A. Fernando Ribeiro

Grupo de Automação e Robótica, Departamento de Electrónica Industrial, Universidade do Minho, Campus de Azurém, 4800 Guimarães, Portugal
fernando.ribeiro@dei.uminho.pt

VIRTUAL SENSORS FOR AUTONOMOUS MOBILE ROBOTS THROUGH THE USE OF IMAGE PROCESSING TOOLS

RESUMO

Nos dias que correm existem muitas competições de robótica móvel dos mais variados tipos. Na maior parte dos casos, as equipas que participam utilizam as mesmas plataformas móveis para participar em diferentes competições precisando apenas de acrescentar/alterar alguns sensores, apesar de o tempo normalmente disponivel ser relativamente curto. A sua implementação pode ser algo demorado e mais complexo do que se previa, para além de necessitar de algum tempo para debugging. Este artigo descreve um conjunto de ferramentas de processamento de imagem para criar sensores virtuais através de visão por computador. Utilizando apenas uma simples camara e aplicando algumas destas ferramentas, estes sensores virtuais podem ser facilmente criados através de uma linha de instrução escrita num ficheiro de configuração, não sendo mesmo necessário a recompilação do código. Estes sensores virtuais são extremamente flexiveis e os seus parâmetros são facilmente editáveis. São aqui descritas estas ferramentas, a estrutura de dados de cada sensor, assim como alguns exemplos. Estas ferramentas foram já testadas com sucesso.

ABSTRACT

Many mobile robotics competitions exist nowadays. In most cases many teams use a unique mobile platform to participate in different competitions being only necessary to adapt different sensors, although this has to be carried out in a very short period of time. Its implementation through electronics can take some time and some debug, which is not always easy and can take some precious time. This paper describes a new set of image processing tools for creating virtual sensors through the use of computer vision. By using a single camera with these tools, these virtual sensors can be easily created through a line of code written in a configuration file, avoiding even the need for computer program recompilation. These virtual sensors are very flexible and the parameters are easily and quickly edited. This paper describes these tools, the data structure that each sensor uses, and describes some examples.

KEY WORDS: Digital Image Processing, software tools, autonomous mobile robotics, virtual sensors

1. INTRODUÇÃO

Há cada vez mais competições de robótica móvel, não apenas para demonstrações lúdicas, mas também para fomentar a investigação científica com um desafio único onde se podem comparar diferentes soluções para o mesmo problema, ou ainda para motivar o estudo em alunos com um problema prático de engenharia. O resultado dessa investigação científica pode e é normalmente aplicado na indústria. Exemplos de competições são vários tais como o RoboCup, Eurobot, Festival Nacional de Robótica, Micro-rato, etc.

Na maior parte destas competições, apesar de o objectivo ser distinto e das regras serem completamente diferente, os robôs usados funcionam segundo um mesmo princípio. Quanto maiores forem os requisitos de um robô e a sua flexibilidade, maior será o número de sensores necessários para que ele execute a sua tarefa com a melhor performance possivel. A autonomia é também um dos
maiores problemas nos robôs móveis. Para isso são precisos vários sensores para a percepção dos mais variados tipos de informação.

Algumas das equipas de robótica, de modo a optimizar o tempo de execução de projectos, utilizam a mesma plataforma móvel para diferentes competições, substituindo apenas alguns sensores e corrigindo o software de controlo e estratégia, para além de acrescentar drivers para os novos sensores. Esta tarefa implica sempre dispendio de algum tempo na criação de hardware e debug, o que por vezes se torna mais extenso do que o desejado.

Para simplificar a fusão sensorial e reduzir a aplicação de um grande número de sensores, pode por vezes utilizar-se uma (ou mais) câmaras. A partir de uma imagem adquirida é depois necessário retirar toda a informação necessária, e que substitui os sensores tradicionais.

A pensar nestes problemas, foi desenvolvido um conjunto de ferramentas de processamento de imagem capazes de criar sensores virtuais através de visão por computador. Estas ferramentas são standard, parametrizáveis através de um ficheiro de texto extremamente simples, e não necessitam de recompilação de software. Com a utilização destas ferramentas, reduz-se o tempo de criação de hardware, permite acrescentar novos sensores completamente parametrizaveis ao robô num curto espaço de tempo, permitindo concentrar mais o tempo nas áreas de investigação realmente desejáveis.

2. SENSORES TRADICIONAIS

A participação em várias provas de robótica móvel, permitiram adquirir experiência em vários campos, como a utilização de vários sensores e sua fusão sensorial. Dar autonomia a um robô é uma tarefa que exige uma elevada capacidade sensorial e computacional.

De prova para prova tornava-se necessário acrescentar novos sensores, o que implicava a criação de novo hardware, novos testes à sua fiabilidade, novo software para fusão sensorial, etc. e tudo isto pode demorar bastante mais tempo do que o disponivel.

Os sensores mais comuns nos eventos de robótica móvel e autónoma, são:

- **Infra-vermelhos** – para detecção de distâncias, de presença de obstáculos, para localização, para comunicação sem fios, etc.
- **Sonars** – para detecção de distâncias, de obstáculos, para localização, etc.
- **Interruptoress** – Para detecção de colisões, ou presença de objectos.
- **Emissores/Receptores de rádio** – para comunicações sem fios, para localização, etc.
- **Lasers** – para detectar obstáculos, localização do robô, etc.

Todos estes sensores têm vantagens e desvantagens, tais como:

- **Infra-vermelhos** – são direccionais, a distância perceptivel é relativamente limitada, fraca reflexão de alguns materiais ou cores, confundem-se quando outros robôs também os usam mesmo que para outros fins, etc.
- **Sonars** – problemas com os ecos, ou com sonares de outros robôs, etc.
- **Interruptores** – Dão resposta apenas após o contacto.
- **Emissores/Receptores de rádio** – frequências limitadas, frequências autorizadas variam de país para país, etc.
- **Lasers** – bastante caros, perigo na sua utilização, etc.

Esta lista não pretende ser exaustiva, nem serve de referência, apenas identifica alguns dos principais problemas que surgem quando são utilizados estes sensores nestas situações.
Mas existe uma caracteristica geral a todos estes sensores que consiste na necessidade de alguma electrónica adicional, para além de uma conversão de valores Analógicos para Digitais, para poderem ser interpretados pelo processador do robô.
Mesmo conhecendo as vantagens e desvantagens da utilização de processamento de imagem, na maior parte dos casos, a aquisição podia ser feita através de visão. Com a utilização de uma câmara, o
problema fica por vezes mais simplificado, visto que necessita apenas de uma placa de captura de dados (tarefa simples mesmo para pessoas que não da área da electrónica ou informática).
Há mesmo câmaras que não precisam de placa de captura (ISA ou PCI), sendo o seu interface através de uma porta USB, embora estas câmaras não tenham ainda a mesma qualidade de imagem, nem a mesma velocidade de aquisição (número de frames por segundo) que as tradicionais.
Visão deste tipo é actualmente utilizada em vários robôs futebolistas (quer em Portugal [1], [2], [3], quer no estrangeiro [4], [5], [6]) para leitura de informação vária. Este processo não invalida a utilização de outros sensores, apenas reduz essa necessidade.
Mas os problemas relacionados com o futebol robótico não são apenas os da visão mas muitos outros tal como descrito em [7].
No trabalho apresentado e descrito neste artigo, a câmara utilizada é uma SONY a cores com a referência TC-5173 sistema PAL, com uma lente Canon TF2812 1:1.2. A Frame Grabber tem um chip Bt848 e consegue 50 frames por segundo, tem entrada S-Video e Video Composto. O computador utilizado para fazer e testar estas ferramentas de processamento de imagem tem um processador de 200 MHz MMX e 16 Mbytes de memória. O tamanho do disco não é muito relevante pois a aplicação foi escrita em Linguagem C e Assembly para o sistema operativo MS-DOS.
A razão da utilização de uma motherboard relativemente lenta, foi precisamente para testar estas ferramentas em máquina lentas. Em qualquer microprocessador de velocidade superior os resultados serão sempre superiores.

3. DEFINIÇÃO DAS CORES - THRESHOLDS

Torna-se necessário definir as cores possíveis de utilizar nestas ferramentas.

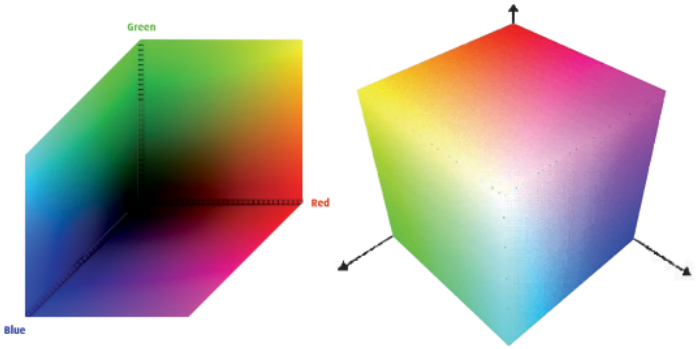

Figura 1 Planos cartesianos de cores RGB e Cubo RGB.

A captura de imagem é feita por estas ferramentas no formato RGB (Red, Green, Blue), ou seja três componentes de cor (Vermelho, Verde, Azul) de um byte cada (8 bits), o que prefaz um total de 24 bits, permitindo 224 cores diferentes (16,777,216) e que compõem o cubo RGB descrito na figura acima. Por razões óbvias, as cores disponíveis nestas ferramentas são as cores obtidas com as componentes no seu máximo ou no seu mínimo e suas respectivas combinações. Isso prefaz 23 cores sendo elas o vermelho (R), verde (G), azul (B), azul marinho (C), roxo (M), amarelo (Y), preto (K) e branco (W). Mas para uma destas 8 cores ser ideal, ela teria os valores máximos (255) ou mínimos (0) nas suas três componentes RGB (ver figura seguinte).

R	G	B	
255	0	0	R
0	255	0	G
0	0	255	B
0	255	255	C
255	0	255	M
255	255	0	Y
0	0	0	K
255	255	255	W

Figura 2 Composição das oito cores ideias através das suas componentes RGB.

Mas isso raramente acontece e assim foi necessário definir critérios para cada cor. Deste modo, uma cor é definida quando o valor das suas três componentes tem uma determinada diferença mínima, aqui identificada por Threshold. Por exemplo, para que um pixel seja considerado vermelho (R), a sua componente vermelha (R) tem de ser pelo menos maior que a componente G e B de um dado valor de Threshold. Assumindo um threshold de 120 para o vermelho, um pixel que tenha os valores RGB (210, 63, 80) é considerado vermelho, mas um pixel com os valores RGB (210, 63, 100) apesar de ter substancialmente mais vermelho na sua componente R do que nas outras duas, não é considerado vermelho, como descrito na figura seguinte.

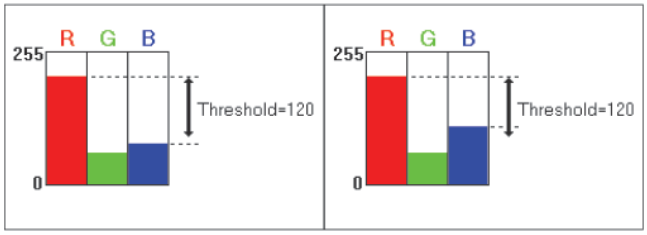

Figura 3 Determinação da cor de um pixel: esquerda) pixel é considerado vermelho, direita) pixel não é considerado vermelho.

Este threshold tem o mesmo significado quer para cores básicas R, G ou B (compostas por uma componente apenas), quer para cores que dependam de mais de uma componente. Se considerarmos o amarelo (Y), para uma valor de Threshold 150, o pixel com as componentes (205, 220, 23) é considerado amarelo, mas um pixel com as componentes (205,220,60) já não é considerado amarelo. No caso da cor branca, o Threshold funciona como o valor mínimo para qualquer uma das componentes, e no caso da cor preta o Threshold funciona como o valor máximo para cada uma das componentes.
Assim a definição dos vários Thresholds requer algum cuidado e pode variar de acordo com a quantidade de luz ambiente.

O valor destes Thresholds é definido no ficheiro de configuração do seguinte modo:

```
# Thresholds dos Filtros de cor (pode variar de 0-255)
RED=      100     // diferenca entre o R e os G,B
GREEN=     50     // diferenca entre o G e os R,B
BLUE=      90     // diferenca entre o B e os R,G
CYAN=     110     // diferenca entre o G, B e o R
MAGENTA=  105     // diferenca entre o R, B e o G
YELLOW=    90     // diferenca entre o R, G e o B
WHITE=    200     // valor mínimo para todos R,G,B
BLACK=      5     // valor máximo para todos R,G,B
```

4. TIPOS DE SENSORES

Esta ferramenta de processamento de imagem está em permanente desenvolvimento, mas actualmente permite a utilização de 3 tipos de sensores:

- Sensor de Cor – permite a localização de uma entidade através da sua cor
- Sensor de área – utilizado para detecção de obstáculos
- Sensor de Histograma – permite procurar características na imagem, como por exemplo uma posição de acordo com a sua distribuição de cor.

A definição dos sensores virtuais é feita num ficheiro de texto, o qual pode ser editado em qualquer altura sem ser necessário recompilar o código. Nesse ficheiro de configuração é ainda descrita a configuração da placa grafica (o modo ou resolução, e o endereço desta) bem como as características do cursor. Este ficheiro tem a extensão ".GAR" e todas as linhas que começam por # são consideradas comentários (linhas vazias são também ignoradas). A informação adquirida pelas rotinas de visão é guardada numa estrutura de dados de modo a poder utilizar-se no software de controlo dos robos.
Importa realçar que estas ferramentas foram desenvolvidas em linguagem C e Assembly para optimizar o tempo de processamento o qual pode ser bastante crítico.

4.1 Sensor de Cor

O principio de funcionamento deste sensor consiste em varrer toda a área da janela (linhas e colunas), e considerar/mostrar apenas os pixeis da cor pre-definida, devolvendo ainda as coordenadas da maior concentração dessa cor. Isso é feito através da soma da componente da cor desejada numa área correspondente a dois perimetros de pixeis vizinhos.

Um exemplo da utilização deste tipo de sensor é visível na figura seguinte:

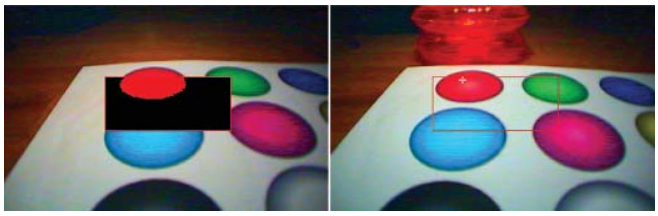

Figura 4 Utilização de sensor de cor para procurar vermelho: esquerda) visível o filtro (mostra apenas os pixeis vermelhos), direita) indica com uma cruz branca a posição do máximo de vermelho

Um exemplo prático da utilização deste tipo de sensores seria no programa de futebol robótico em que têm participado os robôs da Universidade do Minho, para detecção da baliza adversária. Seria assim definido um rectangulo onde se procuraria a cor azul e outro rectangulo onde se procuraria a cor amarela (cores das balizas).

4.2 Sensor de Área

O principio de funcionamento do sensor de área consiste em varrer toda a área da janela, e calcular o somatório das três componentes de cor RGB, e o respectivo valor médio. No final, essas componentes são comparadas com o threshold utilizado nesse sensor e se os valores encontrados forem superiores então a flag é activada (e o rectangulo que delimita o sensor fica a vermelho) indicando que a área foi activada.
Um exemplo da utilização deste sensor encontra-se na figura seguinte. Pretende-se encontrar um rectangulo de cor branca.

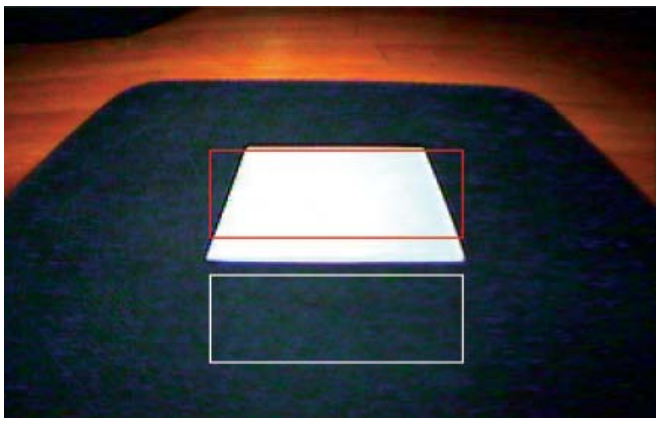

Figura 5 Dois sensores de área (para procurar área branca): o sensor de cima está activado (rectangulo a vermelho), o de baixo desactivado (rectangulo a branco).

4.3 Sensor de Histograma

O sensor de histograma gera três vectores com as respectivas componentes de cor RGB somadas por coluna. Estes valores permitem dar uma perspectiva de cor ao longo da largura da imagem. O respectivo histograma é desenhado por baixo da imagem capturada.
Um exemplo da utilização deste sensor encontra-se na figura seguinte. Pretende-se encontrar a posição de duas linhas brancas (por exemplo as bermas de uma estrada), de forma a que o robô não saia dos seus limites. Os dois picos deste histograma indicam onde se encontra a maior porção de pixeis brancos por coluna.

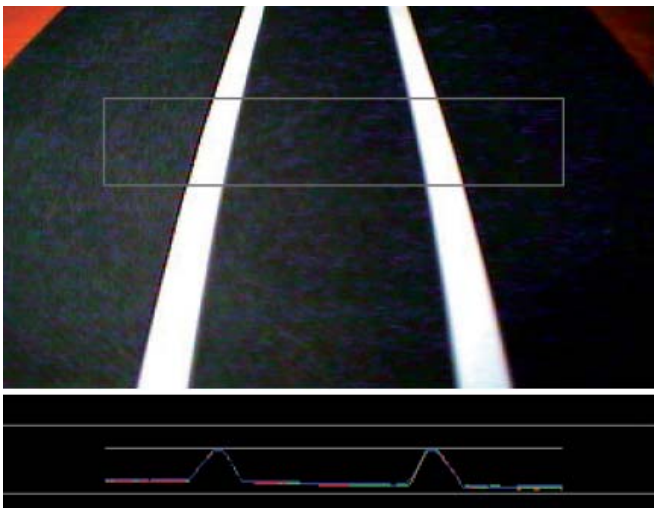

Figura 6 Sensor de Histograma, permite localizar facilmente as linhas brancas na imagem, através dos dois picos ou máximos do histograma.

5. CONFIGURAÇÃO DOS SENSORES

A imagem utilizada para testar os sensores de cor e os sensores de histograma, consiste em 8 esferas (uma com cada cor das 8 reconhecidas por estes sensores), tendo estas esferas as várias tonalidades propositadamente para simular o maior número de casos possiveis.

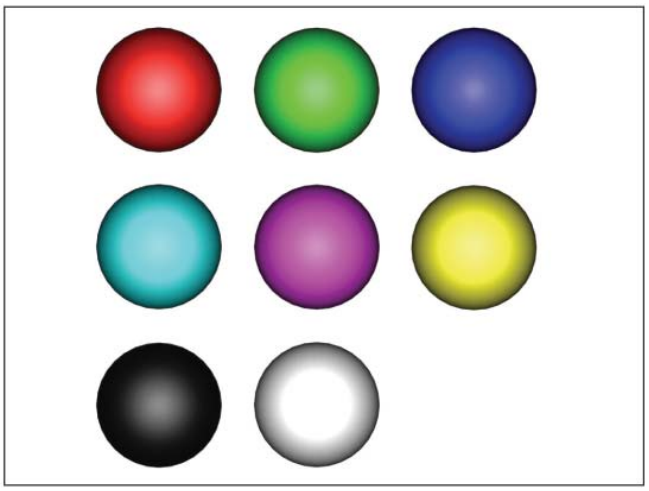

Figura 7 Imagem utilizada para testar os sensores de cor e de histograma

5.1 Sensor de cor

Para criar um sensor de cor é necessário indicar:

a) tipo de sensor (neste caso é um sensor de cor definido pela abreviatura FTLC)
b) número do sensor (número sequencial)
c) cor que se pretende visualizar (uma das RGBCMYKW)
d) coordenadas da janela do sensor (dadas por x1, y1, x2, y2)
e) valor mínimo para considerar que se encontra uma cor ou não (threshold do sensor).

Assim, um exemplo de definição de um sensor deste tipo, seria:

```
FLTC1=G,80,50,245,110 150
```

As primeiras quatro letras indicam o tipo de sensor (FILTRO DE COR), o caracter seguinte indica o número do sensor (número de ordem), a seguir ao sinal = (obrigatório) um caracter que indica a cor a procurar (neste caso verde G), separado por vírgula indica-se as 4 coordenadas que localizam o sensor na imagem, e por fim o indicador de valor mínimo para se considerar se o objecto foi encontrado.
A linha de exemplo define um sensor de cor com o indice 1, que procura a cor verde (Green) dentro da janela com as coordenadas (80, 50) e (245,110) e com um threshold mínimo de 150.

Existem ainda outras variáveis gerais a todos os sensores de cor:

```
LIGADOFLTC=1
VERIMAGEM=0
VERECTANGULOS=1
```

- A variável LIGADOFLTC é uma variável boleana que indica se os sensores estão ligados ou desligados (com o valor 1 ou 0 respectivamente). Isto é utilizado quando se pretende desligar os sensores sem que se queira perder todas os seus parâmetros. Deste modo eles continuam a existir mas não são calculados.
- Com os sensores de cor activos, vê-se na imagem o rectangulo que delimita o respectivo sensor mas com a imagem real por baixo. Deste modo, apesar de todos os calculos estarem a ser efectuados, não é muito perceptivel ao olho humano a visualização dos respectivos pixeis da cor definida. Assim, foi criada a variável VERIMAGEM (do tipo boleana) que indica se se pretende ver a imagem da câmara por baixo do sensor ou se se pretende ver apenas os pixeis filtrados (da respectica cor) sobre fundo preto.
- A variável VERECTANGULOS é uma variável boleana que permite ligar ou desligar (valor 1 ou 0 respectivamente) a vizualização dos rectangulos que delimitam o sensor de cor. Usado apenas por razões estéticas ou de optimização de código.

A estrutura de dados que guarda a informação do sensor de cor extraida da imagem é definida da seguinte forma:

```
struct FILTRO_COR
{
   struct fltc
   {
      char cor;
      int  x1, y1, x2, y2;
      byte threshold;
      int  flag;
      int  max;
   } F[MAX];
   int n_FLTC;
   int flagFLTCligado;
   int flagverectangulos;
   int flagverimagem;
} FLTC;
```

A estrutura **fltc** contém os dados relativos a cada sensor de cor (a sua cor, as coordenadas da janela, o valor do threshold mínimo, uma flag que indica se foi detectado ou não o objecto da cor pretendida e o valor máximo da cor definida), sendo por isso um vector de MAX elementos.

Por defeito as variáveis **flagFLTCligado, flagverimagem, flagverectangulos**, são inicializadas a 1. Isto quer dizer que se estas variáveis não estiveram definidas no ficheiro de configuração (.GAR) o programa automaticamente inicializa-as a 1 por defeito. A variável n_FLTC é inicializada a 0, ficando depois com o número de sensores de cor definidos no ficheiro GAR.

5.2 Sensor de área

Para criar um sensor de área é necessário indicar o tipo de sensor (neste caso é definido pela abreviatura AREA), sendo o resto dos parâmetros idêntico aos do sensor de cor descrito na secção anterior. O último parâmetro indica o valor mínimo (médio) dos pixeis nessa cor para considerar o sensor como activo (como tendo detectado a área). Assim, um exemplo de definição de um sensor deste tipo seria:

```
AREA1=R,125,180,205,195 140
```

Neste sensor, caso o valor médio dos pixeis dentro dessa área seja superior ao valor dado pelo threshold (140), a variável flag (da estrutura de dados) é activada e o rectangulo da janela fica vermelho em vez de branco (quando desactivado).

```
struct AREAS
{
   struct area
   {
      char  cor;
      int   x1, y1, x2, y2;
      dword threshold;
      int   flag;
   } A[MAX];
   int n_AREA;
   int flagAREAligado;
   int flagverectangulos;
} AREA;
```

As variáveis **flagAREAligado** e **flagverectangulos** são por defeito inicializadas a 1 e têm o mesmo significado que as suas homólogas para os sensores de cor. As outras variáveis da estrutura são identicas às da estrutura do sensor de cor.

5.3 Sensor de Histograma

Com um formato idêntico ao dos sensores de cor e de área, para criar um sensor de histograma é necessário indicar o número do sensor (número sequencial) e as coordenadas do sensor na imagem (dadas por x1, y1, x2, y2). Assim, um exemplo de definição de um sensor deste tipo seria:

```
HIST1=5,170,310,190
```

Neste exemplo definiu-se um sensor de histograma dentro de uma janela definida pelas coordenadas (5,170) e (310, 190). Por razões de optimização de código e para permitir uma maior velocidade de cálculo, este sensor devolve um vector com o valor das três componentes de cor. Pode assim utilizar-se as mesmas 8 cores dos outros sensores embora não fiquem guardadas na estrutura de dados do sensor histograma. A estrutura de dados que compõe este sensor encontra-se descrita a seguir.

1 pg
publicidade

```
struct HISTOGRAMA
{
   struct hist
   {
      int x1, y1, x2, y2;
      int pico;
      int flag;
      int max;
   } H[MAX];
   int n_HIST;
   int flagHISTligado;
   int flagverectangulos;
} HIST;
```

As variáveis **flagHISTligado** e **flagverectangulos** são por defeito inicializadas a 1 e têm o mesmo significado que as suas homólogas para os sensores de cor e de área.

Um exemplo da utilização deste tipo de sensor é visível na figura seguinte:

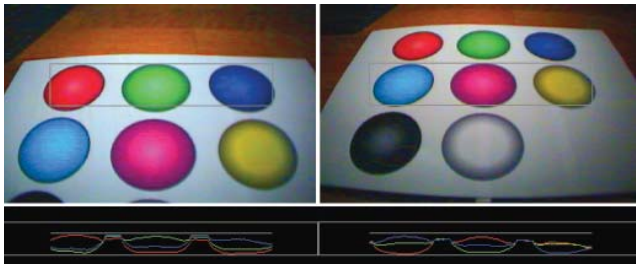

Figura 8 Dois exemplos de sensores de Histograma (imagem capturada em cima e histograma em baixo).

Na figura da esquerda pode ver-se a imagem capturada com a localização da janela do sensor (rectangulo cinzento), e por baixo encontra-se o respectivo histograma. O histograma contém três linhas (uma para cada cor RGB). Assim, é visível no início do histograma que a componente de vermelho é superior às componentes azul e verde, devido à esfera vermelha se encontrar nesse local, depois tem uma zona com as três componentes máximas (cor branca na janela), seguida de um maior valor de verde em relação às outras duas componentes que se refere à esfera verde, etc.

Na figura da direita encontram-se as cores definidas por duas componentes RGB (CMY), e assim temos logo no início as duas componentes azul e verde bastante superiores à componente vermelha (que dão a cor azul marinho ou CYAN), seguida das três componentes altas (branco), etc. É ainda visível na zona da esfera amarela que as duas componentes vermelha e verde são maiores que a componente azul embora sejam mais baixas que a média do histograma devido à pouca luz na zona da esfera amarela.

5.4 Caso Prático

Um exemplo muito simples onde se utilizou uma combinação de sensores foi o descrito na figura seguinte. Pretendia-se capturar as linhas delimitadoras de uma pista, assim como a existência de uma passadeira. Utilizaram-se para isso dois sensores de histogramas para extrair os limitas da pista, e um sensor de área para detectar a passadeira. Assim o robô deslocava-se e virava de acordo com os picos dos sensores de histograma e parava quando o sensor de área detactava a passadeira (quando a sua flag passava a 1). Existia ainda um semáforo do lado direito da pista, para o qual se utilizou um sensor de cor para percepção da cor ligada (embora este não apareça na figura).

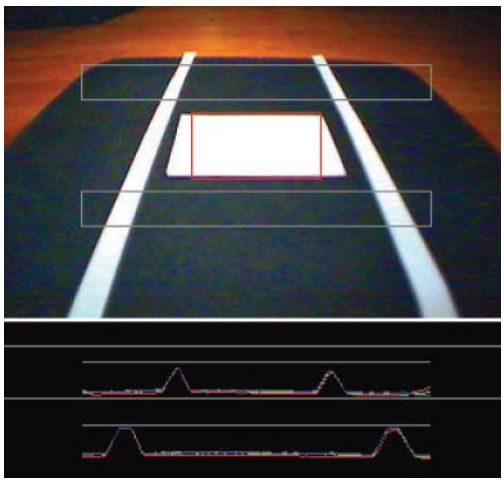

Figura 9 Utilização de dois sensores de histograma e um sensor de área.

6. CONCLUSÕES

São várias as conclusões a tirar da utilização destas ferramentas. Em primeiro lugar importa realçar que todas estas rotinas funcionam e demonstram os seus valores em tempo real. Nas experiências realizadas, utilizou-se uma motherboard com um CPU a correr a 200MHz e conseguiram-se cerca de 40 a 50 frames por segundo. Evidentemente que quanto mais sensores se utilizarem menor será este valor devido ao acréscimo de calculos necessários.

Estas ferramentas são bastante flexiveis, facilmente parametrizaveis, sem necessidade de recompilar o código, extremamente fiáveis e profundamente testadas. Apesar disso, continuarão a ser desenvolvidas e melhoradas de acordo com as necessidades que forem surgindo.

Estes sensores virtuais têm bastante aplicação prática podendo ser utilizados em qualquer tipo de aplicação que utilize visão por computador, mesmo em aplicações industriais.

As cores utilizadas por estas ferramentas são as 8 acima descritas devido à simplicidade na sua captura, mas importa realçar que as margens de erro são definidas pelo utilizador através dos valores de threshold. Mesmo assim, nas competições robóticas já participadas (RoboCup, Eurobot, ROBOTICA'200X, etc.) estas cores são as usadas por se encontrarem nos vértices do cubo RGB.

Todos estes tipos de sensores foram já testados em vários robôs que participaram em competições como o RoboCup (com sensores de cor), no Robótica'2002 (com sensores de Histograma para detectar as linhas da pista, de cor para detectar semáforos, e de áreas para detectar a passadeira onde o robô teria de parar), e no Eurobot'2002 (sensor de cor para detectar as bolas, sensor de área para detectar os cestos onde as bolas seriam inseridas, e sensores de histograma para detectar as linhas no campo).

Para que o processamento de imagem fosse suficientemente rápido de modo a tirar partido máximo da velocidade do microprocessador, o código está extremamente optimizado tendo sido escrito em linguagem C (código genérico) e Assembly (as rotinas de processamento de imagem).

O ficheiro de configuração guarda não apenas informação sobre os sensores virtuais, mas também informação sobre o tipo de placa gráfica e o seu modo e resolução (permitindo maior portatibilidade do software), bem como informação sobre o cursor (cor, posição, incremento, etc). Para além disso é possivel guardar outras variáveis genéricas que se pretenda sejam parametrizaveis e facilmente/rapidamente editaveis (tais como variáveis de controlo do movimento, constantes, velocidades máximas, etc.).

Podem utilizar-se todos os tipos de sensores na mesma aplicação sem problemas de compatibilidade ou limites no seu número. O número máximo de sensores a utilizar numa aplicação é teoricamente o limite fisico da memória do computador.

A ordem das instruções neste ficheiro de configuração não é rígida (pode ser aleatória), mas recomenda-se que seja seguida alguma ordem lógica para ser mais fácil a sua leitura por parte dos utilizadores.
Uma das desvantagens consiste na dificuldade em reconhecer objectos através da sua cor quando em ambientes com pouca iluminação, com bastante ruido ou mesmo em ambientes com muitas e variadas cores de fundo. Para além disso, é sempre necessária uma Calibração a qual não é sempre fácil de obter e depende bastante da quantidade de luz âmbiente (entre outros factores). Esta calibração é necessária para calcular os valores ideias de threshold.

7. TRABALHO FUTURO

Este trabalho está em constante desenvolvimento, e virá no futuro a incluir outros tipos de sensores. Estes novos sensores a acrescentar a estas ferramentas serão criados de acordo com as necessidades, embora os próximos estejam já planeados. Assim, serão criados Sensores de Movimento que permitirão dizer quando um determinado objecto de moveu e Sensores de Distância que permitirão dizer a que distância se encontra determinado objectivo, etc.
Estas ferramentas irão também ser adaptadas a outros sistemas operativos como Linux.
Pretende-se também que estas ferramentas sejam menos sensíveis a alterações na quantidade de luz ambiente, quer através de uma calibração, quer através de algorítmos de correcção dos dados adquiridos.

REFERÊNCIAS

[1] Carlos Machado, Sérgio Sampaio, Bruno Martins e António Ribeiro, "Image Processing Applied to a robotic Football Team", Workshop on EuRoboCup'2000, Amsterdão, Holanda, 28 Maio – 2 Junho, proceedings em CD-Rom, Springer.

[7] Sérgio Monteiro, Fernando Ribeiro, Paulo Garrido, "Problems, Solutions and Trends in Middle-Size Robot Soccer - A review", Robotica'2001 - Festival Nacional de Robótica, 25-28 Abril 2001, Guimarães, Portugal.