sistema, ficando $x = [x\ \dot{x}\ y\ \dot{y}\ z\ \dot{z}\ \Omega_x\ \Omega_y\ \Omega_z]$. Assim, a equação anterior pode ser escrita, na versão contínua como:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{bmatrix} \times \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \Omega_y \dot{z} - \Omega_z \dot{y} \\ \Omega_z \dot{x} - \Omega_x \dot{z} \\ \Omega_x \dot{y} - \Omega_y \dot{x} \end{bmatrix} \tag{43}$$

Uma vez que Ω é praticamente constante, a sua derivada pode ser considerada ruído branco. Nestas condições, as equações do modelo contínuo são:

$$\dot{x}^{(i)}(t) = \begin{bmatrix} 0 & I_{3\ \times\ 3} & 0 \\ 0 & M_\Omega & 0 \\ 0 & 0 & 0 \end{bmatrix} x^{(i)}(t) + \begin{bmatrix} 0 \\ 0 \\ I_{3\ \times\ 3} \end{bmatrix} w^{(i)}(t) \tag{44}$$

onde M_Ω representa a matriz de produto externo para o vector Ω dada por

$$M_\Omega = \begin{bmatrix} 0 & -\Omega_z & \Omega_y \\ \Omega_z & 0 & -\Omega_x \\ -\Omega_y & \Omega_x & 0 \end{bmatrix} \tag{45}$$

Desta forma o modelo contínuo é não linear, uma vez que a matriz apresentada em 44 depende de Ω, que pertence também ao vector de estado. Para evitar esta não linearidade é possível acrescentar outras condições ao desenho do modelo, que consistem em assumir que o movimento circular a velocidade constante está contido numa superfície planar e que $\dot{\Omega} = 0$. Esta condição é equivalente a dizer que $\Omega \perp v$. Dadas as condições, diferenciando 42 resulta:

$$\dot{a} = \Omega \times a = \Omega \times (\Omega \times v) = -\omega^2 v \tag{46}$$

em que ω representa o módulo da velocidade angular. Assim, adicionando a componente de ruído ao modelo, pode-se escrever:

$$\dot{a} = -\omega^2 v + w \tag{47}$$

Considerando agora o vector de estado $x = [x\ \dot{x}\ \ddot{x}\ y\ \dot{y}\ \ddot{y}\ z\ \dot{z}\ \ddot{z}]^T$ a versão deste modelo já linearizada é:

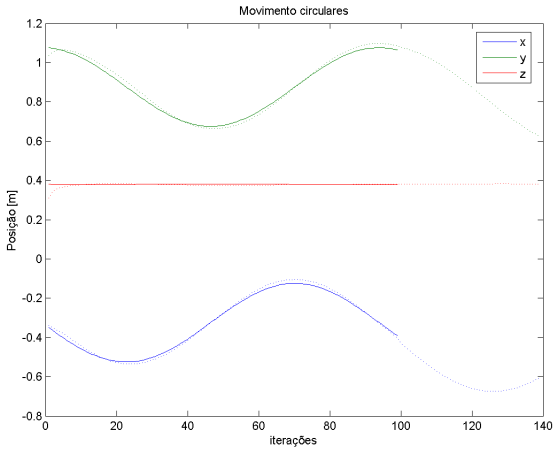
$$\dot{x}^{(i)}(t) = \begin{bmatrix} 0 & I_{3\times3} & 0 \\ 0 & 0 & I_{3\times3} \\ 0 & \omega^2 I_{3\times3} & 0 \end{bmatrix} x^{(i)}(t) + \begin{bmatrix} 0 \\ 0 \\ I_{3\times3} \end{bmatrix} w^{(i)}(t) \tag{48}$$

O seu equivalente discreto é:

$$\begin{gathered} x^{(i)}(k+1) = A^{(i)} x^{(i)}(k) + w^{(i)}(k) \\ \\ z^{(i)}(k) = C^{(i)} x^{(i)}(k) + H^{(i)} \mu^{(i)}(k) \\ \\ A^{(i)} = \begin{bmatrix} 1 & \frac{sin(\omega T)}{\omega} & \frac{1-cos(\omega T)}{\omega^2} \\ 0 & cos(\omega T) & \frac{sin(\omega T)}{\omega} \\ 0 & -\omega sin(\omega T) & cos(\omega T) \end{bmatrix} \quad C^{(i)} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} H^{(i)} = [1] \end{gathered} \tag{49}$$

As matrizes A, C, R e Q necessárias para a implementação com filtro de kalman são então:

Figura 16: Predição da posição com modelo de movimento circular



$$
\begin{aligned}
A &= diag(A^{(1)}, A^{(2)}, A^{(3)}) \\
C &= diag(C^{(1)}, C^{(2)}, C^{(3)}) \\
R &= diag(\sigma_{\mu 1}^2 H^{(1)}, \sigma_{\mu 2}^2 H^{(2)}, \sigma_{\mu}^2 H^{(3)}) \\
Q &= diag(S_w^{(1)} Q^{(1)}, S_w^{(2)} Q^{(2)}, S_w^{(3)} Q^{(3)})
\end{aligned}
\tag{50}
$$

$$
Q^{(i)} = cov(w^{(i)}(k)) = \begin{bmatrix}
\frac{6\omega T - 8sin(\omega T) + sin(2\omega T)}{4\omega^5} & \frac{2sin^4(\omega T/2}{\omega^4} & \frac{-2\omega T + 4sin(\omega T) - sin(2\omega T)}{4\omega^3} \\
\frac{2sin^4(\omega T/2)}{\omega^4} & \frac{2\omega T - sin(2\omega T)}{4\omega^3} & \frac{sin^2(\omega T)}{2\omega^2} \\
\frac{-2\omega T + 4sin(\omega T) - sin(2\omega T)}{4\omega^3} & \frac{sin^2(\omega T)}{2\omega^2} & \frac{2\omega T + sin(2\omega T)}{4\omega}
\end{bmatrix}
$$

onde $S_w^{(i)}$ representa a densidade espectral de potência em cada uma das coordenadas.

Neste modelo existe um acoplamento entre coordenadas através da velocidade de rotação ω. Neste trabalho a velocidade de rotação constitui um parâmetro de desenho do modelo, tendo portanto que ser ajustada *a priori* ao movimento do alvo. Para ultrapassar esta limitação, poderia acrescentar-se a possibilidade de estimar este parâmetro juntando-o ao vector de estado, contudo assim voltaria-se a ter um modelo altamente não linear.

Alguns dos resultados obtidos com este modelo para movimentos circulares encontram-se no gráfico 16.

Este modelo, apesar de mais complexo que os anteriores de velocidade e aceleração constante, continua a apresentar uma rápida convergência para a posição real do alvo. No que respeita à sua adaptação ao

movimento verifica-se um pior desempenho, apresentado um erro quadrático médio de $1.48cm$. Este resultado pode ser explicado pelo facto de não estar a ser realizada a estimação da velocidade de rotação do móvel, sendo utilizado um valor estabelecido *a priori*. Este modelo apresenta assim uma boa relação desempenho/complexidade.

4.2.5 Movimentos Curvilíneos

Assumindo velocidade angular constante e perpendicular à velocidade tangencial obtém-se um modelo demasiado restritivo. O modelo que agora se descreve não assume que o alvo se desloca sobre uma circunferência, podendo por isso ser utilizado no seguimento de móveis com trajectórias mais complexas.

O primeiro passo a dar para a construção deste modelo passa por determinar como se relacionam as variações nos vectores de velocidade no referencial inercial e no referencial do alvo. A relação fundamental da cinemática [18] estabelece que:

$$\frac{du^I}{dt} = \frac{du^\Lambda}{dt} + \Omega_{\Lambda I} \times u \tag{51}$$

onde I refere-se às grandezas expressas no referencial inercial e Λ refere-se às grandezas expressas no referencial do móvel.$\Omega_{\Lambda I}$ representa a velocidade angular entre os dois referências. Aplicando esta relação ao vector velocidade linear resulta:

$$\dot{v} = \frac{dv^\Lambda}{dt} + \Omega_{\Lambda I} \times v \tag{52}$$

Aplicando novamente a mesma relação mas agora ao vector $\dot{v}$ fica:

$$\ddot{v} = \frac{d}{dt}\left(\frac{dv^\Lambda}{dt} + \Omega \times v\right)^\Lambda + \Omega \times \left(\frac{dv^\Lambda}{dt} + \Omega \times v\right) = \frac{d^2v^\Lambda}{dt^2} + \dot{\Omega} \times v + 2\Omega \times \dot{v} - \Omega \times (\Omega \times v) \tag{53}$$

Nas situações em que a velocidade angular é perpendicular à velocidade linear a última equação pode ser reescrita na forma

$$\dot{a} = -2\alpha a - (2\alpha^2 + \omega^2)v + w \tag{54}$$

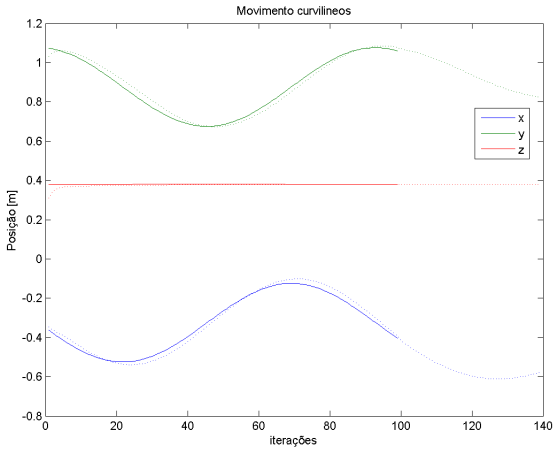
onde

$$w = \frac{d^2v^\Lambda}{dt^2} + \dot{\Omega} \times v \quad \omega = \|\Omega\| = \frac{\|v \times a\|}{\|v\|^2} \quad \alpha = -\frac{v \cdot a}{\|v\|^2} \tag{55}$$

Nas equações 54 e 55 o factor w corresponde ao efeito das forças e momentos sobre o móvel e é modelado como ruído branco (como em [19]). Este factor é o responsável pela capacidade do modelo lidar com trajectórias não circulares, uma vez que, no caso de ser nulo, as equações correspondem a um movimento com velocidade angular variável sobre uma circunferência. O parâmetro α é o factor de amortecimento e expressa a desaceleração tangencial do movimento do alvo.

Ambos os factores são à partida desconhecidos, podendo-se proceder à sua estimação alargando o vector de estado. Esta alteração complica a implementação do sistema, pelo que não foi considerada neste trabalho.

Figura 17: Predição da posição com modelo de movimentos curvilíneos



O modelo contínuo pode ser obtido reescrevendo 54 na forma matricial, resultando para a coordenada i

$$\dot{x}^{(i)}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -(\alpha^2+\omega_c^2) & -2\alpha \end{bmatrix} x^{(i)}(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} w^{(i)}(t) = \tag{56}$$

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -w_n^2 & -2\zeta\omega_n \end{bmatrix} x^{(i)}(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} w^{(i)}(t)$$

em que ω_c representa a frequência de oscilação actual e ω_n representa a frequência de oscilação natural, não amortecida.

O equivalente discreto deste sistema pode então ser obtido, a fim de poder ser implementado, contudo as matrizes A e Q são demasiado complexas para incluir neste texto, podendo no entanto ser consultadas em [20]. Os resultados obtidos com este modelo encontram-se a seguir.

À semelhança da situação presente no modelo anterior, também aqui é possível constatar algum desajuste entre a resposta do modelo e a posição real do móvel, devida também à não estimação dos parâmetros de desenho. Para este modelo o erro quadrático médio foi de $1.53cm$

Os efeitos desta opção estão também presentes no troço correspondente à predição, onde é possível verificar uma diminuição da amplitude do raio da trajectória. Este resultado era contudo previsível, uma vez que este modelo atribui maior liberdade para o movimento do alvo, conduzindo a piores resultados no caso de não proceder à correcta adaptação dos respectivos parâmetros.

4.2.6 Movimentos Balísticos com colisões Elásticas

Um movimento balístico é caracterizado por ter uma dada velocidade inicial e apresentar uma aceleração constante. Uma das diferenças é que o valor desta aceleração é conhecido e é igual à aceleração da gravidade. Este modelo resulta da adaptação do modelo de velocidade constante apresentado, possibilitando-o operar como modelo balístico considerando ainda a existência de colisões com uma superfície conhecida à partida. Desta forma as matrizes A, G, C, e R são iguais às apresentadas para o modelo de velocidade constante.
Uma vez que a aceleração segundo o eixo dos y de acordo com o referencial inercial estabelecido é conhecida e corresponde à aceleração da gravidade, optou-se por incorporar esta informação no modelo através da matriz B. Para este caso a matriz B utilizada é:

$$B = \begin{bmatrix} 0 & 0 & -gT^2 & T & 0 & 0 \end{bmatrix}^T \tag{57}$$

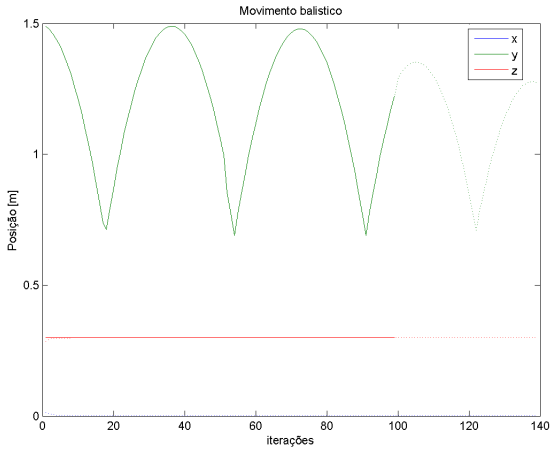
em que g representa a aceleração da gravidade, considerada constante e igual a $9.8ms^{-2}$.
A matriz Q da covariância do ruído do modelo pode ser obtida fazendo:

$$Q^{(i)} = varw^{(i)} A^{(i)} G^{(i)} \left(A^{(i)} G^{(i)}\right)^T = varw^{(i)} A^{(i)} G^{(i)} G^{(i)T} A^{(i)T}$$
$$Q^{(i)} = \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix} \tag{58}$$

$$q_{11} = \frac{9T^4}{4} \quad q_{12} = \frac{3T^3}{2} \quad q_{21} = \frac{3T^3}{2} \quad q_{22} = T^2$$

A fim de permitir a presença de colisões com um plano conhecido, foi adicionado ao funcionamento habitual da estimação com filtro de kalman uma condição específica para este modelo. Esta alteração consiste em monitorizar o valor da coordenada y. Quando esta atingir a superfície do plano horizontal conhecido à partida, procede-se à substituição da matriz A.
Esta substituição é feita numa só iteração e pretende apenas substituir o valor da velocidade em y pelo seu simétrico.
Para permitir um melhor desempenho deste modelo em situações onde exista perda de energia por parte do móvel, foi feita uma tentativa em que o sentido da velocidade não era simplesmente comutado, sendo também multiplicado por um factor que era função da altura máxima atingida pelo alvo. Este factor, que corresponde à perda da energia na colisão, podia ser determinado através da relação entre o intervalo de tempo previsto entre colisões, dada a velocidade do móvel ao embater no plano, e o intervalo de tempo realmente verificado. Contudo esta tentativa foi infrutífera, uma vez que não se conseguia efectuar uma boa estimativa do tempo entre colisões, essencialmente devido ao ruído do modelo.
A alteração efectuada na matriz do sistema é compatível com o formalismo do filtro de kalman, uma vez que é possível utiliza-lo em situações onde as matrizes são variantes no tempo, como nesta situação.

Figura 18: Predição da posição com modelo balístico



Uma vez que a matriz Q é função da matriz A também é necessário proceder à sua substituição. As matrizes A e Q utilizadas no instante em que o móvel atinge o plano são:

$$A^{(1,3)} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \quad A^{(2)} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$Q^{(1,3)} = \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix} \tag{59}$$

$$q_{11} = \frac{9T^4}{4} \quad q_{12} = \frac{3T^3}{2} \quad q_{21} = \frac{3T^3}{2} \quad q_{22} = T^2$$

$$Q^{(2)} = \begin{bmatrix} q'_{11} & q'_{12} \\ q'_{21} & q'_{22} \end{bmatrix}$$

$$q'_{11} = \frac{T^4}{4} \quad q'_{11} = -\frac{T^3}{2} \quad q'_{11} = -\frac{-T^3}{2} \quad q'_{11} = T^2$$

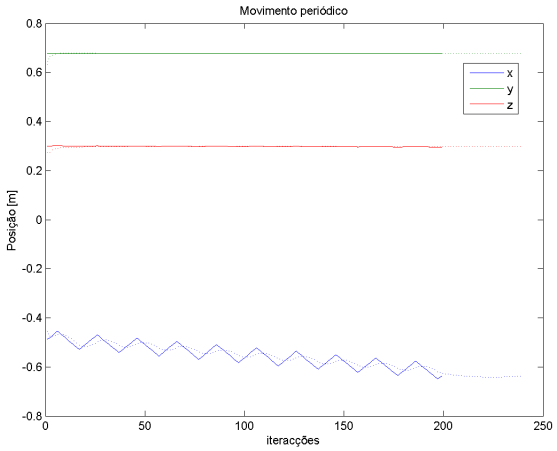
Como habitual, a matriz Q fica então:

$$Q = diag(\sigma^2_{w1} Q^{(1)}, \sigma^2_{w2} Q^{(2)}, \sigma^2_{w3} Q^{(3)}) \tag{60}$$

Alguns resultados obtidos por simulação com este modelo são agora analisados.

Dada a escolha realizada para os ruídos da medida e do modelo utilizados nesta simulação, atribuindo vantagem significativa à leitura da posição, é possível obter uma resposta do modelo praticamente coincidente com a posição real do alvo. Esta opção acarreta sempre alguns inconvenientes, nomeadamente na vulnerabilidade do modelo à presença de ruído na leitura da posição. Esta opção tinha contudo de ser tomada, a

Figura 19: Resposta do estimador MMAE para movimentos com periódicos



fim de permitir um bom desempenho nos instantes de tempo imediatamente após a colisão.

No que respeita ao troço correspondente à predição verifica-se que os resultados são satisfatórios, mantendo a evolução da posição obtida anteriormente.

Neste troço é ainda possível verificar que o modelo esperava uma diminuição mais significativa da altura máxima. Tentou-se resolver este problema, adoptando um esquema de estimação da altura máxima com base no intervalo de tempo entre duas colisões, não se tendo no entanto tido sucesso. O erro de seguimento para este modelo é $1.49cm$

4.3 Movimentos Periódicos

Os movimentos periódicos constituem uma vasta família de movimentos para os quais nenhum dos modelos referidos faz o devido tratamento.

Como se pode ver na figura 19, o melhor resultado possível com a combinação dos modelos analisados não é satisfatória, especialmente na presença de movimentos com um pequeno período, como neste caso.

Para colmatar esta falha, optou-se por adicionar um modelo para seguimento de movimentos periódicos baseado na autocorrelação do sinal. Contudo a introdução deste modelo levou à adaptação do método de estimação, uma vez que para este caso não existe vector de estado, nem é feita qualquer hipótese sobre o tipo de ruído presente.

A autocorrelação de um sinal consiste, de forma grosseira, na multiplicação do sinal com as suas réplicas desfasadas temporalmente, isto é, a correlação do sinal consigo próprio. Esta medida fornece o grau de semelhança entre os dois elementos multiplicados, ou seja, a forma como o sinal combina com uma versão deslocada temporalmente de si próprio. A correlação é feita por janelas, isto é, por um conjunto de N

amostras consecutivas do sinal, em que N é o tamanho da janela.

Para sinais periódicos é de esperar que a autocorrelação das amostras de uma janela com outra desfasada temporalmente produza máximos da autocorrelação sempre que o desfasamento temporal seja o período do sinal. Assim, uma forma simples de detectar a presença de um sinal periódico consiste em fazer a sua correlação para diferentes desfasagens temporais e monitorizar o seu valor máximo. Se este valor exceder um determinado limiar é declarada a detecção de periodicidade, sendo o período precisamente a desfasagem que conduziu ao referido pico.

Existe uma relação directa entre o tamanho da janela e o período máximo possível de detectar, tendo-se à partida $N = 2T_{max}$. Na prática por questões relacionadas com a implementação da autocorrelação que serão abordados mais à frente é necessário ter $N > 3T_{max}$. Quanto ao período mínimo, a única limitação está associada à frequência de amostragem do sistema, tendo-se no melhor dos casos $T_{min} = T$.

O tamanho da janela e consequentemente o período máximo não podem ser no entanto arbitrariamente grandes, por um lado porque levaria à necessidade de adquirir demasiados pontos até poder produzir um qualquer resultado e por outro levaria à necessidade de repetir demasiadas vezes movimentos com período mais curtos, impossibilitando a aplicação deste cálculo em casos onde a frequência de comutação entre tipos de movimento seja mais elevada.

Existem diversos coeficientes de correlação definidos de acordo com a especificidade da aplicação, contudo o mais utilizado é o *coeficiente de Pearson*. Este coeficiente obtém-se dividindo a covariância de duas variáveis pelo seu desvio padrão, como apresentado a seguir.

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E((X-\mu_X)(Y-\mu_Y))}{\sigma_X \sigma_Y} \tag{61}$$

resultando para a autocorrelação

$$R(t,s) = \frac{E[(X_t-\mu)(X_s-\mu)]}{\sigma^2} = \frac{E((X-\mu_X)(Y-\mu_Y))}{\sigma_X \sigma_Y} \tag{62}$$

para o caso de processos estacionários de segunda ordem, a autocorrelação depende apenas da diferença de tempo entre t e s, resultando para a estimativa da autocorrelação de um sinal discreto,

$$\hat{R}(k) = \frac{1}{\sigma^2} \sum_{t=1}^{n-k} (X_t - \mu)(X_{t+k} - \mu) \tag{63}$$

onde k é a desfasagem temporal.

Nos gráficos da figura 20 apresenta-se o resultado da autocorrelação de um sinal triangular

Uma forma simples de implementar (62) consiste em admitir que o sinal toma valores não nulos apenas nos instantes compreendidos na janela. Esta escolha acarreta no entanto alguns inconvenientes, nomeadamente a supressão de máximos. Imagine-se que se tem uma janela exactamente com o dobro do período do sinal. À partida deveria ser possível identificar o troço como periódico, uma vez que se dispõe de dois períodos

Figura 20: Autocorrelação de um sinal periódico

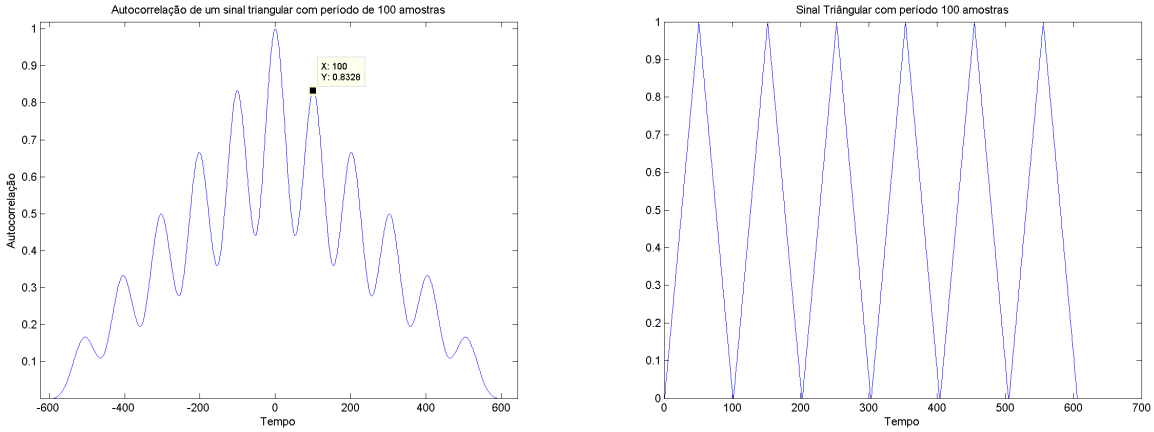
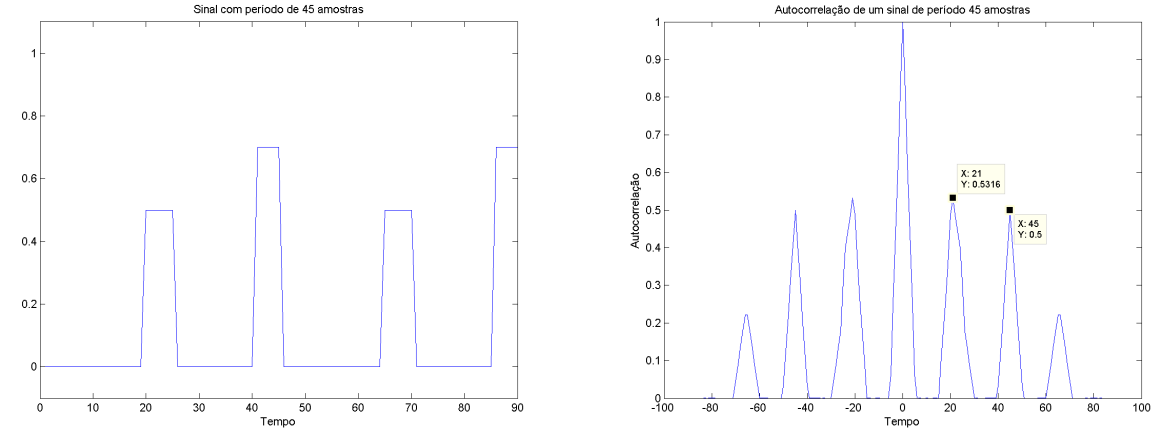


Figura 21: Correlação de sinal periódico com distribuição de energia não uniforme



do sinal, contudo isto pode não ser verdade. Se o sinal for composto por uma parte claramente mais energética e se este troço do sinal corresponder, por exemplo, às amostras de um extremo da janela corre-se o risco de não ser detectado, uma vez que para desfasagens temporais correspondentes ao período do sinal muitas das amostras mais energéticas do sinal seriam multiplicadas por zero, levando à atenuação do pico e eventualmente à sua eliminação.

Um exemplo ilustrativo deste problema encontra-se na figura 21, onde se representa um período do sinal e a sua autocorrelação calculada com esta implementação.

Por esta razão, o período máximo possível detectar tem de ser limitado a um terço do tamanho da janela. A saída deste algoritmo é composta precisamente pelas T_d amostras do sinal correspondentes às primeiras posições da janela, em que T_d é a desfasagem temporal que conduziu ao pico da correlação e consequentemente

ao período estimado.

Dadas as características deste algoritmo, nomeadamente o seu peso computacional, não é possível, em cada instante proceder a um novo ciclo do algoritmo. Assim, uma iteração deste algoritmo consiste em acumular amostras suficientes para preencher uma janela e verificar se existe ou não periodicidade. Caso exista é feita uma estimativa para os N instantes de tempo seguintes, correspondendo a réplicas do período detectado.

5 Selecção de Modelos

Feitos os modelos para o movimento do alvo, falta definir como é feita a escolha do modelo que vai ser realmente utilizado para fazer a predição da posição do móvel. Como já foi referido anteriormente, os modelos vão estar a trabalhar em paralelo, visto que se pretende fazer, a cada instante, a comparação do desempenho dos resultados produzidos por cada um dos modelos.

De cada modelo utilizado resulta, para cada instante, uma estimativa do vector de estado. Fazendo uma analogia com a utilização de múltiplos sensores na presença de ruído, a situação óptima corresponde a combinar a informação proveniente de todos os modelos, produzindo assim a melhor estimativa possível para o vector de estado.

À partida, esta opção pode parecer inviável uma vez que existem modelos com vectores de estado diferentes. No entanto, uma vez que a única diferença é a ausência da componente da aceleração nas três coordenadas é possível realizar a combinação dos vectores. Isto é possível porque existe uma relação directa entre a marginalização das funções densidade de probabilidade e a eliminação de componentes do vector de estado. Desta forma, a ausência da componente de aceleração pode ser entendida como a marginalização em ordem a esta componente, resultando, por exemplo, que o modelo de velocidade constante é, do ponto de vista do vector de estado, equivalente a um dos outros modelos assumindo conhecida a aceleração.

Existem diversas hipóteses para proceder à combinação dos modelos, mas uma vez que se está a lidar com sistemas onde se assume ruído branco gaussiano há uma técnica normalmente utilizada nestas condições: *Multiple Model Adaptive Estimation* [21]. O processo de combinação dos vectores de estado é descrito de seguida.

5.1 MMAE

O estimador adaptativo de múltiplos modelos consiste num banco de filtros de kalman colocados em paralelo, cada um com o seu modelo, e um algoritmo de teste de hipóteses. Cada um dos filtros tem acesso ao vector de medida e produz uma estimativa do vector de estado e um resíduo. O algoritmo de teste de hipóteses utiliza a informação proveniente dos resíduos e da matriz de covariância do estado P a fim de calcular as probabilidades condicionais p_m das várias hipóteses modeladas nos filtros tendo como base o conjunto de leituras realizadas até ao instante actual. Estas probabilidades, que indicam o desempenho relativo de cada um dos modelos, são então utilizadas para calcular uma estimativa do vector de estado ponderada X_{MMAE}.

O teste de hipóteses assenta no facto de os resíduos produzidos pelos filtros de kalman baseados em hipóteses correctas terem a particularidade de serem gaussianos, de média nula e matriz de covariância conhecida. O MMAE compara a magnitude dos resíduos devidamente escalada, para lidar com diversos níveis de ruído, e escolhe a hipótese que corresponde ao resíduo com menor magnitude nos instantes passados.

O resíduo é a diferença entre a medida real e a predição do filtro para esta mesma medida, impossibilitando

assim a utilização desta técnica em situações em que o sistema não produza saídas suficientemente grandes, uma vez que os resíduos assim produzidos não seriam suficientemente energéticos para poder calcular devidamente as probabilidades condicionais. Para a situação em que se pretende utilizar este algoritmo este facto não apresenta qualquer problema uma vez que o deslocamento do móvel entre dois instantes de amostragem, é suficientemente grande para excitar o sistema e produzir resíduos apreciáveis.

A matriz de covarância de um filtro assumindo que a hipótese é coerente com o movimento do alvo é

$$A_m = C_m P_m^- C_m^T + R_m \tag{64}$$

em que o índice m denota cada um dos modelos testados.

A função densidade de probabilidade condicionada da medida z do modelo m no instante k, tendo em conta as medições feitas em todos os instantes anteriores é dada por

$$f_{z(k)|h,z_{k-1}}(z \mid h_m, z_{k-1}) = \beta_m e^{-2q_m(k)}$$

$$\beta_m = \frac{1}{(2\pi)^{n/2}|A_m|^{1/2}} \tag{65}$$

$$q_m(k) = r_m^T(k) A_m^{-1} r_m(k)$$

onde z_{k-1} representa o conjunto de medidas realizadas até ao instante k, h_m denota a hipótese correspondente ao modelo m, $r_m(k)$ o resíduo do modelo m no instante k e $q_m(k)$ o quociente de verosimilhança do modelo m em k. A probabilidade condicionada que se pretende calcular para cada um dos modelos é

$$p_m(k) = P(h = h_m \mid z(k) = z) \tag{66}$$

que pode ser obtida de acordo com [22] e [23] por:

$$p_m(k) = \frac{f_{z(k)|h,z_{k-1}}(z|h_m, z_{k-1}) p_m(k-1)}{\sum_{j=1}^{M} f_{z(k)|h,z_{k-1}}(z|h_j, z_{k-1}) p_j(k-1)} \tag{67}$$

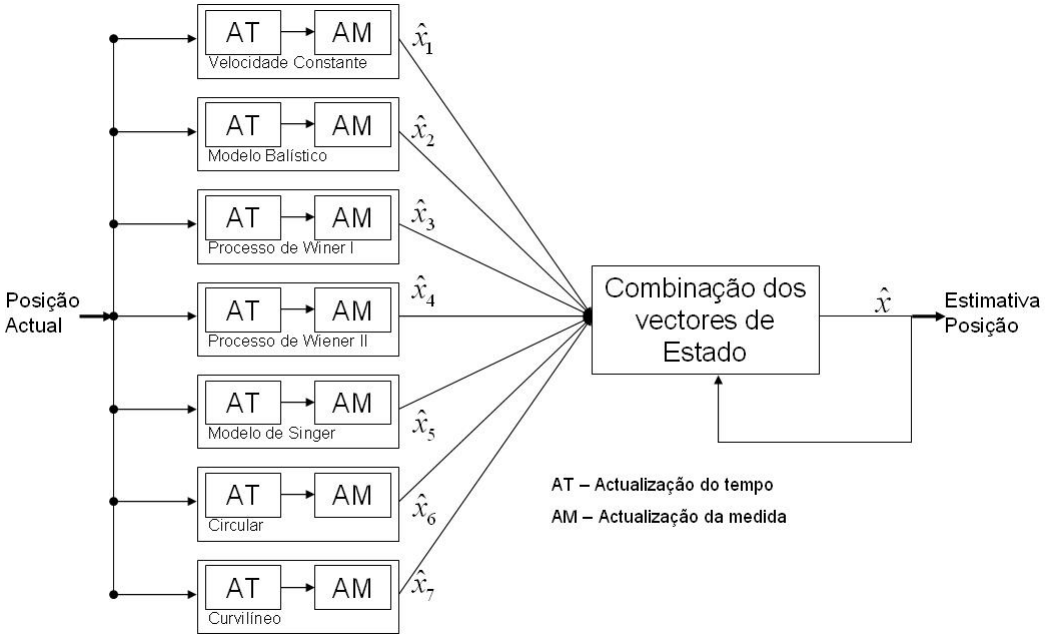
em que as probabilidades condicionadas do instante precedente $p_m(k-1)$ foram utilizadas para pesar a densidade de probabilidade no instante actual, normalizadas sobre todos os modelos, de forma a que $p_m(k)$ respeite a condição

$$\sum_{j=1}^{M} p_j(k) = 1 \tag{68}$$

A identificação da melhor hipótese é simples bastando determinar qual o modelo que maximiza esta probabilidade. A estimativa do vector de estado x_{MMAE} não é contudo o estado correspondente a este modelo, uma vez que, como já foi referido anteriormente, é mais vantajoso fazer a combinação dos vectores de estado cada um pesado pela respectiva probabilidade.

Desta forma, a estimativa do vector de estado é dada por:

Figura 22: Diagrama do estimador de posição 3D



$$x_{MMAE}(k) = \sum_{j=1}^{M} p_j(k) x_m(k) \tag{69}$$

Uma abordagem mais detalhada sobre este algoritmo e outras alternativas podem ser encontradas em [21, 24, 25, 26]. O diagrama do sistema de estimação completo apresenta-se na figura 22

Seguidamente apresentam-se alguns gráficos que mostram a evolução das probabilidades de cada um dos modelos ao longo do tempo, para diversos tipos de movimento.

Nos gráficos das figuras 23 e 24 é possível ver o desempenho do banco de filtros seguido do combinador de vectores de estado. Apresenta-se aqui a posição resultante deste bloco, assim como a evolução das probabilidades associadas a cada um dos modelos. No primeiro caso foi simulado um movimento de velocidade constante, tendo-se obtido uma boa adaptação à posição real do móvel, obtendo-se um erro quadrático médio de $0.036cm$, inferior ao obtido quando foi utilizado apenas o modelo de velocidade constante. No que respeita à evolução das probabilidades, verifica-se que passadas algumas iterações o modelo de velocidade constante acaba por se tornar no modelo com mais peso para o vector de estado, uma vez é o modelo que apresenta uma hipótese de movimento mais coerente com o deslocamento do móvel.

No segundo caso, simulou-se um movimento acelerado. Também aqui se conseguiu uma boa adaptação da saída do estimador de MMAE e da posição real do móvel, com um erro quadrático médio de $0.068cm$, também inferior ao obtido nos modelos de aceleração constante. Neste caso o modelo ao qual é atribuído mais peso é o de Wiener.

Figura 23: Predição da posição e evolução das probabilidades dos modelos para diferentes movimentos

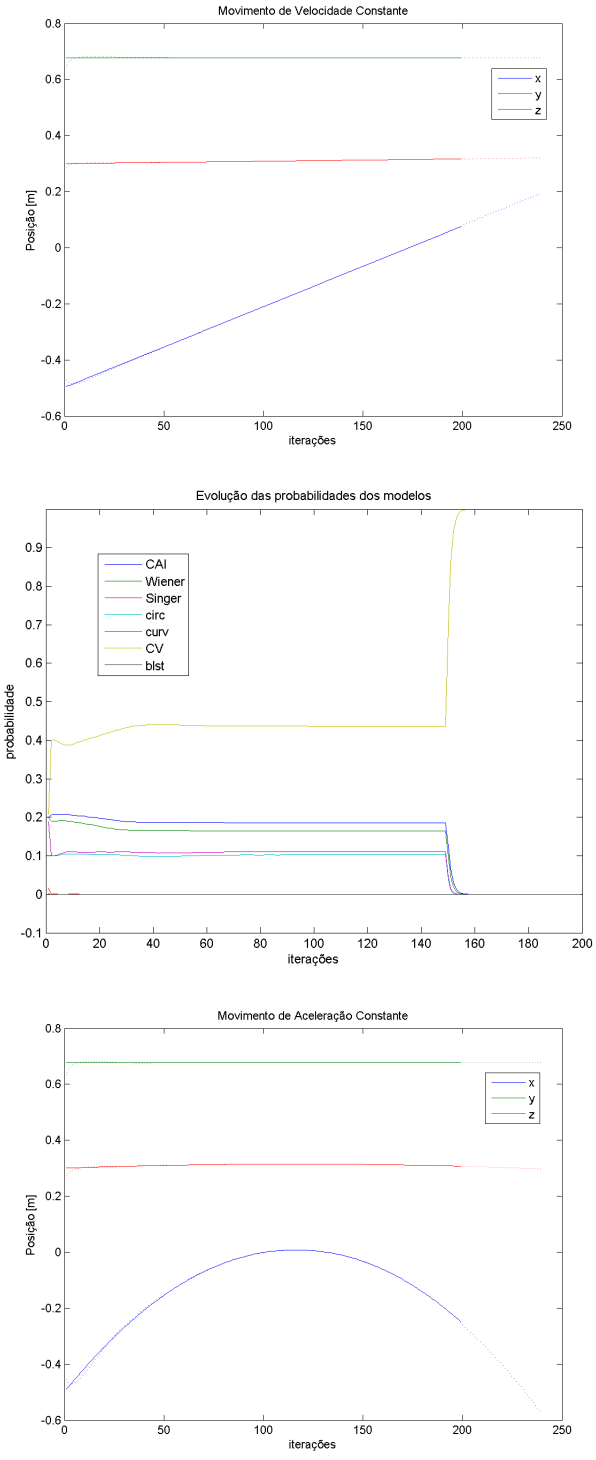
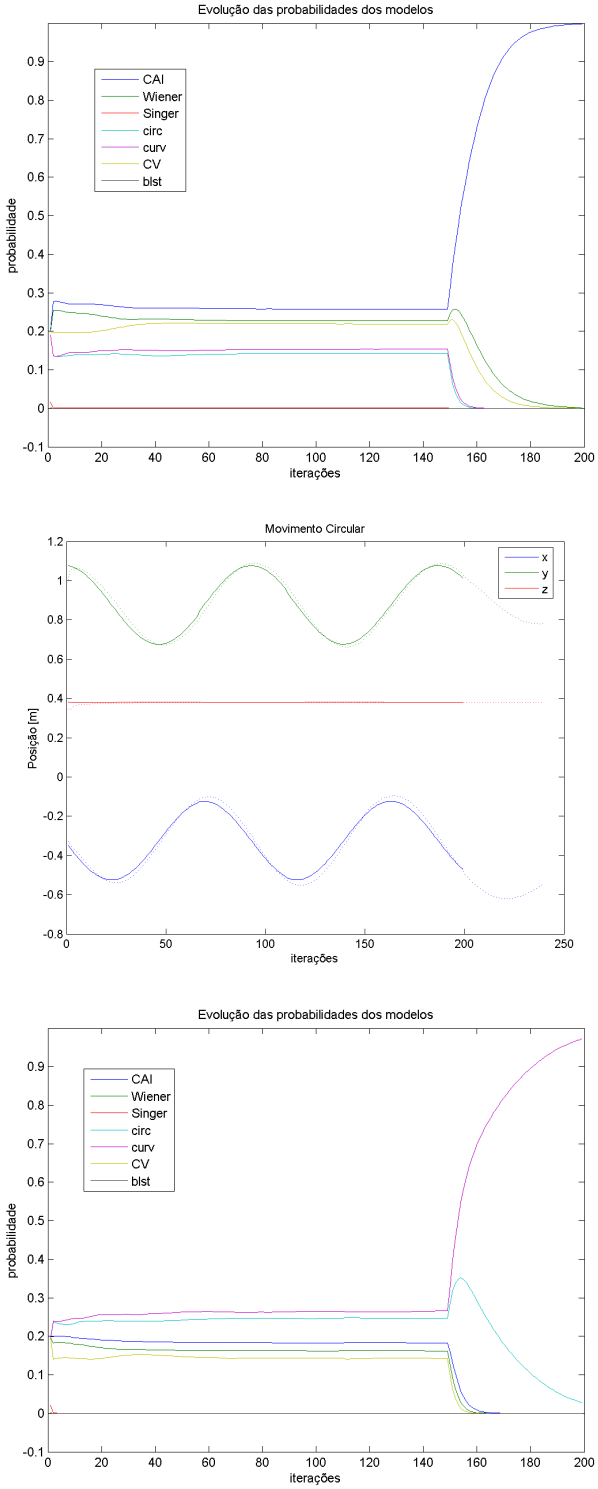


Figura 24: Predição da posição e evolução das probabilidades dos modelos para diferentes movimentos



Nos últimos gráficos, apresentam-se os resultados de simulação para o movimento circular. Neste caso o modelo predominante é o curvilíneo, mantendo no entanto um peso de cerca de 20% para o modelo de movimentos circulares. O erro quadrático médio para esta simulação foi também inferior ao obtido aquando da utilização dos modelos independentes, tendo ficado em $0.22cm$.

Para todas as simulações verificou-se uma redução significativa do erro quadrático médio entre a posição estimada e a posição real, evidenciando as vantagens da combinação de diversos modelos, mesmo no caso das suas hipóteses de movimento não serem verificadas pela trajectória do alvo.

Uma vez produzida a estimativa baseada nos filtros de kalman falta ainda fazer a sua combinação com o resultado do estimador de movimentos periódicos.

Para esta tarefa são consideradas duas situações: Nos instantes em que não foi detectado um período a saída corresponde directamente à estimativa do banco de filtros. Quando é detectado um período é feito um teste simples de hipóteses χ^2, sendo portanto a saída do estimador proveniente de um método ou de outro.

Alguns detalhes sobre o procedimento efectuado para fazer este teste de hipóteses são apresentados a seguir.

5.2 Teste de Hipóteses

Um teste de hipóteses é realizado sempre que se pretende tomar uma decisão sobre uma variável aleatória com base apenas em amostras desta. O teste de hipóteses consiste no estabelecimento de regiões de decisão sobre os valores tomados por uma dada medida estatística. São definidas duas zonas: Uma designada por região de aceitação e outra chamada de região crítica, onde a hipótese é rejeitada.

Cada teste tem associado um parâmetro que corresponde à probabilidade de cometer um erro na escolha da hipótese. Existem dois tipos de erro, o erro do tipo I corresponde a rejeitar a hipótese no caso de esta ser na realidade verdadeira, e erros do tipo II que ocorrem quando se aceita a hipótese apesar desta ser falsa. À probabilidade do erro do tipo I chama-se nível de significância do teste. Os valores típicos para este parâmetro são normalmente inferiores a 5%.

Admitindo conhecida a função densidade de probabilidade, o nível de significância corresponde à área por baixo desta função que corresponde às zonas críticas. Desta forma, é possível determinar os pontos correspondentes aos limites de decisão, levando a que o teste de hipóteses se resuma a uma comparação com o valor tomado pela grandeza sobre a qual se pretende realizar o teste e os limiares correspondentes aos extremos das zonas de decisão.

Contudo para o estabelecimento das regiões de decisão é necessário conhecer a função densidade de probabilidade. A distribuição chi-quadrado tem neste contexto um papel relevante, uma vez que é capaz de produzir uma boa aproximação da distribuição de probabilidade para um grande número de aplicações (apesar de não ser muitas vezes a solução óptima), utilizando quantidades facilmente calculáveis. Esta distribuição é função do parâmetro k que representa o número de graus de liberdade da grandeza que pretendemos descrever. O

Figura 25: Função densidade de probabilidade da distribuição Chi-Quadrado

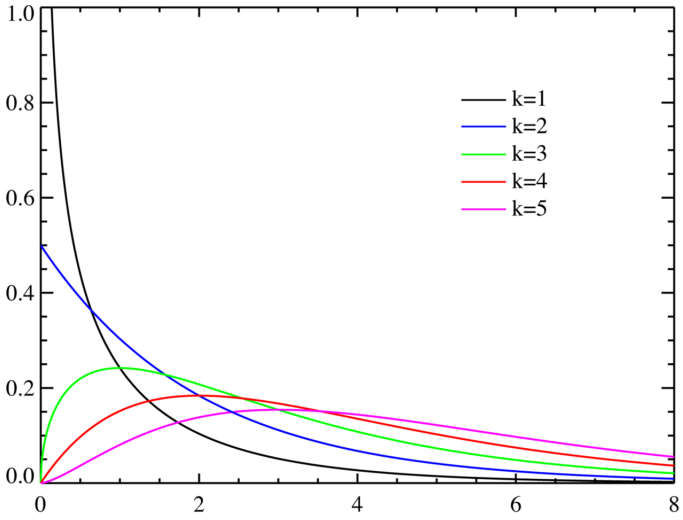


gráfico desta função encontra-se representado na figura 25 para diversos graus de liberdade.

É sabido que para uma variável aleatória $y \sim N(\mu, \Sigma)$ se tem que $(y - \bar{y})^T \Sigma^{-1} (y - \bar{y})$ tem distribuição χ^2k. No contexto de modelos de movimento é frequente utilizar-se este teste sobre os resíduos produzidos devidamente condicionados ([27]). A grandeza sobre a qual é efectuado o teste de hipóteses é então dada por

$$\begin{aligned} \epsilon(\mathrm{k}) &= (z(\mathrm{k}) - \hat{z}(\mathrm{k} \mid \mathrm{k}-1))^T S(\mathrm{k}) (z(\mathrm{k}) - \hat{z}(\mathrm{k} \mid \mathrm{k}-1)) \\ S(\mathrm{k}) &= cov(z(\mathrm{k}) - \hat{z}(\mathrm{k} \mid \mathrm{k}-1)) \end{aligned} \tag{70}$$

Para este trabalho utilizou-se uma distribuição chi-quadrado com 3 graus de liberdade e um nível de significância de 5%.

Os resultados obtidos por simulação para este modelo apresentam-se de seguida. Na figura 26 apresenta-se apenas os resultados provenientes deste modelo, pelo que é necessário esperar que a primeira janela encha para se começarem a produzir resultados. A partir deste momento, a saída deste bloco apresenta uma evolução semelhante ao movimento do alvo, separada, contudo, de um *offset*. Este desvio deve-se ao facto do movimento não ser periódico. Na figura mais abaixo é possível ver em detalhe o troço final do primeiro gráfico, onde é possível ver o resultado da actualização da janela com a respectiva compensação do offset. O erro quadrático médio para esta simulação é $0.96cm$.

O esquema completo utilizado para a estimação da trajectória de um móvel apresenta-se na figura 27. Deste bloco de processamento resulta um conjunto de parâmetros que permitem fazer a predição da posição do móvel. No caso de ser escolhido o banco de filtros, a saída é composta pelas probabilidades atribuídas a cada um dos modelos e os respectivos vectores de estado, enquanto que no caso de ser escolhida a outra

Figura 26: Resultado da predição de um movimento periódico utilizando a autocorrelação. Na figura inferior é possível ver em pormenor a predição do algoritmo

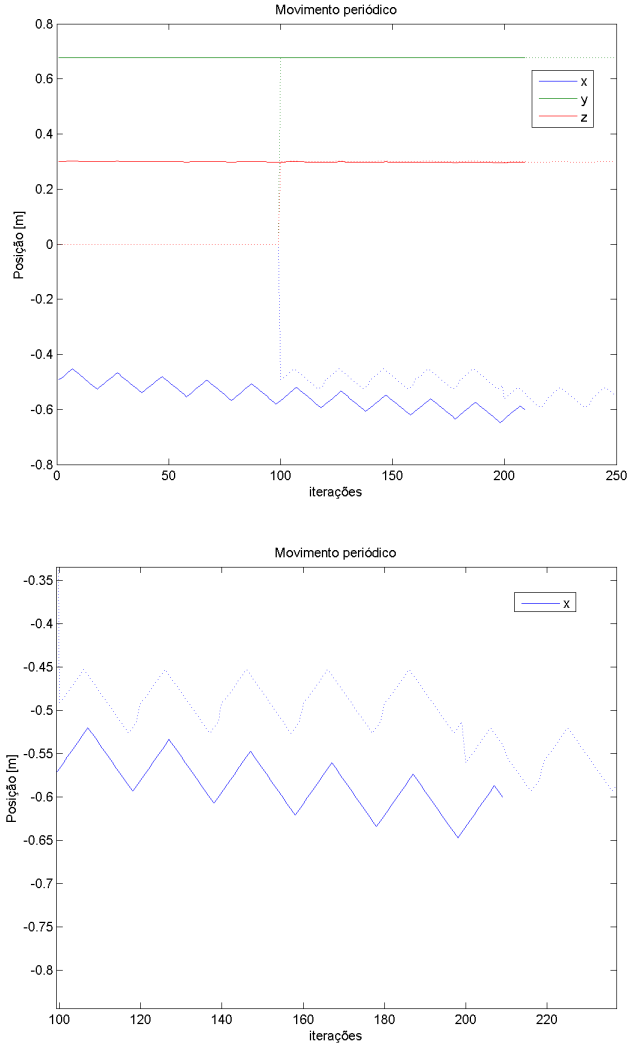
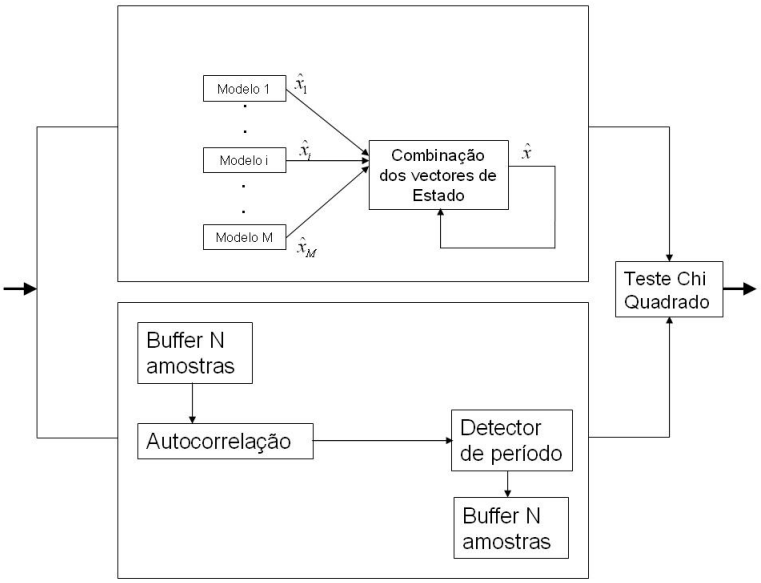


Figura 27: Esquema completo do preditor de posição 3D



hipótese, a saída consiste no conjunto de N coordenadas (x, y, z) correspondentes à posição predita para o móvel nos próximos N instantes de tempo.

6 Predição da Posição

A predição da posição é essencial para conseguir posicionar um corpo com inércia na trajectória de um objecto móvel. A predição neste trabalho é feita utilizando a estrutura apresentada na secção anterior.
As equações que permitem determinar a posição do móvel, para os modelos implementados através das equações às diferenças do estado são as seguintes:

$$\begin{aligned} x(k) &= \hat{x}(k) \\ x(k+1) &= Ax(k) + Bu(k) \\ y(k+1) &= Cx(k) \end{aligned} \tag{71}$$

em que $\hat{x}(k)$ é a estimativa do vector de estado obtida do banco de filtros. Para o caso da estimação de posição baseada na autocorrelação do sinal, uma vez que está disponível um vector com a predição da posição para os próximos instantes, a única tarefa a realizar é aceder ao instante pretendido.
Determinada a forma de obter a posição do móvel no instante de tempo futuro, continua a faltar determinar qual o horizonte de predição necessário para alcançar o móvel. Neste aspecto poderiam ser seguidas diferentes abordagens, nomeadamente no que diz respeito à forma como é feita a intercepção do móvel com o braço do robot. Poderia considerar-se, por exemplo, as hipóteses de minimização do tempo de alcance, colocação do braço de forma a garantir a possibilidade de agarrar o móvel, ou de forma a optimizar o seu seguimento. Por questões de simplicidade será apenas considerada a abordagem que minimiza o erro entre a posição do braço do robot e a trajectória do alvo. Depois de realizados alguns ensaios no simulador com os modelos, verificou-se que o horizonte máximo de predição não deve ultrapassar as 20-30 iterações, a fim de garantir que a posição predita não é muito diferente da posição real.
A determinação do horizonte de predição é feita de forma iterativa, uma vez que para uma resolução analítica seria necessário resolver um conjunto complexo de equações não lineares. Desta forma, para cada horizonte de predição entre um e vinte passos, é determinado o tempo que o braço robótico leva a alcançar a posição predita. O horizonte escolhido é então o que minimiza o módulo da diferença do tempo necessário para alcançar a posição e o tempo disponível devido ao cálculo da predição, isto é, o número de instantes predictos. Sintetizando, o horizonte de predição é dado de acordo com a expressão seguinte:

$$\begin{cases} \hat{t}_p = \min_{t_p} \mid t_{min} - t_p \mid \\ t_p > t_{min} \end{cases} \tag{72}$$

em que t_p é o número de instantes da predição e t_{min} o tempo mínimo necessário para atingir uma dada posição.

Para a utilização desta expressão falta ainda proceder à determinação do tempo necessário para alcançar uma dada posição. Uma vez que todas as juntas que se pretende actuar são activadas praticamente em

simultâneo, o tempo mínimo para que o braço atinja uma dada posição corresponde ao tempo máximo necessário para as juntas chegarem à posição final, ou seja

$$t_{min} = \max_j t_j \tag{73}$$

onde t_j corresponde ao tempo necessário para que a junta j se desloque até à posição final.

Visto que não se pretende resolver o problema de forma analítica, uma alternativa poderia passar por determinar *a priori* através de simulações e experiências no robot real, o tempo necessário para o braço se deslocar entre os pontos P_A e P_B. O resultado deste processamento seria então guardado e consultado de acordo com os pontos pretendidos.

Esta abordagem, apesar de eficaz, seria demasiado trabalhosa, pelo que se optou por seguir outra via. De facto, neste trabalho assume-se a hipótese mais simples possível. A dinâmica do sistema, tanto devida a inércia dos corpos, como devida à resposta dos controladores dos motores não é tida em conta, considerando-se que o movimento de cada uma das juntas do braço do robot corresponde ao deslocamento entre a posição inicial e final com velocidade constante e igual à velocidade máxima possível para cada junta.

Como é óbvio o tempo realmente necessário para realizar o trajecto será sempre superior, na medida em que existe sempre um período de aceleração (aceleramento e travagem). Para compensar esta diferença é adicionado ao horizonte de predição determinado em 72 um factor de correcção que permite absorver esta diferença. Esta opção acarrecta contudo algumas consequências, uma vez que se deixou de encontrar a solução óptima para o problema. Na realidade esta margem de segurança pode levar à escolha de um horizonte de predição não óptimo, uma vez que não é feita qualquer outra consideração no que respeita ao seu valor. Realizados alguns ensaios no simulador optou-se por utilizar um factor de correcção de dois instantes de tempo.

Para garantir a convergência de todos os modelos, nos primeiros trinta instantes de amostragem não é aplicado qualquer comando ao braço do robot, nem sequer é feita qualquer predição.

Após realizar alguns ensaios no simulador verificou-se que o horizonte de predição estava a oscilar demasiado, apresentando alguns picos e levando a oscilações desnecessárias na posição da mão. Para reduzir este efeito, foi imposta uma restrição à variação do horizonte de predição, não sendo permitido que entre dois instantes consecutivos exista uma diferença superior a três passos.

7 Resultados Experimentais

Neste capítulo são apresentados os resultados da aplicação do algoritmo desenvolvido. São utilizadas duas plataformas: o simulador e o robot real. Na primeira secção descreve-se alguns detalhes referentes à construção do simulador, sendo apresentadas algumas características introduzidas. Apresenta-se também aqui alguns detalhes referentes ao robot real e ao protocolo de comunicação utilizado assim como são referidas algumas rotinas de calibração utilizadas. Nas últimas secções apresentam-se os resultados experimentais obtidos incluindo alguns comentários referentes ao desempenho do algoritmo.

7.1 Simulador

A robótica é responsável pelo estabelecimento do interface entre diversas áreas do conhecimento, nomeadamente electrónica, mecânica e software. Esta interligação é em geral complexa e complicada, levando a um investimento avultado. A construção de um bom simulador torna-se assim num ponto decisivo para o sucesso de qualquer projecto no âmbito da robótica. A recriação do mundo real num simulador é bastante complexa e tem sempre associado os inevitáveis erros de modelação do sistema. Contudo, apresenta claras vantagens na medida em que permite desacoplar a fase de desenvolvimento dos algoritmos do sistema físico, facilitando bastante o período de *debug* e experimentação e permitindo a concentração dos projectistas na parte realmente criativa do trabalho.

O *Webots* é uma plataforma que permite simular ambientes com múltiplos robots. Tem já disponível um vasto conjunto de bibliotecas que permitem implementar rapidamente as funções típicas presentes nos projectos da robótica. Esta plataforma integra o simulador ODE - Open Dynamic Engine, que simula a dinâmica de objectos rígidos a 3D [28], tornando assim os modelos mais realistas.
Para cada robot introduzido é possível especificar um controlador desenvolvido em C ou C++. Este controlador é responsável pelo comportamento do robot a que está associado e tem já definido um conjunto de funções que permitem fazer leituras dos sensores e aplicar comandos aos actuadores. Estes controladores devem ter uma secção de inicialização do robot e uma secção que é executada em todos os ciclos de amostragem.

Nas secções seguintes abordam-se várias questões relacionadas com o desenvolvimento do simulador para o *Baltazar*. Na primeira secção estabelece-se o protocolo de comunicação entre o simulador e os algoritmos de seguimento desenvolvidos neste trabalho. Na secção seguinte descreve-se o processo de criação do modelo do robot e finalmente na terceira secção acrescentam-se alguns objectos móveis ao simulador para permitir o teste dos algoritmos.

7.1.1 Comunicação com o robot

O *Baltazar* real é residente numa maquina ligada em rede. As leituras dos sensores e cameras e os comandos para os actuadores são enviados por uma outra máquina e a comunicação é feita através do Yarp (Yet Another Robot Platform) [29]. Esta plataforma foi desenvolvida, por um lado para permitir lidar com o elevado peso computacional normalmente presente nas aplicações robóticas, e por outro, para proporcionar a separação entre as camadas associadas directamente ao hardware e as camadas relacionadas com o software propriamente dito.

O Yarp pode ser instalado em qualquer sistema operativo e é composto por três camadas essenciais: a camada de interface com o sistema operativo, a camada de interface com o robot e a camada de software que disponibiliza já um conjunto de ferramentas que são normalmente utilizadas em robots humanóides.

Esta plataforma foi desenvolvida para operar em tempo real, pelo que o overhead introduzido tem de ser reduzido e consequentemente o conjunto de funcionalidades disponibilizadas pela plataforma tem de ser limitada. Este facto não obsta a que por exemplo, sejam atribuídos nomes aos portos, em vez dos típicos endereços IP, tornando a comunicação mais intuitiva e simples de gerir.

Com o intuito de facilitar a adaptação do software desenvolvido para o robot real e para o simulador, optou-se por acrescentar esta funcionalidade no simulador. Sendo assim foi necessário desenvolver o software correspondente à camada de interacção com o robot, que neste caso é virtual. Este software estabelece o interface entre as estruturas disponibilizadas pelo Yarp e as funções já fornecidas pelo *Webots* para interacção com o robot. Assim, sempre que se põe o simulador a correr é criado um conjunto de portos que permite comandar remotamente o robot do simulador, tal como acontece com o sistema real. Desta forma, consegue-se simular o próprio interface com o robot.

A lista de portos criados e a respectiva função e estrutura de dados associada são apresentados na tabela 3. Todas as leituras e escritas em portos no *Webots* são síncronas garantindo o sincronismo entre o algoritmo de predição e controlador do simulador.

O vector de comandos contém apenas uma posição e indica qual o objecto que se pretende fazer seguimento, conforme será referido na devida subsecção. Os vectores de recepção e envio das posições das juntas da cabeça têm 3 posições e correspondem ao movimento de *pan*, *tilt* e *vergence*.

O vector com as posições actuais dos objectos tem 6 posições, sendo as primeiras três referentes às coordenadas 3D do objecto em tracking e as restantes posições correspondentes às coordenadas da mão do *Baltazar*.

Finalmente o vector correspondente à posição angular das juntas do braço, contém 6 posições que estão associadas, respectivamente, aos movimentos de *shoulder flexion/extension, shoulder abduction/adduction, internal/external rotation, elbow flexion/extension, wrist supination/pronation* e *wrist flexion/extension*. Na figura 28 pode-se ver a localização e orientação correspondente a cada um destes movimentos.

A interacção com o robot real é feita por intermédio de *devices* específicos que se encarregam do interface

Tabela 3: Listagem dos portos criados no simulador para interacção com aplicação externa

Nome do Porto	Função	Estrutura de dados
/webots/out/img/L	Envio imagem camera esquerda	ImageOf<PixelRgb>
/webots/out/img/R	Envio imagem camera direita	ImageOf<PixelRgb>
/webots/out/realObjPos	Envio posição real	Vector
/webots/out/actHeadJoints	Envio posição angular - cabeça	Vector
/webots/out/actArmJoints	Envio posição angular - braço	Vector
/webots/in/commands	Recepção comandos do simulador	Vector
/webots/in/headJoints	Recepção posição angular - cabeça	Vector
/webots/in/armJoints	Recepção posição angular - braço	Vector

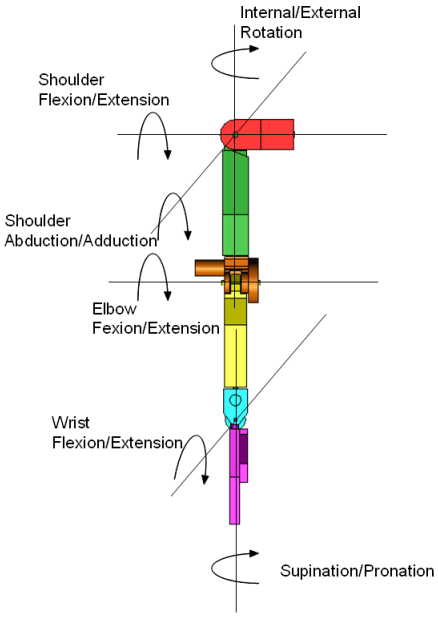


Figura 28: Juntas e movimentos do braço do *Baltazar*

Tabela 4: Listagem dos portos criados no robot real para interacção com aplicação externa

Nome do Porto	Função	Estrutura de dados
/LeftEye	Envio imagem camera esquerda	ImageOf<PixelRgb>
/RightEye	Envio imagem camera direita	ImageOf<PixelRgb>
/BaltaHead	Interface com o driver da cabeça	Device
/BaltaArm	Interface com o driver do braço	Device

entre os portos yarp e os *drivers* específicos do *hardware* do robot, nomeadamente os *drivers* das placas de controlo dos motores do braço e da cabeça e das cameras. Estes *devices* do yarp implementam um conjunto de métodos definidos numa interface genérica, permitindo obter um maior nível de abstracção por parte do utilizador.

Para utilizar estas ferramentas basta criar um conjunto de classes já disponível no yarp e utilizar os métodos disponibilizados. De entre as várias funções disponibilizadas destacam-se as essenciais, nomeadamente a configuração de acelerações e velocidades máximas para cada um dos motores e os comandos para leitura e actuação em velocidade e posição de cada uma das juntas.

Na tabela 4 apresenta-se a listagem dos portos criados para comunicação com o robot real. Foram omitidos nesta tabela todos os portos auxiliares lançados pelos *devices*, nomeadamente os porto de *quit* e os portos de entrada e saída de comandos rpc e estado do sistema.

A distância focal das cameras do robot real foi determinada utilizando uma funcionalidade já disponível no Yarp. O *CamCalibConf* permite fazer automaticamente a estimativa de um conjunto de parâmetros das cameras recorrendo a um padrão com estrutura e dimensões conhecidas. Este programa detecta os cantos internos do tabuleiro de xadrez e actualiza a estimativa dos parâmetros da camera sempre que o utilizador selecciona uma imagem para processar. Para realizar a estimativa é necessário uma sequência de imagens em que o padrão tenha sofrido diversas transformações.

7.1.2 Modelo do *Baltazar*

A fim de minimizar as discrepâncias entre o robot real e o simulado, deve-se ter especial atenção à criação do modelo do robot, tornando o modelo tão fiel quanto possível. Para tal é necessário conhecer certos parâmetros, nomeadamente massa das peças, matrizes de inércia, centros de gravidade, dimensões, velocidades máximas e limites físicos de cada um dos motores, entre outros.

Contudo, a construção de um simulador deve ser sempre feita à medida das necessidades, uma vez que não faz sentido construir um simulador muito fiável e consequentemente com um grande peso computacional, se a tarefa a que se destina não exigir muito rigor nalguns aspectos da realidade. O modelo 3D das peças do robot obteve-se importando para o *Webots* um ficheiro VRML 2.0 (Virtual Reality Modeling Language)

gerado previamente no *SolidWorks 2006*. Este programa de CAD em 3D específico para o desenvolvimento de modelos em computador, permitiu ainda a determinação das massas, centros de gravidade e matrizes de inércia, uma vez que se encontravam já disponíveis neste formato todos os modelos utilizados no desenho e construção do próprio robot [1].

Ao tentar introduzir toda a informação disponível no simulador verificou-se um aumento significativo do seu peso computacional, levando à realização de algumas simplificações, uma vez que se estava a comprometer bastante a possibilidade de validação rápida dos algoritmos desenvolvidos.

Neste sentido optou-se por considerar que cada um dos troços do robot era constituído por uma única peça de alumínio com densidade de $1200kg/m^3$, eliminando todos os detalhes dos vários elementos constituintes de cada troço do braço disponíveis nos modelos CAD.

As massas de cada uma das peças constituintes do braço robótico apresentam-se na tabela 5. Nesta mesma tabela apresenta-se também os centros de gravidade de cada uma das peças e as respectivas matrizes de inércia. Obtido o modelo pode-se então proceder à fase de calibração do simulador.

Nesta fase foram realizadas algumas experiências com o robot real e com o robot simulado de forma a ajustar alguns parâmetros do simulador, nomeadamente a resposta dinâmica dos motores de cada uma das juntas. Dado que a resposta de cada um dos motores depende da dinâmica de todo o corpo, e dado que esta depende da configuração em que se encontra o robot, teve-se o cuidado de fazer testes semelhantes no simulador e no sistema real. Nas figuras 7.1.2 e 7.1.2 é possível ver a resposta ao escalão de cada uma das juntas do braço do robot simulado assim como a resposta ao mesmo escalão no braço real. Na coluna da esquerda encontram-se os gráficos respeitantes ao robot real e na coluna da direita encontra-se a resposta do simulador.

Como é possível constatar, apesar de se terem introduzido todos os parâmetros físicos permitidos pelo *webots* no simulador do *Baltazar*, não foi possível simular de forma exacta a resposta ao escalão. Como é claro nestes gráficos, o troço de aceleração e desaceleração é bem mais evidente na situação real. No entanto o tempo de estabelecimento é aproximadamente o mesmo, viabilizando assim a utilização do simulador para análise do desempenho do algoritmo de predição.

As cameras do robot estão também, o quanto possível, modeladas no simulador, tendo-se especificado a resolução e a distância focal destas. O aspecto do robot no simulador é o da figura 30.

Feito o simulador para o robot, pretende-se agora que este interaja com outros objectos pelo que é também necessário simulá-los.

7.1.3 Objectos para interacção

Para o teste do desempenho no seguimento de objectos interessa introduzir corpos com diferentes modelos de movimento, pelo que se optou por introduzir um robot móvel diferencial, que permite simular facilmente

Tabela 5: Parâmetros físicos de cada troço do robot introduzidos no simulador

Troço	Massa [Kg]	Centro de massa [m]	Matriz de Inércia [Kg/m^2]
Ombro	0.69693	(-0.11501,0,-0.01954)	$\begin{bmatrix} 0.00225 & 0 & 0.00141 \\ 0 & 0.01215 & 0 \\ 0.00141 & 0 & 0.01024 \end{bmatrix}$
Braço	1.05542	(-0.00024,-0.12706,-0.05313)	$\begin{bmatrix} 0.02457 & 0.00001 & 0.00001 \\ 0.00001 & 0.00377 & 0.00723 \\ 0.00001 & 0.00723 & 0.02126 \end{bmatrix}$
Cotovelo	0.46068	(0.00830,-0.03352,-0.00604)	$\begin{bmatrix} 0.00121 & -0.00022 & 0.00016 \\ -0.00022 & 0.00093 & -0.00003 \\ 0.00016 & -0.00003 & 0.00114 \end{bmatrix}$
Antebraço	0.65363	(0.02662,-0.10908,-0.01685)	$\begin{bmatrix} 0.01013 & -0.00189 & -0.00029 \\ -0.00189 & 0.00099 & 0.00124 \\ -0.00029 & 0.00124 & 0.01028 \end{bmatrix}$
Punho	0.21565	(0,-0.06916,-0.00390)	$\begin{bmatrix} 0.00125 & 0 & 0 \\ 0 & 0.00017 & 0.00010 \\ 0 & 0.00010 & 0.00114 \end{bmatrix}$
Mão	0.51484	(0.00415,-0.08888,-0.02616)	$\begin{bmatrix} 0.00604 & -0.00013 & -0.00003 \\ -0.00013 & 0.00088 & 0.00125 \\ -0.00003 & 0.00125 & 0.00528 \end{bmatrix}$

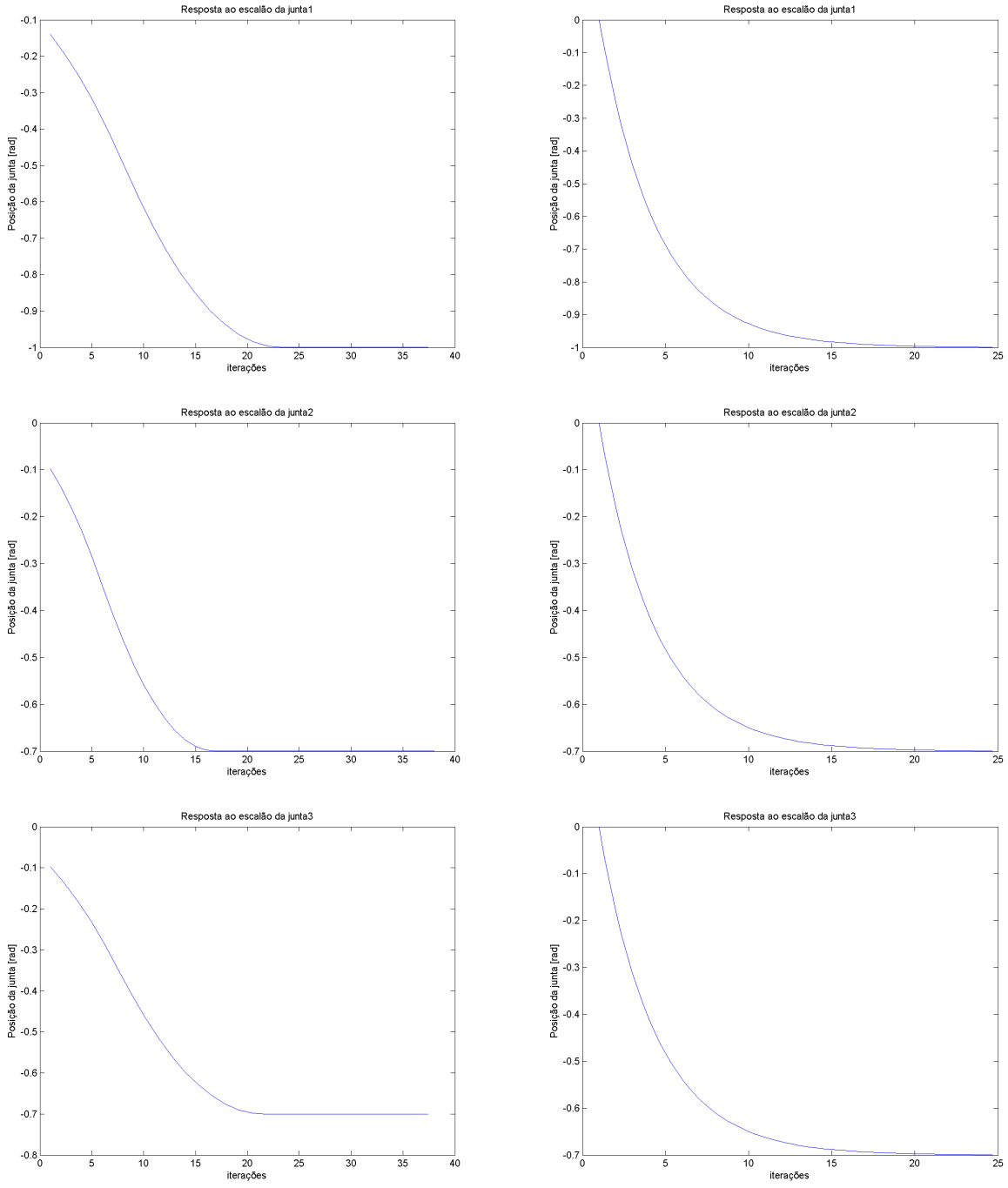


Figura 29: Resposta ao escalão de cada uma das juntas do *Baltazar* real (esquerda) e simulado(direita)

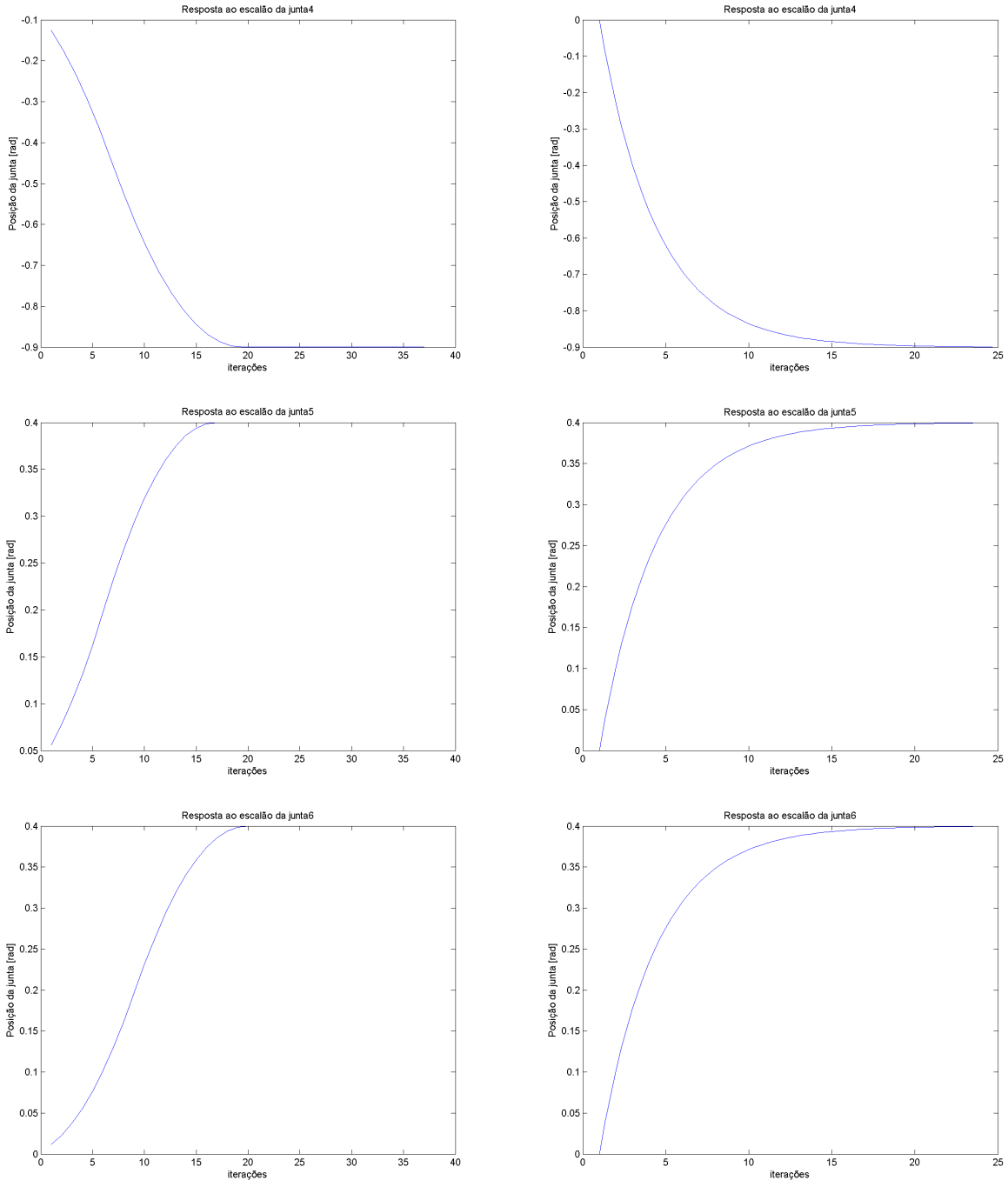


Figura 30: Resposta ao escalão de cada uma das juntas do *Baltazar* real (esquerda) e simulado(direita)

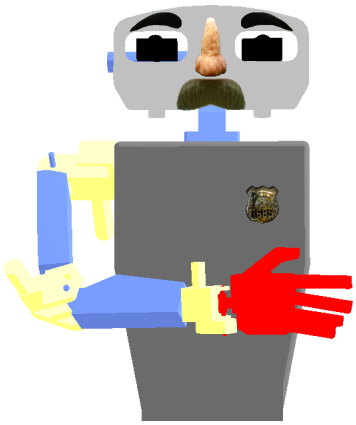


Figura 31: Simulador do *Baltazar*

Tabela 6: Lista de objectos e respectivos canais

Canal	Objecto
1	Robot Móvel
2	Bola Elástica
3	Pêndulo
4	Hélice

diversos tipos de movimento no plano horizontal, consoante os comandos aplicados a cada uma das rodas. Criou-se ainda um pêndulo e um objecto para simular movimentos circulares num plano não horizontal. Para o teste de modelos balísticos foi acrescentado uma superfície e uma bola elástica. O aspecto de cada um destes objectos pode ser visto nas imagens seguintes (figura 7.1.3). Todos os objectos utilizados para fazer seguimento têm cores saturadas, uma vez que melhora o desempenho do algoritmo de *tracking*.
Para simplificar o desenvolvimento do algoritmo de seguimento e dado a possibilidade de controlo do estado do mundo virtual, foi adicionada a possibilidade de ter acessível no algoritmo de predição a posição real de cada um dos objectos, facilitando o cálculo do erro de seguimento dos corpos e a avaliação quantitativa de cada uma das técnicas abordadas. Assim, deve-se indicar qual o objecto que se pretende seguir afim deste fornecer a respectiva posição. Cada um dos objectos tem associado um transmissor de infravermelhos com um canal respectivo no qual é enviada a posição real do objecto. No *Baltazar* está disponível um receptor de infravermelhos, bastando comutar o canal para receber a posição dos diferentes objectos. A escolha do canal do receptor de infravermelhos é feita pelo porto de comandos e consiste num número, como apresentado na tabela 6.

Para facilitar o tracking dos vários objectos foram utilizadas cores bastante saturadas, e foram evitadas

oclusões dos objectos a que se pretende fazer seguimento.

7.2 Resultados - Simulador

Apresentam-se agora alguns resultados de simulação obtidos com o sistema completo. Os gráficos encontram-se nas figuras 33, 34 e 35. À semelhança do realizado anteriormente, as simulações realizadas correspondem a movimentos de velocidade constante, aceleração constante e circulares. Para cada uma das simulações apresentam-se três gráficos: um que representa a evolução das coordenadas da mão do robot e as coordenadas do móvel; outro que representa o erro absoluto entre a posição da mão e do alvo, ao longo do tempo e finalmente um gráfico que representa qual o modelo escolhido em cada instante da simulação.

Em todos os casos aqui apresentados o modelo ao qual é atribuído mais peso é o que corresponde à hipótese de movimento que se verifica na realidade. Comparando o erro cometido no posicionamento do braço, verifica-se que o erro máximo acontece no movimento circular, o que seria de esperar, uma vez que é o movimento mais complexo destes três.

Os erros de posicionamento presentes em todas as simulações não são exclusivamente devidos ao facto de não se conseguir compensar a totalidade do atraso imposto pela inércia do sistema, mas também à falta de precisão das cinemáticas. Na realidade, apesar de se solicitar ao bloco responsável pela cinemática que coloque o braço numa dada posição, este coloca-o numa posição diferente devido a erros de modelação e a incoerências entre o robot e a sua representação utilizada para fazer os cálculos da cinemática.

Os gráficos seguintes apresentados nas figuras 36 a 41 dizem respeito apenas à simulação do movimento circular e representam o erro absoluto correspondente à utilização de cada um dos modelos.

Como seria de esperar o erro de seguimento é máximo par o modelo de velocidade constante visto ser o menos flexível, rondando quase os 20cm. Para os modelos de aceleração constante já é possível obter um erro ao dos modelos circular e curvilíneo na ordem dos 10cm. Estes resultados vêm confirmar a escolha do MMAE e do teste Chi-quadrado que atribui mais peso ao modelo curvilíneo numa simulação idêntica. De facto este é o modelo que conduz a menores erros de seguimento, devendo portanto ser o modelo predominante.

Para que se possa comparar o acréscimo no desempenho do sistema devido à predição, apresentam-se de seguida os gráficos correspondentes à simulação dos mesmos tipos de movimentos sem recorrer à predição, ou seja, utilizando apenas a posição actual do móvel. Como se pode ver para todas as simulações apresentadas nas figuras 42- 44 o erro absoluto entre a posição da mão e a posição do móvel é inferior quando se utiliza a predição, verificando-se uma diminuição do erro da ordem dos 50% de acordo com os gráficos das figuras 33-35.

Apesar desta melhoria, é também possível observar nestas figuras um aumento do ruído no sistema, especialmente no caso do movimento circular.

Este ruído é devido fundamentalmente a comutação do horizonte de predição, levando a que exista uma

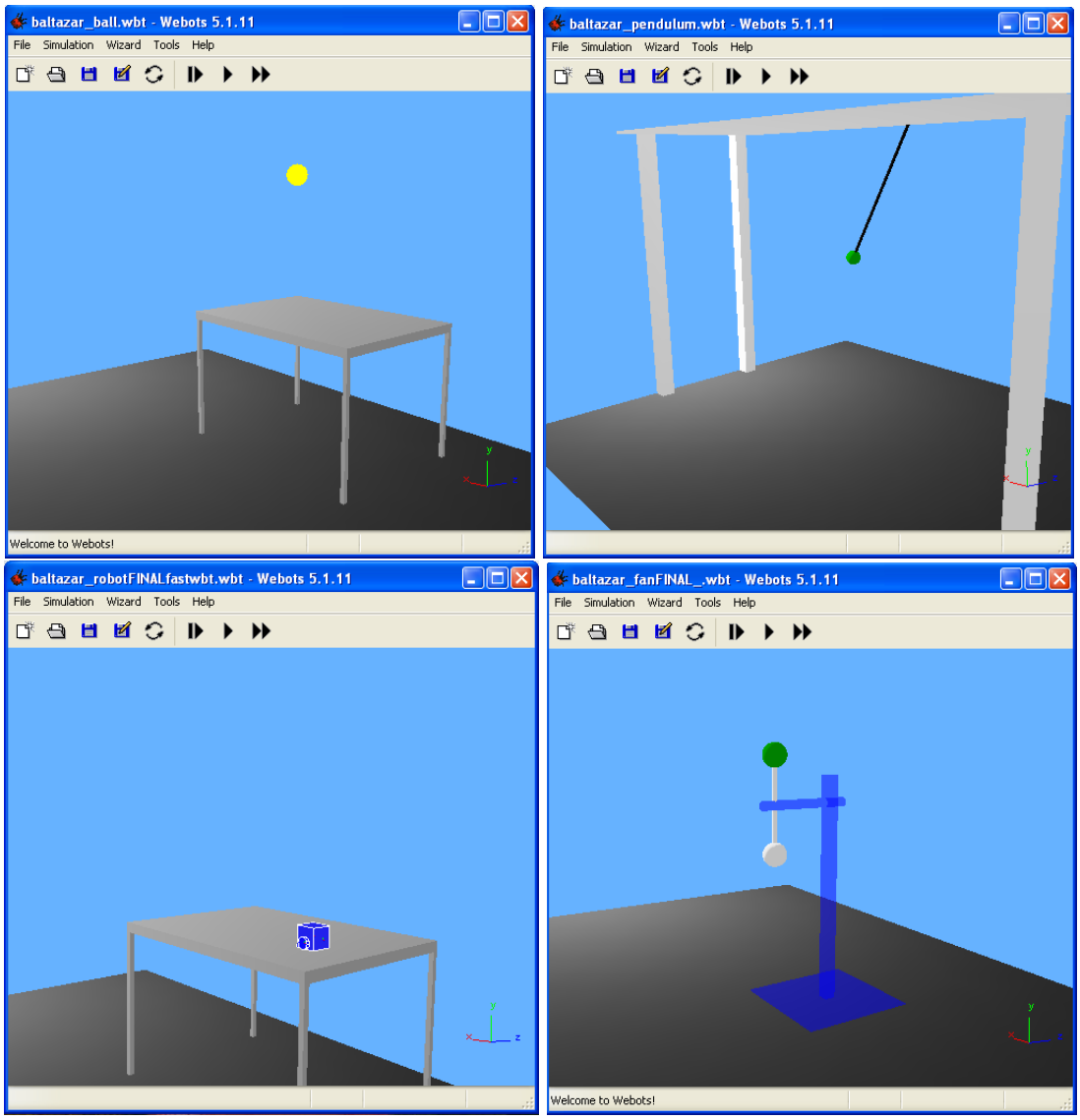


Figura 32: Objectos introduzidos no *Webots* para simulação de diversos modelos de movimento

Figura 33: Predição da posição utilizando o preditor completo - movimento de velocidade constante

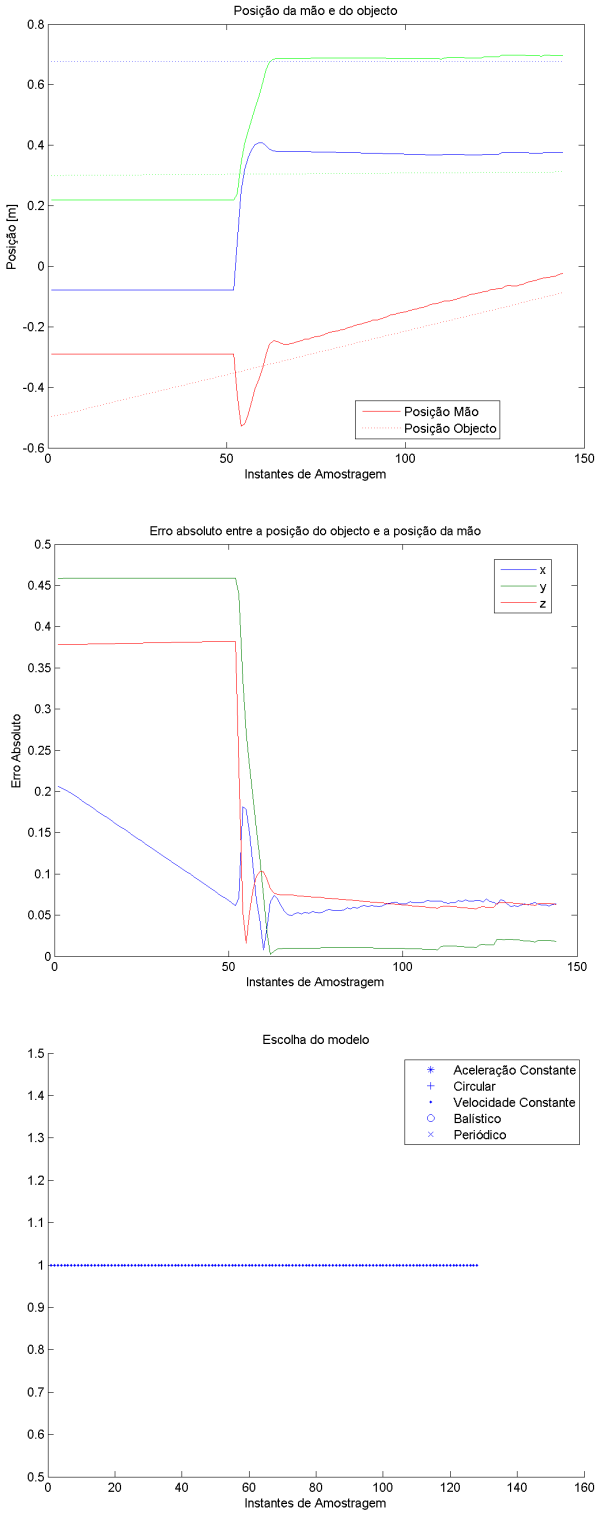
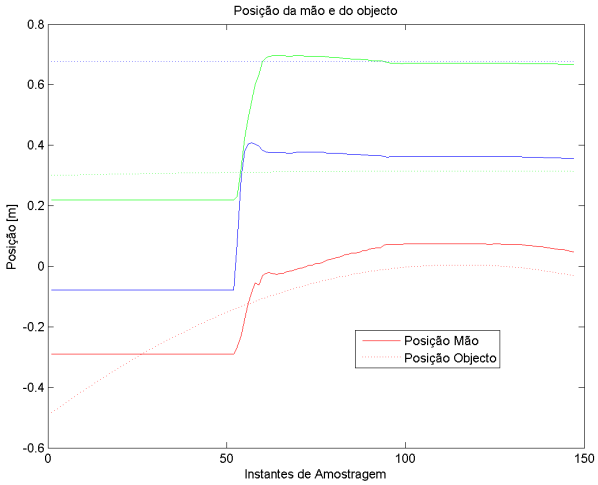
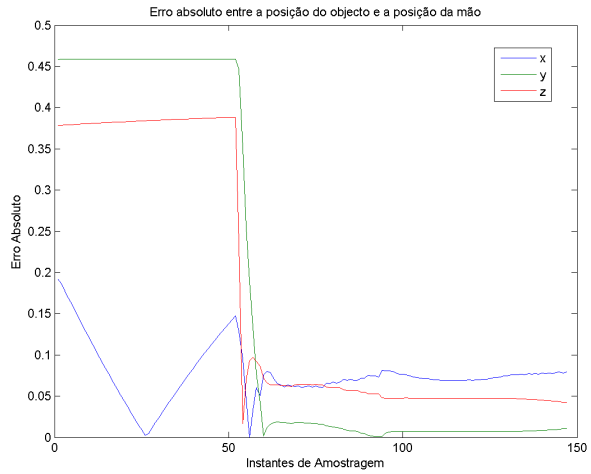


Figura 34: Predição da posição utilizando o preditor completo - movimento de aceleração constante





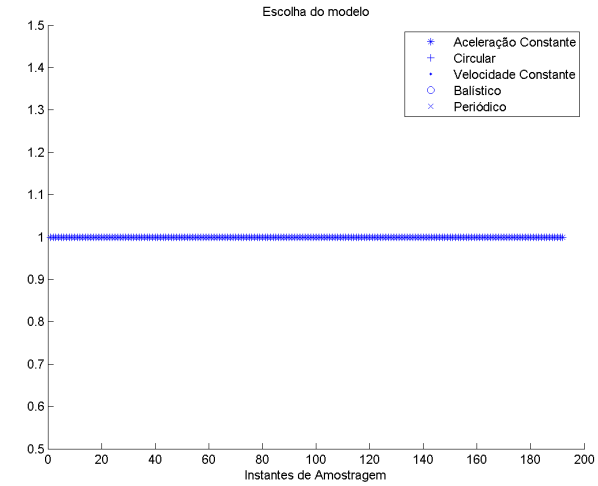


Figura 35: Predição da posição utilizando o preditor completo - movimento circular

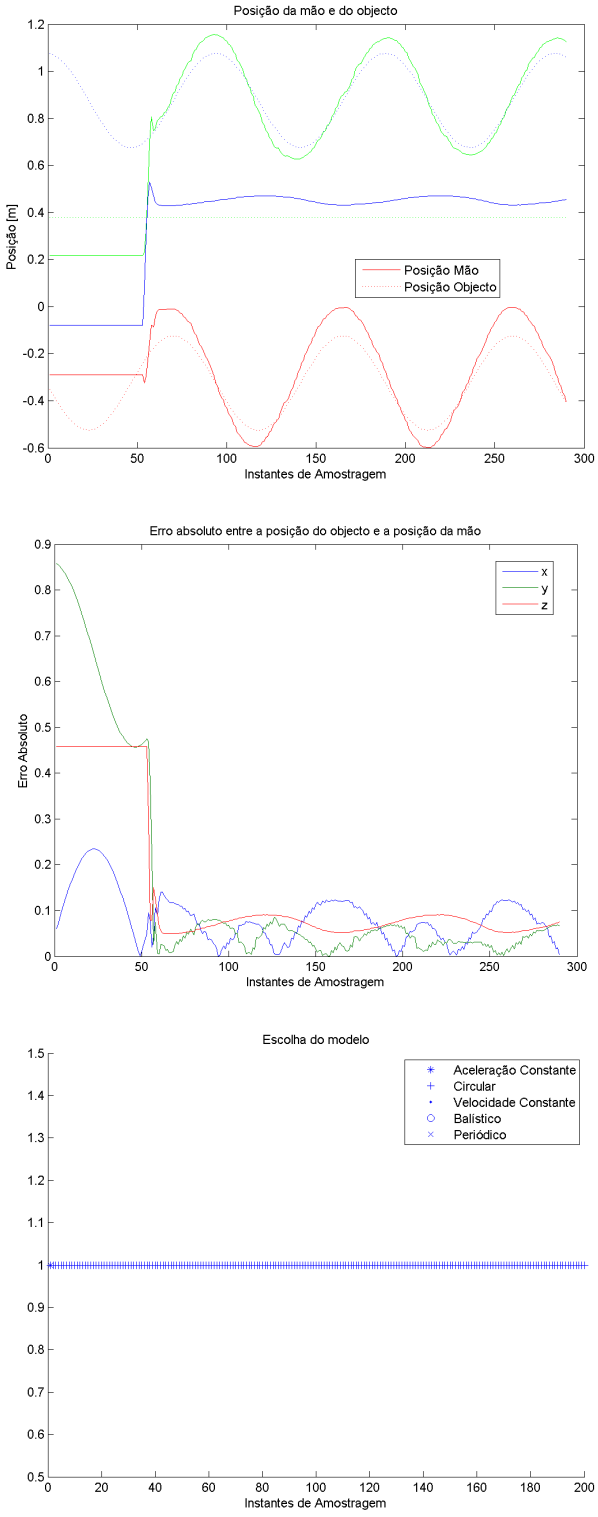


Figura 36: Erro de predição para alvo com trajectória circular - Modelo de Velocidade constante

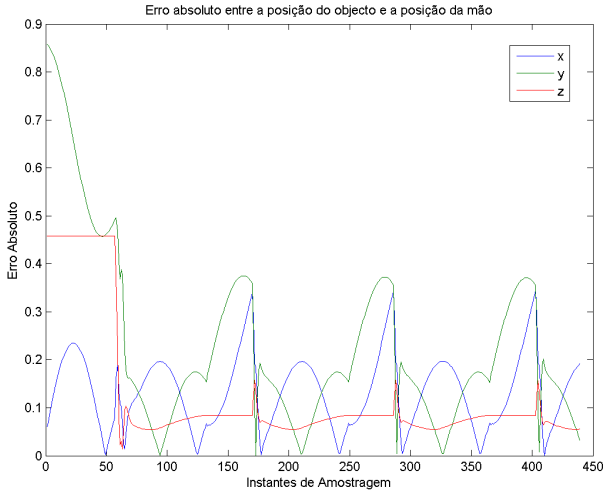


Figura 37: Erro de predição para alvo com trajectória circular - Processo de Wiener

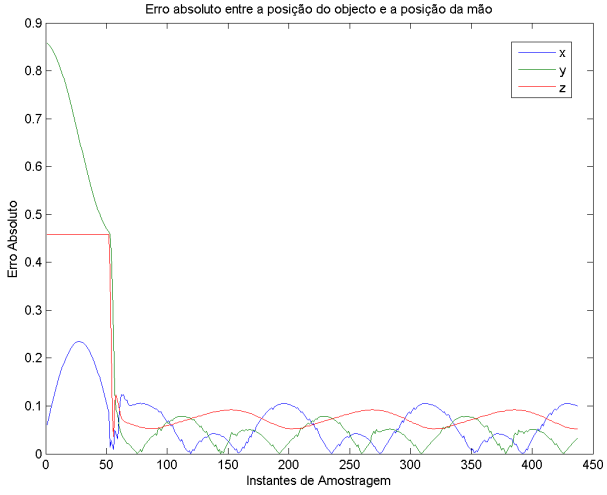


Figura 38: Erro de predição para alvo com trajectória circular - Processo de Wiener: incrementos independentes

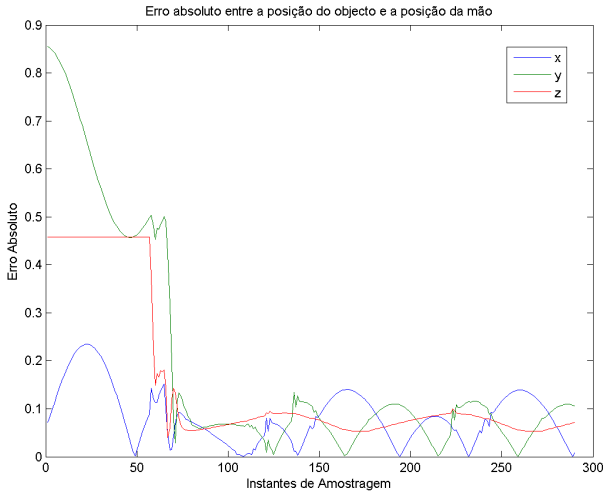


Figura 39: Erro de predição para alvo com trajectória circular - Modelo de Singer

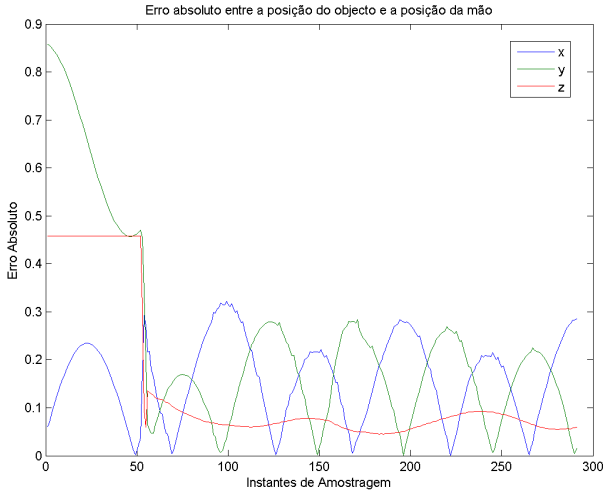


Figura 40: Erro de predição para alvo com trajectória circular - Movimentos Circulares

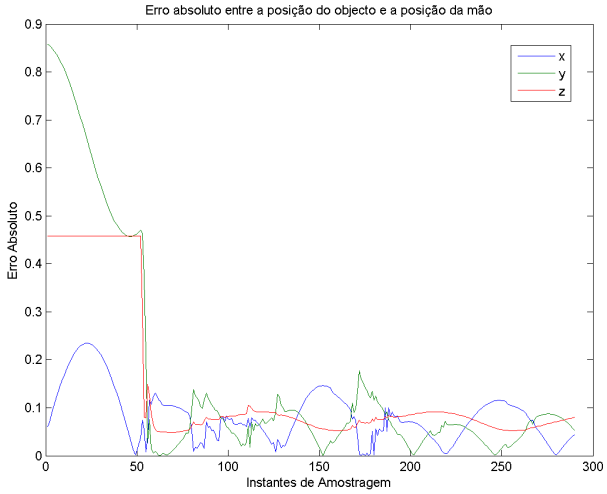


Figura 41: Erro de predição para alvo com trajectória circular - Movimentos Curvilíneos

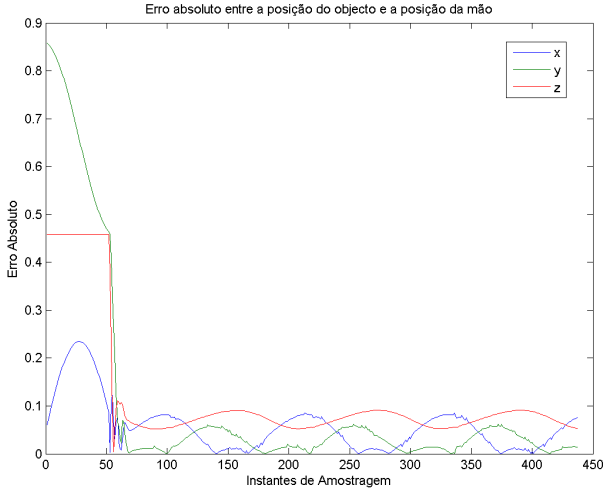


Figura 42: Desempenho do sistema sem realizar predição da posição - Modelo de Velocidade constante

Figura 43: Desempenho do sistema sem realizar predição da posição - Modelo de Aceleração constante

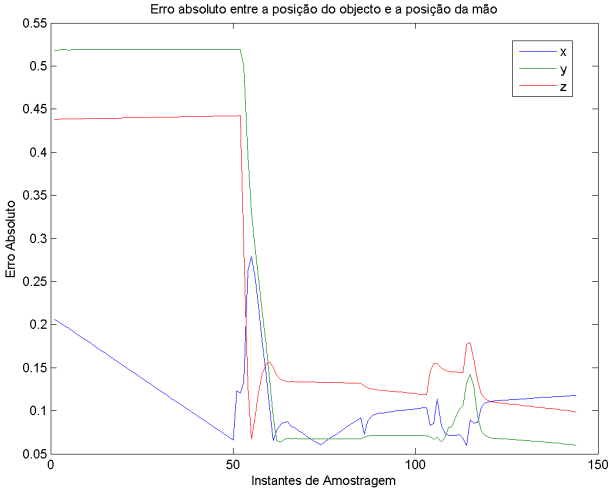
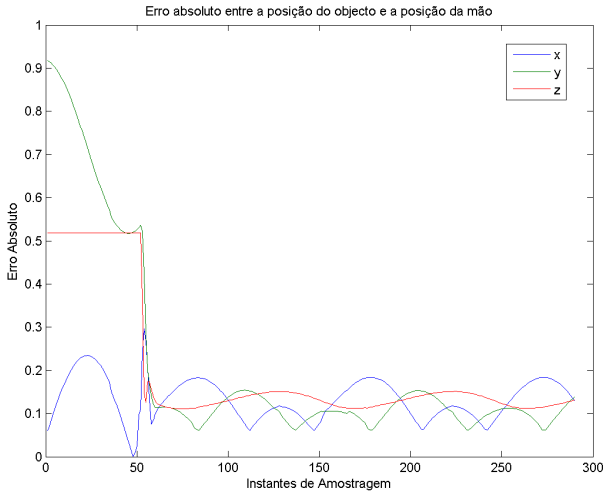


Figura 44: Desempenho do sistema sem realizar predição da posição - Modelos Curvilíneos



ligeira flutuação nas posições determinadas.

No gráfico 45, onde se apresenta a evolução temporal do horizonte de predição, é bem evidente a origem deste ruído através da constante comutação do horizonte de predição.

Convém ressalvar que na abordagem seguida a posição do braço num dado instante apenas influencia o horizonte de predição, não repercutindo qualquer efeito sobre as estimativas dos modelos.

Para tornar o ambiente de simulação no *webots* mais semelhante à situação presente no robot real, foi adicionado ruído na leitura da posição do móvel, tendo-se verificado um bom comportamento do sistema na presença de ruído até aos 10cm de desvio padrão.

Nas figuras 46 e 47, encontra-se a evolução das coordenadas 3D e da respectiva predição na presença de ruído branco, para o caso de um móvel com trajectória circular. O ruído introduzido tem desvio padrão de 6cm nas coordenas x e y e 12cm na coordenada z.

7.3 Resultados - Robot real

Nesta secção apresentam-se os resultados experimentais obtidos com o robot real.

Como era previsível os resultados obtidos com o robot são menos favoráveis que os obtidos com o simulador. O nível de ruído obtido na estimação da posição 3D é bem mais elevado que o existente no simulador, por um lado devido a falhas na segmentação da imagem, mas por outro devido também a problemas no *encoder* que dá a posição angular da camera esquerda.

Como é possível constatar em todas as experiências apresentadas a seguir, existe uma redução significativa do ruído da trajectória, possibilitando um comportamento mais estável no posicionamento do braço do robot.

Figura 45: Evolução do horizonte de predição

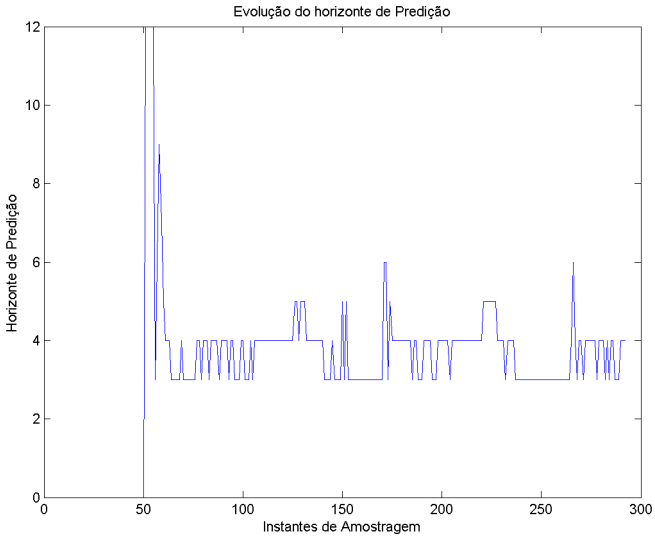


Figura 46: Posição do móvel na presença de ruído

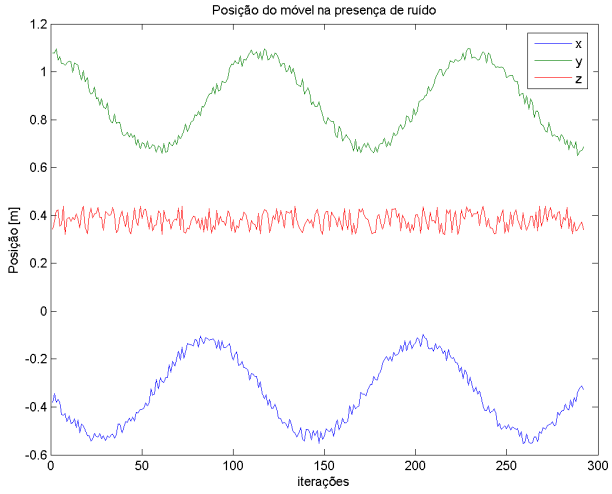
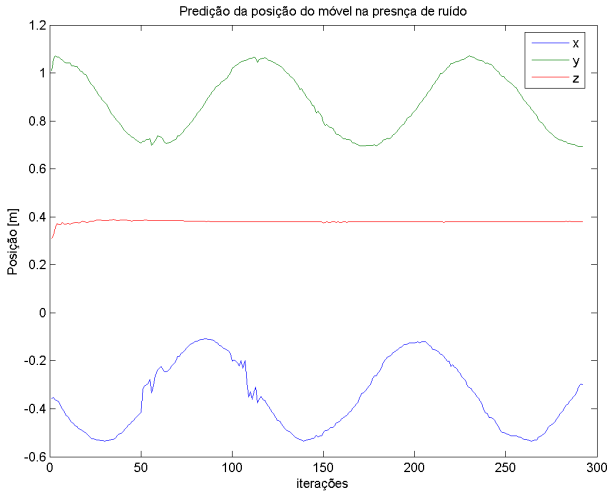


Figura 47: desempenho do estimador na presença de ruído



Num primeiro ensaio, que permite aferir o ruído do sistema, o objecto é simplesmente colocado numa posição fixa. Nos gráficos das figuras 48 é possível ver a evolução da estimação do ponto 3D e a localização da mão do robot.

Esta localização é calculada através da cinemática directa, sendo portanto também uma estimativa. Nos primeiros gráficos a estimação do ponto 3D é directamente utilizada na cinemática inversa, enquanto que nos restantes apresentados na figura 49 é utilizado o preditor desenvolvido.

A posição da mão utilizando o filtro e o preditor é bastante mais estável, não se tirando, no entanto qualquer contrapartida com o uso do preditor no que respeita ao tempo necessário para intercepção, uma vez que o objecto se encontra fixo.

Seguidamente é feito um ensaio com uma trajectória aproximadamente circular. Nesta experiência, para além da redução do ruído, é possível verificar um atraso de seguimento menor quando se utiliza o preditor, passando de cerca de 10 para 2 iterações.

No gráfico da figura 52 é possível verificar que apesar de o movimento não ser perfeitamente circular, o modelo que obtém maior peso é o curvilíneo, comutando inicialmente com o modelo de movimentos periódicos.

Em termos de erro quadrático médio entre as estimativas da localização do objecto e da mão verificou-se uma redução de 30% quando se utiliza o preditor. Para o movimento circular obteve-se um erro de 7.2cm e 4.8cm respectivamente para a versão sem preditor e com preditor.

Em todos os ensaios realizados existe um erro estático significativo entre a posição da mão do robot e o objecto devidos a dois factores. Um deles está relacionado com erros no modelo geométrico do braço, que se manifestam ao operar com o robot real. Nos dados apresentados não é possível no entanto verificar este

Figura 48: Evolução das coordenadas 3D do objecto e do punho do robot - sem predição

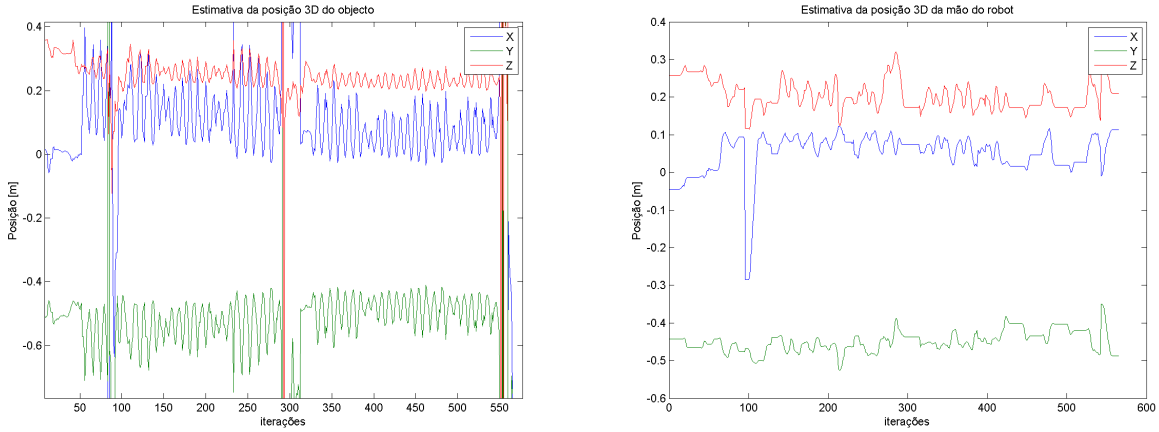


Figura 49: Evolução das coordenadas 3D do objecto e do punho do robot - com predição

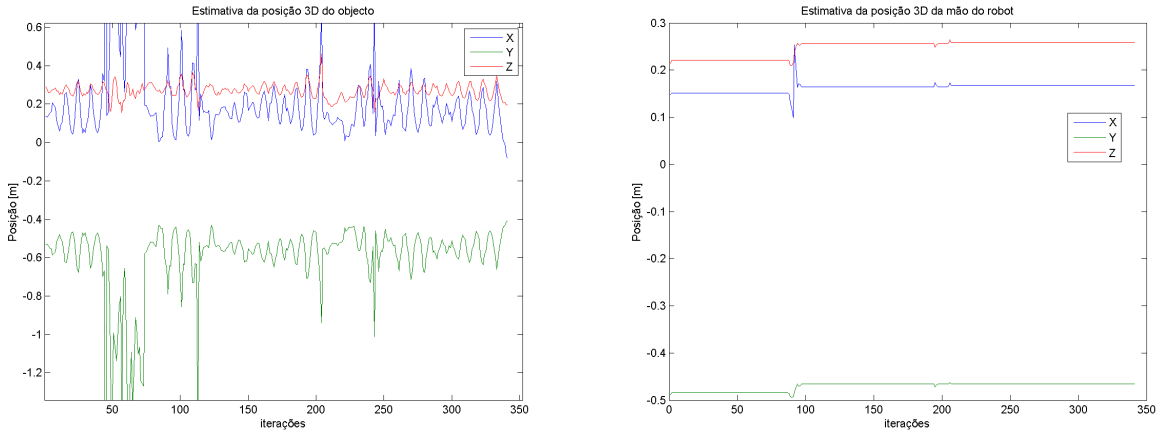


Figura 50: Evolução das coordenadas 3D do objecto e do punho do robot - trajectória circular - sem predição

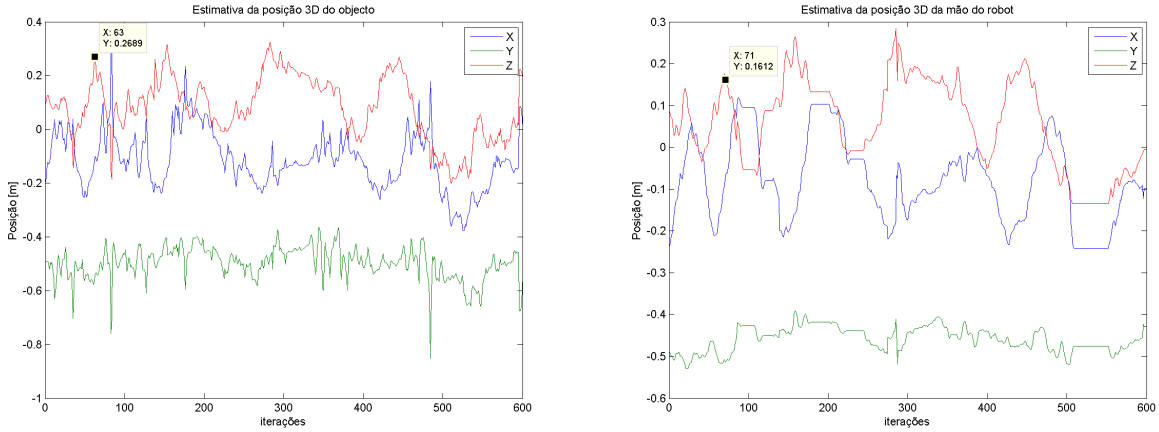


Figura 51: Evolução das coordenadas 3D do objecto e do punho do robot - trajectória circular - com predição

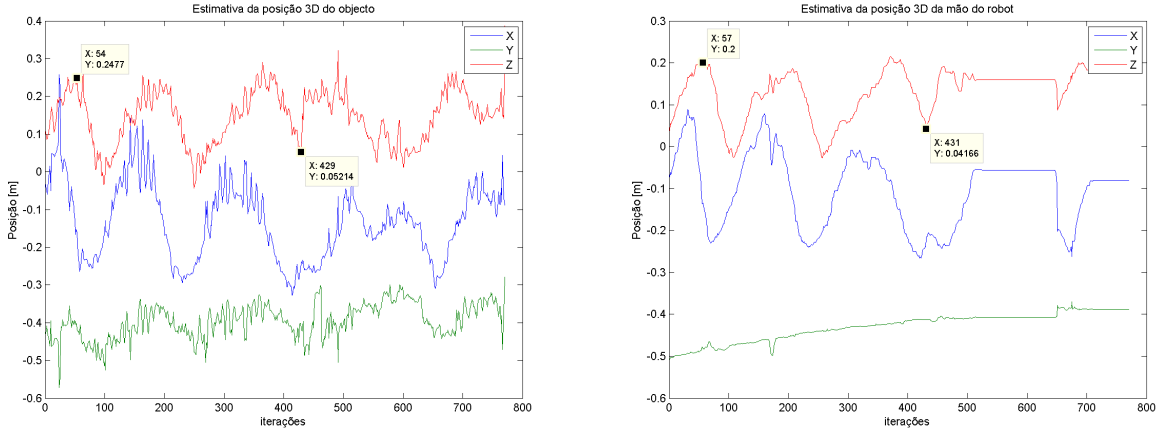
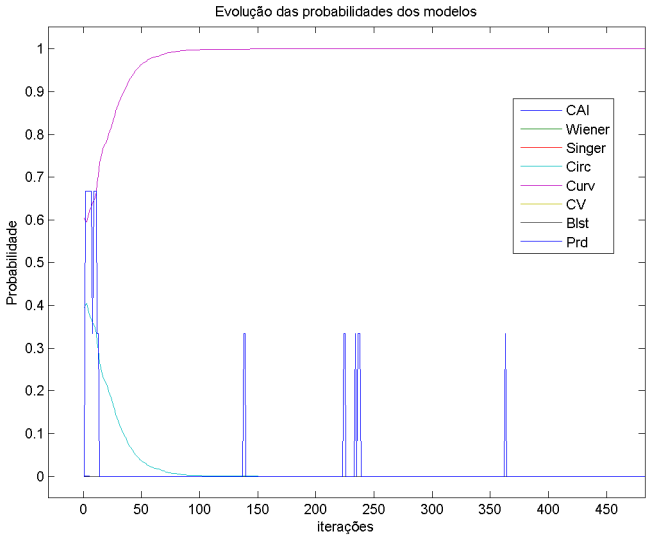


Figura 52: Evolução das probabilidades de cada modelo



erro uma vez que são compensados pelo cálculo da cinemática directa.

Outro dos erros é devido a política de actualização das juntas utilizada. Após realizar alguns ensaios verificou-se que o envio sucessivo de comandos provocava demasiadas oscilações no movimento do braço levando a alterações no funcionamento do algoritmo.

Para diminuir a frequência de envio de dados para o robot, todos os comandos para posições próximas da actual são ignorados. Para obter um melhor desempenho são utilizados diferentes limiares para a actualização dos valores das juntas de acordo com a proximidade do destino do comando, sendo exigida menor discriminação para pequenos movimentos.

Esta opção para controlo da frequência de envio, apesar de reduzir as oscilações do braço do robot, leva contudo à existência de erro no posicionamento do braço do robot.

8 Conclusão

O método apresentado provou ser eficaz, permitindo que o braço do robot alcançasse o alvo móvel. Destaca-se o papel fundamental dos filtros de kalman, por um lado no que respeita à supressão de ruído criado na reconstrução tridimensional da posição do móvel e por outro, no que respeita à possibilidade de predição da posição alguns instantes de tempo à frente.

O filtro de kalman é de facto uma ferramenta de grande utilidade que permite fazer a evolução do estado do sistema na presença de ruído. Apesar do filtro de kalman considerar apenas a presença de ruído branco, não se evidenciou a necessidade de utilizar ferramentas mais complexas para a estimação do vector de estado.

A configuração do *Baltazar* mostrou-se suficientemente flexível, permitindo a operação do robot num volume considerável. De facto o braço do robot, apesar de ser bastante mais simples que o braço humano permite uma boa cobertura do espaço adjacente.

O simulador desenvolvido e utilizado para testes foi bastante útil, tendo possibilitado a pré-validação das técnicas criadas, antes de as utilizar no robot real. Apesar de existir alguma discrepância entre os resultados obtidos no simulador e no robot real é possível, utilizando o simulador, obter uma ideia do desempenho que determinadas opções teriam no robot real, permitindo fazer o teste de conceitos.

O facto do *webots* permitir a programação dos controladores dos robots na linguagem C e C++ aliado às funcionalidades do *yarp* deu a possibilidade de adaptar o interface com o módulo de processamento, de tal forma que quase não é necessário fazer alterações quer se utilize o simulador ou o robot. Por outro lado é também bastante útil estarem já disponíveis um vasto conjunto de funcionalidades relacionadas com a simulação propriamente dita, nomeadamente no que respeita à simulação do regime dinâmico do sistema.

O protocolo de comunicações utilizado é bastante eficiente e permite efectivamente distanciar o equipamento directamente relacionado com o *hardware* do equipamento relacionado com o processamento, possibilitando um maior nível de abstracção no que respeita ao desenvolvimento dos algoritmos. De facto, é possível criar uma dada funcionalidade para um robot e utilizá-la num outro robot sem ser sequer necessário alterar o código fonte do programa, permitindo assim a reutilização do código. Este resultado é conseguido graças à modularidade com que todo o protocolo de comunicações com robots foi projectado.

Um aspecto possível de melhoramentos futuros é o do posicionamento do braço do robot numa dada posição. Apesar da simplicidade e sistematização patente nas soluções adoptadas, surge uma desvantagem relacionada com a necessidade de modelação do robot e da consequente introdução de erros. O problema surge contudo porque não é utilizado qualquer mecanismo de verificação do posicionamento realmente atingido, sendo apenas considerados os dados introduzidos *a priori*.

O algoritmo de segmentação utilizado, baseado no *meanshift* permitiu fazer o *tracking* dos objectos, tendo-se revelado bastante eficiente no seguimento de objectos com cores saturadas, mesmo nas situações que envolvem mudanças de *background*. Este algoritmo implementado com as funções optimizadas do *opencv* permite

fazer seguimento de objectos em imagens com resolução de $640X480$ pixels e 25Hz de frequência.
Uma das limitações desta operação está relacionada com as oclusões, não tendo sido tomada qualquer medida especial para resolver este problema. De facto, uma vez que está disponível uma estimativa da posição do móvel, seria interessante aplicá-la neste contexto, permitindo fazer uma estimativa do movimento mesmo na presença de obstruções.
A estratégia utilizada para combinação dos vários modelos de movimento baseados nas equações às diferenças do vector de estado também se revelou adequada, tendo sempre atribuído mais peso ao modelo cuja hipótese corresponde ao movimento do móvel. Ainda a este respeito convém sublinhar a redução no erro de seguimento conseguida com a combinação dos vectores de estado provenientes dos vários modelos quando comparado com o erro de seguimento obtido quando se utiliza um único modelo.
A adição de mais modelos de movimento aparenta à partida ser desnecessária, uma vez que os modelos já disponíveis têm alguma flexibilidade, modelada pelo ruído do modelo, que permitem a sua utilização noutros contextos. De facto, todos os movimentos possíveis são, considerando intervalos de tempo suficientemente pequenos, de velocidade constante. Para garantir uma maior flexibilidade especialmente para os movimentos mais bruscos é suficiente considerar os modelos de aceleração constante. Não fica contudo excluída a hipótese de acrescentar modelos mais específicos ao conjunto agora desenvolvido, à semelhança da situação dos movimentos periódicos. De facto é possível obter um desempenho melhor para esta vasta classe de movimentos utilizando modelos específicos.
O modelo de autocorrelação implementado apresenta algumas limitações, especialmente relacionadas com as taxas de actualização das estimativas. Este modelo garante um bom desempenho apenas na presença de movimentos perfeitamente periódicos, devendo-se optar, em melhoramentos futuros, por uma outra solução onde não seja necessário esperar a duração de uma janela para proceder à actualização da saída.
A técnica utilizada para a estimação do tempo de predição é suficiente para alcançar o objectivo da intercepção com o móvel, contudo é bastante limitada, não garantido a convergência para a solução óptima. Este estimador deveria na realidade ser melhorado a fim de permitir também a selecção de diversos modos para alcançar o móvel.
Apesar do trabalho proposto permitir ultrapassar as consequências que o regime dinâmico impõe no alcance de objectos móveis, nada faz no sentido de o controlar. De facto este regime continua a manifestar-se na íntegra, permitindo assim a presença de alguns aspectos negativos, nomeadamente no que respeita às oscilações do braço do robot provocadas pelo envio dos diferentes comandos para as juntas do braço. Seria interessante desenvolver uma técnica que minimizasse este efeito.

Referências

[1] M. Lopes, R. Beira, M. Praça e J. Santos-Victor, *An anthropomorphic robot torso for imitation: design and experiments*, IEEE, Intelligent Robots and Systems, Outubro 2004.

[2] J. Caenen, J. Angue, *Identification of Geometric and non Geometric Parameters of Robots*, IEEE, 1990.

[3] M. Lopes, *A Developmental Roadmap for Learning by Imitation in Robots*, Dissertação para Obtenção do Grau de Doutor em Engenharia Electrotécnica e de Computadores, Maio 2006

[4] Gary R. Bradski, *Computer Vision Face Tracking For Use in a Perceptual User Interface*, Microcomputer Research Lab, Santa Clara, CA, Intel Corporation.

[5] John G. Allen, Richard Y. D. Xu, Jesse S. Jin, *Object Tracking Using Camhhift Algorithm and Multiple Quantized Feature Spaces*, School of Information Technologies University of Sydney, NSW 2006.

[6] Jung-ho Lee, Woong-hee Lee, Dong-seok Jeong, *Object Tracking Method using Back-Projection of Multiple Color Histogram Models*, Inha University, South Korea

[7] J. Nascimento, J. Salvador Marques, *Adaptive Snakes Using the EM Algorithm*, IEEE, Transactions on Image Processing, Vol. 14, Issue 11, pp.1678-1686, 2005

[8] A. Pentland, B. Moghaddam, T. Starner, *View-Based and Modular Eigenspaces for Face Recognition*, Perceptual Computing Group, The Media Laboratory Massachusetts Institute of Technology

[9] P. Aguiar, J. Moura, *Detecting and Solving Template Ambiguities in Mot ion Segmentation*, IEEE, Transactions on Image Processing, Vol. 14, Issue 11, pp.1678-1686, 2005

[10] D. Cremers, A. Yuille *A Generative Model Based Approach to Motion Segmentation*, B. Michaelis Ed, Pattern Recognition, Magdeburg, Setembro 2003

[11] A. Abrantes, J. Salvador Marques, *The Mean Shift and the Unified Framework*, Proc. ICPR 2004 - 17th International Conference on Pattern Recognition, Cambridge, UK, Vol. 1, 244-247, 2004

[12] Alexandre Bernardino, José Santos Victor, *Binocular tracking: integrating perception and control*, IEEE, Transactions on Robotics and Automation, volume 15, Issue 6, Page(s):1080 - 1094, Dezembro 1999

[13] G. Welch, G. Bishop, *An Introduction to the Kalman Filter*, University of North Carolina at Chapel Hill, 2004.

[14] P. Maybeck, *Stochastic Models, Estimation, and Control*, Volume 1, Academic Press, Inc, 1979.

[15] R. Brown, P. Hwang, *Introduction to Random Signals and Applied Kalman Filtering, Second Edition*, John Wiley&Sons, Inc, 1992.

[16] O. Jacobs, *Introduction to Control Theory*, 2nd Edition, Oxford University Press, 1993.

[17] R. Singer, *Estimating Optimal tracking Filter Performance for Manned Maneuvering Targets*, IEEE, 1970.

[18] J. Boiffier, *The Dynamics of Flight: The Equations*, Wiley, 1998.

[19] S. Asseo, R. Ardila, *Sensor independent target state estimator design and evaluation*, National Aerospace and Electronics Conference, pp 916-924, 1982

[20] X. Rong Li, V. Jilkov, *Survey of Maneuvering Target Tracking. Part I: Dynamic Models*, IEEE, 2003.

[21] P. Hanlon, P. Maybeck, *Multiple-Model Adaptative Estimation Using a Residual Correlation Kalman Filter Bank*, IEEE, Transactions on Aerospace and Electronic Systems, vol. 36, no. 2, 2000.

[22] X. Li, Y. Bar-Shalom, *Multiple-model estimation with variable structure*, IEEE, Transactions on Automatic Control, vol. 41, no. 4, 1996

[23] P. Maybeck, *Stochastic Models, Estimation and Control, vol.2*, New York Academic Press, 1994.

[24] W. Schmaedeke,K. Kastella *Sensor Management using Discrimination Gain and Interacting Multiple Model Kalman Filters*, Lockheed-Martin Tactical Defense Systems, 1998.

[25] B. Wheaton,P. Maybeck *Second-Order Acceleration Models for an MMAE Target Tracker*, IEEE, Transactions on Aerospace and Electronic Systems, vol. 31, no. 1, 1995.

[26] K. Fisher, P. Maybeck, *Multiple Model Adaptive Estimation with Filter Spawning*, IEEE, Transactions on Aerospace and Electronic Systems, vol. 38, no. 3, 2002.

[27] X. Rong Li, V. Jilkov, *A Survey of Maneuvering Target Tracking - Part IV: Decision-Based Methods*, IEEE, 2003.

[28] O. Michael, *Webots: Professional mobile robot simulation*, International Journal of Advanced Robotic Systems, vol. 1, no.1, pp.39-42, 2004.

[29] P. Fitzpatrick, G. Metta, L. Natale, *YARP User Manual*, Abril 2007.