Departamento de Electrónica, Telecomunicações e Informática

Universidade de Aveiro

Sistema de Visão em Tempo-Real

Sistemas de Tempo-Real 07/08

Docentes:
Luís Almeida
Paulo Pedreiras

Realizado por:
28256 – Mauro Rodrigues

Introdução

Este trabalho surge no âmbito do desenvolvimento de um robô humanóide na Universidade de Aveiro, um projecto de colaboração entre o Departamento de Mecânica e o Instituto de Engenharia Electrónica e Telemática de Aveiro, e da disciplina de Sistemas de Tempo-Real. O humanóide tem como meta, a participação na *Robocup*, categoria *Penalty Kick*. É objectivo deste documento relatar a implementação do Sistema de Visão em Tempo-Real do humanóide.

Para a implementação do sistema é necessária a utilização das funcionalidades de tempo-real uma vez que as tarefas associadas a este sistema têm de ser executadas dentro de um intervalo de tempo finito, não interessa processar imagens capturadas há muito tempo pois o mundo real já se alterou entretanto.

Objectivos

Criação de um sistema de Tempo-Real para um Robô Humanóide:

1. Enquadramento do problema.
2. Definição e modularização do processo.
3. Escalonamento Tempo-Real.
4. Discussão de eventuais conflitos com outras tarefas do humanóide.
5. Comparação com uma implementação Monolítica.

Enquadramento

A plataforma é um Robô Humanóide desenvolvido na Universidade de Aveiro. Este possui como características principais: 22 graus de liberdade, 60 cm de altura, uma massa total de 6 Kg, arquitectura distribuída de controlo constituída por três tipo de unidades – *CPU, Master e Slaves* –, formando uma rede de controladores interligada por um barramento CAN em configuração *Master/Multi-Slave*. É ainda importante referir que todo o é do tipo *embedded*, sendo o *CPU* um processador *standard* PC/104+ com a seguinte configuração – *AMD Geode* LX800@500MHz, 512Mb *RAM, SSD* 1Gb.

Figura 1. PC/104+

A captura do sinal de vídeo é feita através de uma câmara *UniBrain Fire-i* @ 30fps (640x480) ligada a uma placa *PCMCIA FireWire*.

Figura 2. Câmara Unibrain Firewire e Módulo PCMCIA

O sistema operativo é o GNU/Linux *Debian* 40r0 i386 *Net Install*. Para efeitos de programação está a ser usada a linguagem C/C++ e as bibliotecas OpenCV (processamento imagem/vídeo) e PMan (funcionalidades tempo-real).

Definição e modularização do processo

A arquitectura definida para o sistema é a que está representada na figura abaixo (Fig. 3).
Esta consiste em quatro tarefas, *Image Acquisition*, *Image Processing*, *Object Tracking* e *Control*. A primeira delas é a tarefa que faz o *trigger* do sistema, ou seja, após a aquisição da imagem, as tarefas seguintes são despertadas e executam a sua função.

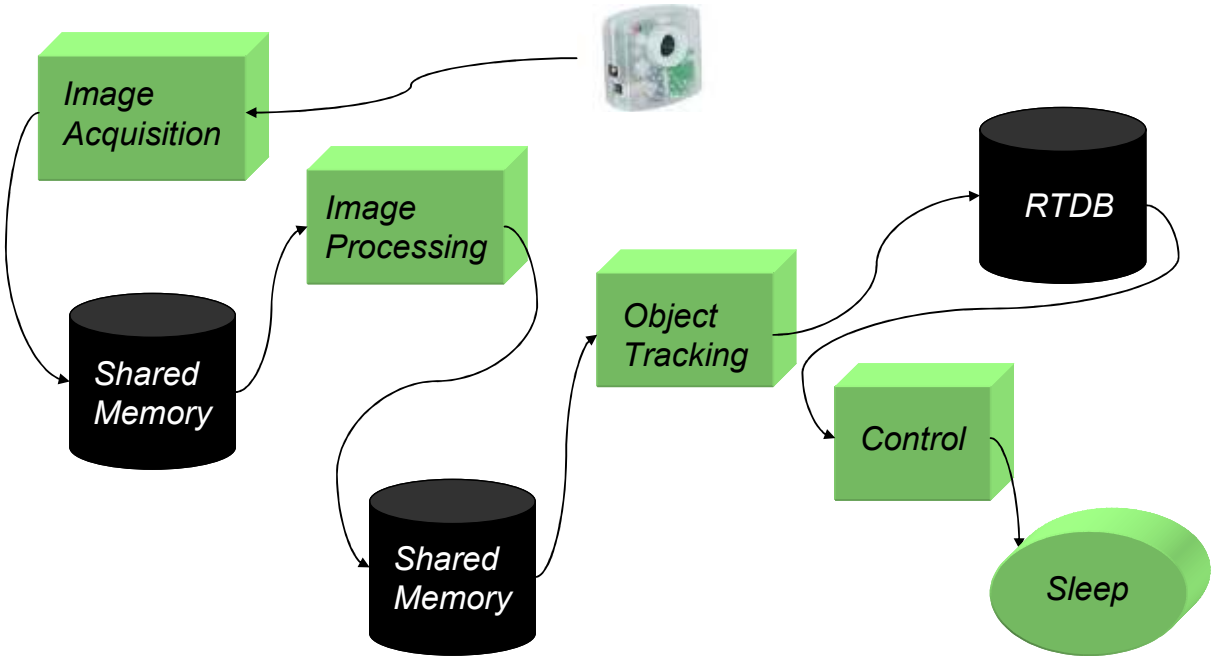

Figura 3. Arquitectura do Sistema de Visão

Image Acquisition

A tarefa *Image Aquisition* é, como já foi referido, o *tick* do sistema, activa os processos seguintes quando uma nova imagem está disponível. A imagem adquirida é armazenada numa *shared memory* (SHM_ATOP) para posterior utilização por parte de outros processos.

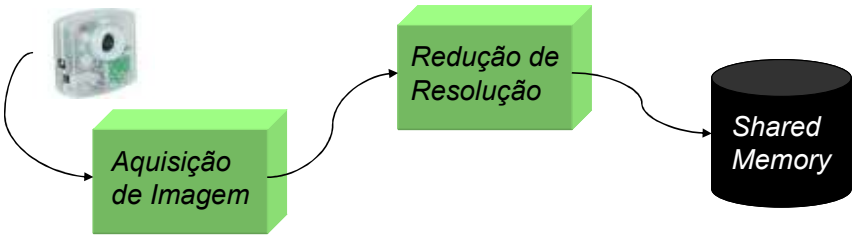

Figura 4. Tarefa de aquisição de imagem

Image Processing

Image Processing recolhe da *shared memory* referida anteriormente (SHM_ATOP) a imagem armazenada e efectua o respectivo processamento de cor, a imagem em RGB é convertida para HSV e posteriormente dividida nas suas componentes H, S e V. Os parâmetros da máscara, previamente ajustados com a tarefa auxiliar *Color Calibration*, são então aplicados a cada componente e estas são de seguida combinadas formando a mesma. Esta máscara é então armazenada numa outra *shared memory* (SHM_PTOT).

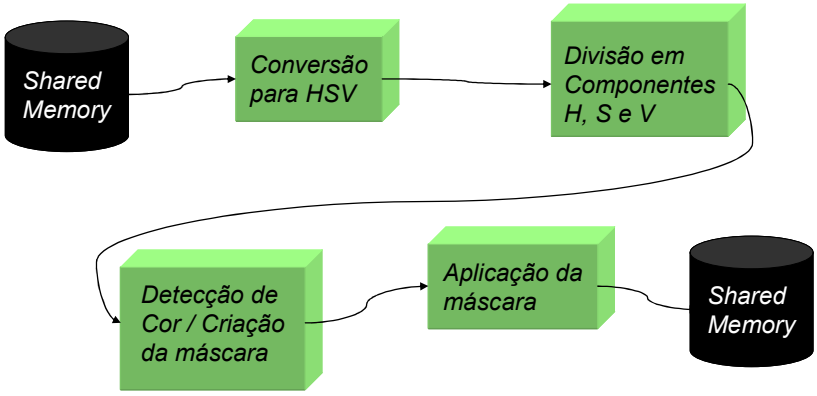

Figura 5. Tarefa de processamento de imagem

Object Tracking

Object Tracking faz a detecção da bola na imagem e respectiva validação através da detecção de círculos na imagem filtrada. É calculado o centro de massa do objecto detectado e determinada a sua localização relativa na imagem, estes parâmetros são então armazenados na base de dados tempo-real (SHM_RTDB).
A sequência dos eventos do processamento de imagem está representada na Figura 7.

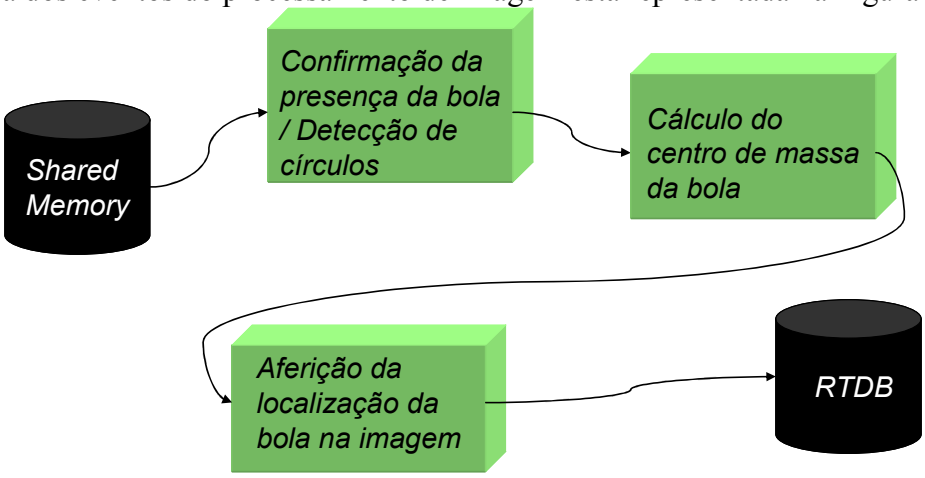

Figura 6. Tarefa de localização da bola

A figura seguinte ilustra os passos desde a aquisição da imagem até à determinação dos parâmetros do objecto, passando criação da máscara através das componentes H, S e V da imagem.

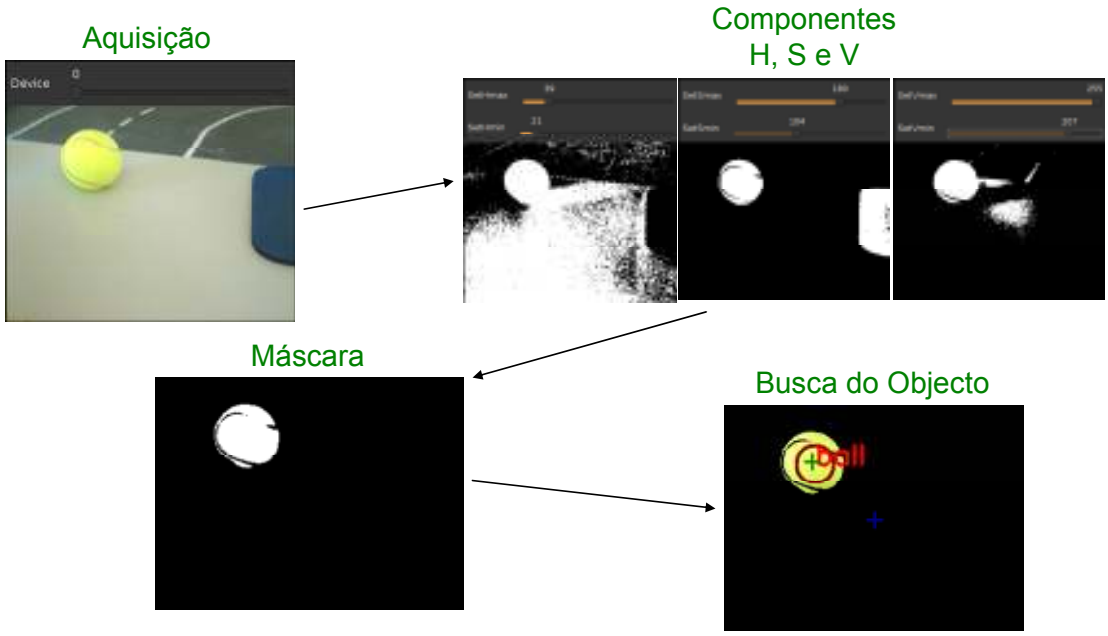

Figura 7. Sequência de processamento de imagem

Control

A tarefa *Control* recolhe da RTDB, os parâmetros calculados pela tarefa *Object Tracking* e faz uma validação dos valores. De seguida, são calculados os parâmetros do movimento a efectuar, por ex., uma deslocação da bola no plano da imagem, corresponde a uma combinação de movimentos das juntas do pescoço necessários para que o humanóide possa seguir a bola com a cabeça (câmara), esta transformação tem o nome Cinemática Inversa.
Nota: O algoritmo de Cinemática Inversa não foi utilizado para a aquisição de resultados pois não se encontra testado devido a problemas de *hardware* ainda não resolvidos. Foi em alternativa utilizado um algoritmo proporcional já anteriormente testado.

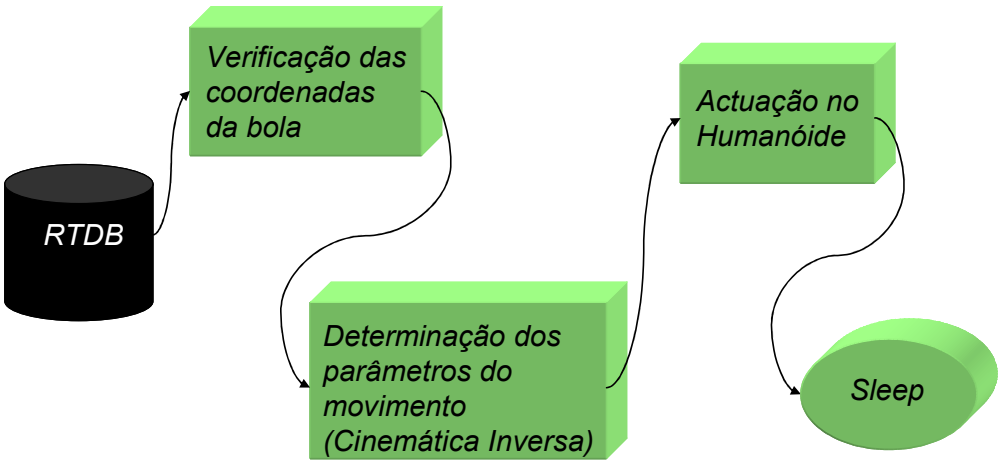

Figura 8. Tarefa de actuação e controlo do Humanóide

Color Calibration

Color Calibration é uma tarefa auxiliar independente que permite calibrar os parâmetros que serão posteriormente utilizados no processamento de imagem. Este processo é efectuado com recurso a janelas com barra de deslocamento que fazem o ajuste "*on-line*" de cada componente (Fig. 9). No final da execução os parâmetros são salvos para um ficheiro de texto chamado *hsv_param.txt*.

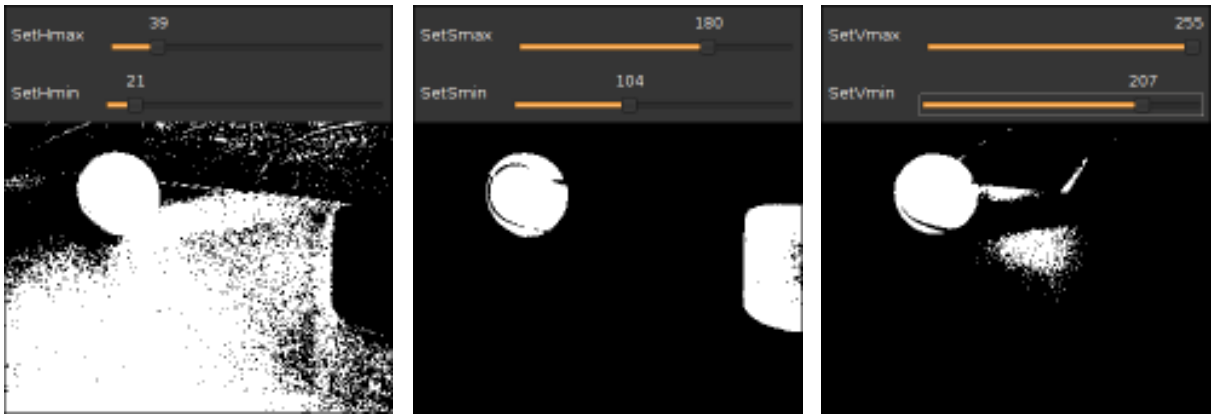

Figura 9. Janelas de calibração

Escalonamento

Precedências

As precedências foram definidas como está ilustrado na figura abaixo. A sequência das tarefas é lógica, captura da imagem, processamento da mesma, extracção dos parâmetros do objecto alvo e finalmente actuação no humanóide concordante com a informação recolhida e tarefa a executar, no caso específico, seguir uma bola.

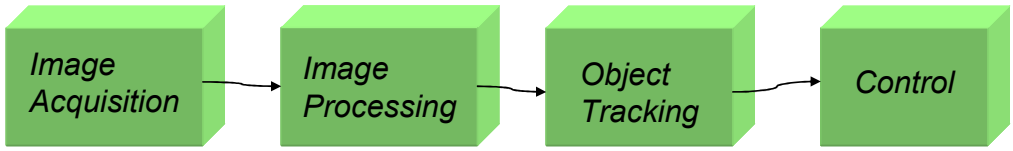

Figura 10. Esquema de precedências

Prioridades

As prioridades foram atribuídas usando o critério "tarefa predecessora, maior prioridade". É de notar que em Linux maior prioridade significa menor valor, ou seja, quanto mais próximo de 0 (zero) for o valor da prioridade de uma tarefa, mais prioritária ela se torna.

A Tabela 1 contém um resumo da caracterização das tarefas relativamente ao escalonamento.

Processo	Período	Lista de Precedências	Prioridade	Descrição
Image Acquisition	1	-	-	Interface com a câmara. *Tick* do sistema.
Image Processing	1	*Image Acquisition*	30	Classificação de cor.
Object Tracking	1	*Image Processing*	40	Busca do objecto (bola).
Control	1	*Object Tracking*	50	Execução do controlo no Humanóide

Tabela 1. Classificação e descrição das tarefas

Apresentação de resultados

Versão Monolítica

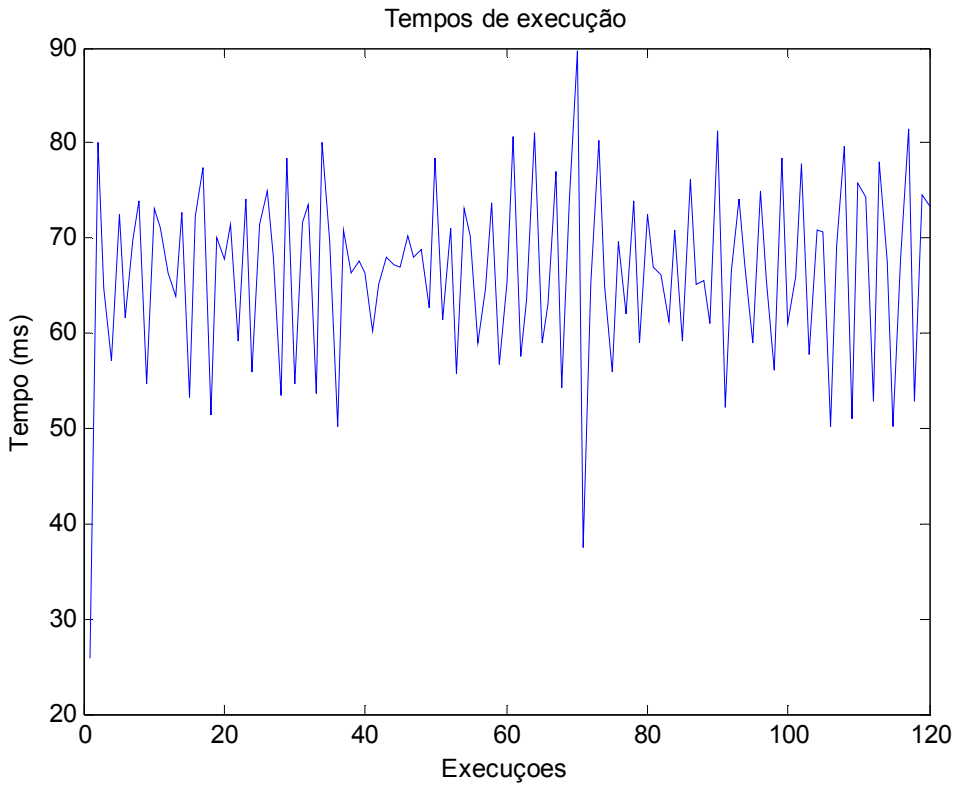

Figura 11. Execução da tarefa *im_acq*

Tempos de Execução (ms)	
Mínimo	25,822
Máximo	89,599
Médio	66,4272
Desvio Padrão	9,6604

Tabela 2. Tempos relativos à execução da versão monolítica

Versão Distribuída

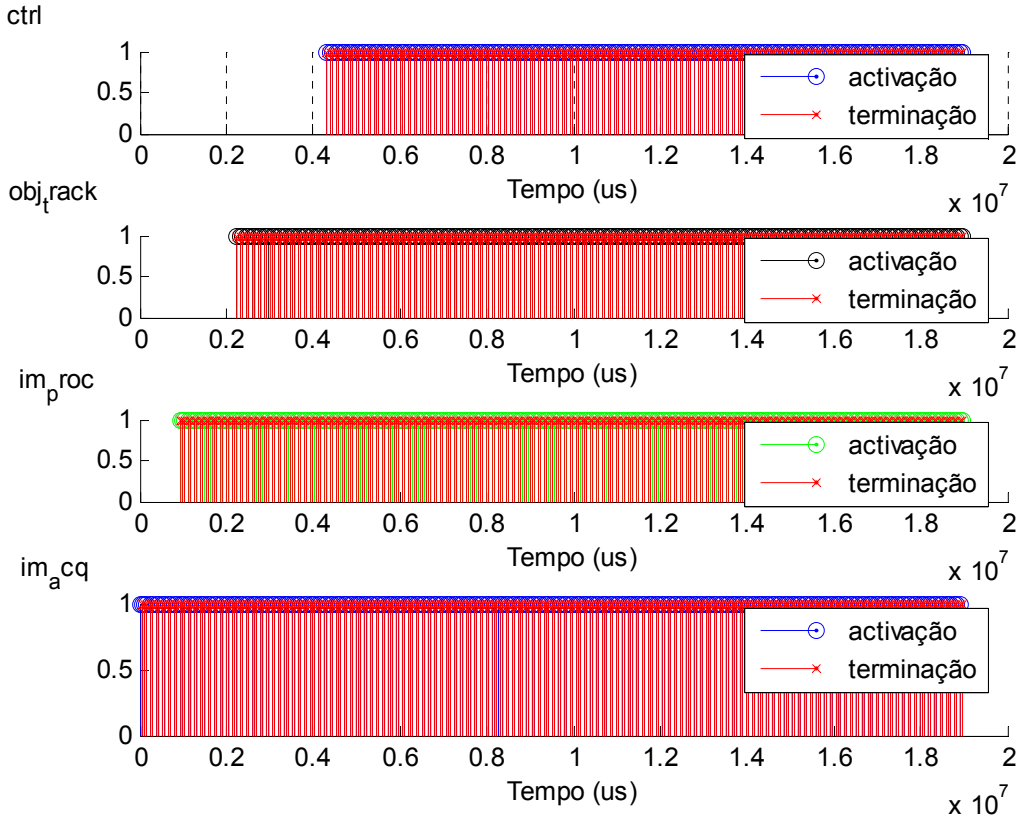

Figura 12. Evolução temporal das tarefas

Tempos de Execução (ms)				
	im_acq	im_proc	obj_track	ctrl
Mínimo	55,8850	6,1850	0,0000	0,0045
Máximo	78,0790	9,5440	0,5510	0,0057
Médio	66,5484	6,3977	0,1649	0,0030
Desvio Padrão	3,8843	0,3613	0,2148	0,083

Tabela 3. Tempos de execução relativos à execução individual de cada uma das tarefas do processo

Conclusão

Verifica-se na versão monolítica que o tempo médio de execução de cada ciclo (~67ms) é cerca do dobro do tempo que demora a adquirir uma nova imagem (30 fps@(640x480) ~ 33ms entre aquisições). Isto significa que a aplicação não aproveita as potencialidades da câmara.
Passando para a versão distribuída verificamos que o tempo médio de execução da tarefa de aquisição é semelhante ao tempo da versão monolítica e os tempos das tarefas subsequentes são muito inferiores quando comparados com o anterior. A diferença que se obtém entre versão monolítica e distribuída aponta para uma pior performance da versão distribuída. Isto pode ser explicado devido a mudanças de contexto e preempções originadas pela concorrência entre tarefas, e ainda acessos a zonas de recursos partilhadas.

Podemos inferir que no estado actual de implementação do Humanóide, que executa um único processo – procura e seguimento da bola, a solução distribuída não será vantajosa.
É necessário estudar futuramente o impacto desta implementação quando se partir para a execução de múltiplos objectivos, procura da bola, baliza, linhas, etc.
Na tarefa de controlo, como já foi referido, o algoritmo de Cinemática Inversa não foi utilizado para a aquisição de resultados pois não se encontra testado devido a problemas de *hardware* ainda não resolvidos (controlo das juntas do pescoço! - Anexo I). Foi em alternativa utilizado um algoritmo proporcional já anteriormente testado. A utilização da Cinemática Inversa irá introduzir algum atraso no processamento global pois recorre a funções trigonométricas, mais complexas de tratar, estando prevista a utilização de funções trigonométricas personalizadas e recorrer a tabelas de ângulos por forma a minimizar este atraso.
É de referir que os testes foram efectuados no computador portátil do autor pois devido à limitada capacidade de processamento da plataforma final, foi decisão do mesmo efectuar os testes preliminares naquele. Por exemplo, fazer o *debbug* do processamento de imagem com recurso a janelas de visualização é uma tarefa que se torna pesada, sendo preferível fazê-lo externamente e posteriormente correr lá uma versão mais madura sem interface gráfica, o que reduz consideravelmente a capacidade de processamento exigida.
Relativamente a tarefas *não-visão* do Humanóide, este aspecto ainda não foi considerado mas será tido em conta futuramente incluindo rotinas de verificação de estado para posterior controlo, por ex., em casos de desequilíbrio.
Para finalizar é importante recordar quais os novos passos a efectuar no futuro próximo:

- Refinamento da tarefa de processamento de imagem;
- Determinação do *bug* de *hardware* e teste do algoritmo de controlo das juntas do pescoço – Cinemática Inversa;
- Execução de testes para determinar o impacto temporal na execução;
- Inclusão das rotinas de verificação de estado do Humanóide e respectivo controlo;

Nota: devido à extensão do código criado, este não será introduzido em anexo, seguindo apenas no ficheiro zip que contém todos os elemento do trabalho.

Bibliografia

- Pedreiras, Paulo e Almeida, Luís; "*Task Management for Soft Real-Time Applications Based on General Purpose Operating Systems*"; 2007

- Pedreiras, Paulo et. al.; "*Enhancing the reactivity of the vision subsystem in autnonomous mobile robots using real-time techniques*"; 2005

- Silva, Filipe M., "*Capítulo 4 – Fundamentos de Cinemática (Robótica)*", 2002-2003

Anexo I – Cinemática Inversa do Humanóide

A cinemática inversa obtém-se invertendo a matriz Jacobiana da cinemática directa. A cinemática directa relaciona os movimentos nas juntas com a deslocação no plano de imagem.

$$\begin{bmatrix} \dot{x}_3 \\ \dot{y}_3 \end{bmatrix} = J \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

Equação 1. Equação da Cinemática Directa

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = k.J^{-1} \begin{bmatrix} \dot{x}_3 \\ \dot{y}_3 \end{bmatrix}$$

Equação 2. Cinemática Inversa

em que J é a matriz Jacobiana

$$J = \begin{bmatrix} \frac{\partial x_3}{\partial \theta_1} & \frac{\partial x_3}{\partial \theta_2} \\ \frac{\partial y_3}{\partial \theta_1} & \frac{\partial y_3}{\partial \theta_2} \end{bmatrix}$$

Equação 3. Jacobiano

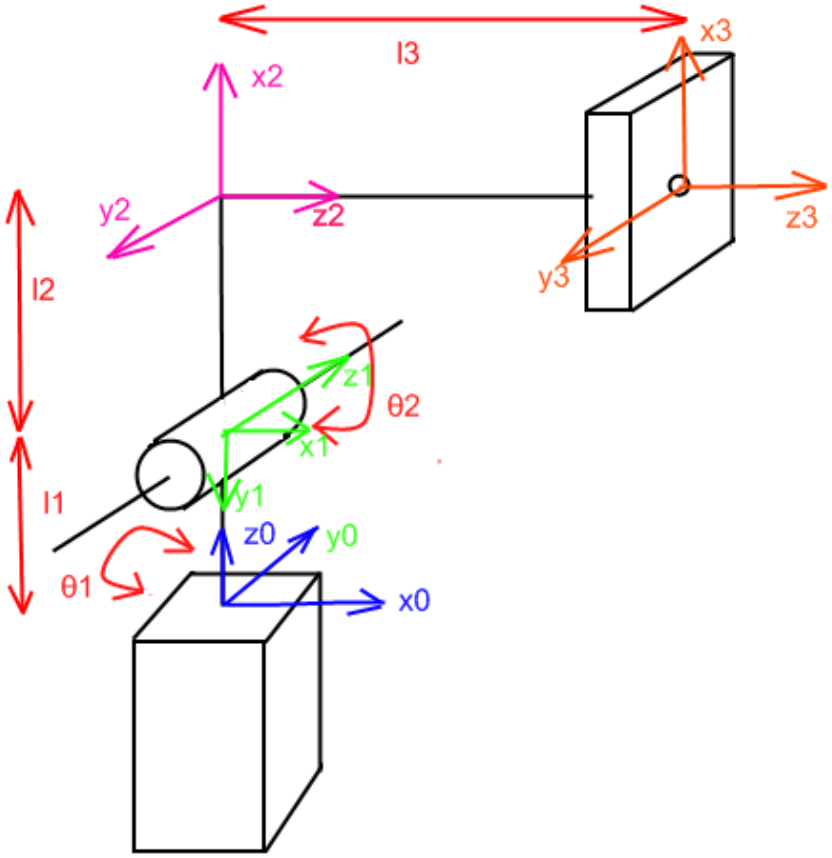

Ilustração 1. Esquema do pescoço do Humanóide

Cinemática Directa – Algoritmo de Denavit-Hartenberg

Transformações entre sistemas de eixos:

	θi	li	di	αi
i	Rot(Zi)	T(xi+1)	T(Zi)	R(xi)
1	θ1	0	l1	-90º
2	θ2-90º	l2	0	-90º
3	0	0	l3	0

Nota: Nos passos seguintes Ci refere-se a cos(θi) e Si a sin(θi).

$$A_i = Rot_z(\theta_i)\cdot T(l_i,0,0)\cdot T(0,0,d_i)\cdot Rot_x(\alpha_i)$$

$$A_1 = \begin{bmatrix} C1 & 0 & -S1 & 0 \\ S1 & 0 & C1 & 0 \\ 0 & -1 & 0 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ A_2 = \begin{bmatrix} S2 & 0 & C2 & l_2\times S2 \\ -C2 & 0 & S2 & -l_2\times C2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2\cdot A_3 = \begin{bmatrix} S2 & 0 & C2 & C2\times l_3 + l_2\times S2 \\ -C2 & 0 & S2 & S2\times l_3 - l_2\times C2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_1\cdot A_2\cdot A_3 = \begin{bmatrix} \dots & \dots & \dots & C1(C2\times l_3 + l_2\times S2) \\ \dots & \dots & \dots & S1(S2\times l_3 - l_2\times C2) \\ \dots & \dots & \dots & -(S2\times l_3 - l_2\times C2) + l_1 \\ \dots & \dots & \dots & \dots \end{bmatrix}$$

$$\begin{cases} x_3 = C1(C2\times l_3 + l_2\times S2) \\ y_3 = S1(S2\times l_3 - l_2\times C2) \\ z_3 = -(S2\times l_3 - l_2\times C2) + l_1 \end{cases}$$

$$J = \begin{bmatrix} \frac{\partial x_3}{\partial \theta_1} & \frac{\partial x_3}{\partial \theta_2} \\ \frac{\partial y_3}{\partial \theta_1} & \frac{\partial y_3}{\partial \theta_2} \end{bmatrix} = \begin{bmatrix} -S1(l_3\times C2 + l_2\times S2) & C1(l_3\times S2 + l_2\times C2) \\ C1(l_3\times C2 + l_2\times S2) & S1(l_3\times S2 + l_2\times C2) \end{bmatrix}$$

$$J^{-1} = \begin{bmatrix} \frac{-S1}{l_2\times S2 + l_3\times C2} & \frac{C1}{l_2\times S2 + l_3\times C2} \\ \frac{C1}{l_2\times C2 - l_3\times S2} & \frac{S1}{l_2\times C2 - l_3\times S2} \end{bmatrix}$$