Automatização do processo de calibração de cor

Carlos Figueiredo, Jorge Neves

Licenciatura em Engenharia Informática e Computação

Faculdade de Engenharia da Universidade do Porto

E-mail: {ei99030, ei99040}@fe.up.pt

Resumo

Este documento visa descrever o processo relacionado com a visão, desenvolvida pela equipa da CMU [1]. Será abordado o processo de visão dando-se maior relevo à técnica relacionada com a calibração da imagem (threshold). Será também apresentada a descrição do trabalho a realizar no âmbito da cadeira de Robótica do 4º ano da Licenciatura em Engenharia Informática e Computação.

1 Introdução

A equipa *CMPack02* [1,6] é uma equipa desenvolvida para participar na Liga de Robôs com Pernas pelo *Multirobot Lab* da *School of Computer Science*. Esta equipa foi desenvolvida em C++ para a arquitectura de robôs *Aibo*[2] *ERS210A* [2]. Esta arquitectura inclui entre outras funcionalidades, câmara fotográfica e vários movimentos de cabeça e pernas.

O processo de análise de imagem permite ao robô detectar objectos durante o jogo. Alguns objectos permitem ao robô orientar-se no campo e tomar decisões quanto às acções a efectuar. Entre os objectos detectados encontram-se a bola de jogo, os colegas de equipa, os adversários e as balizas. A análise de imagem é feita com base na cor das imagens obtidas. Para além da detecção de objectos, o módulo de visão identifica a distancia a que estes se encontram do robô.

2 A visão no CMPack02

O objectivo do módulo de visão é a conversão das imagens em YUV [3] (este é o formato em que são disponibilizadas as imagens da câmara) em informação sobre a localização dos diferentes objectos associados a um jogo desta liga.

[1] Carnegie Mellon University.
[2] Aibo é uma marca registada pela empresa Sony

A visão é feita em dois processos principais: o processo de baixo nível e o processo de alto nível. O primeiro processo é implementado pelo módulo *CMVision* [4,5] e é responsável por identificar regiões da mesma cor numa determinada imagem. O segundo processo pretende detectar objectos com base nas regiões de cor anteriormente detectadas e obter a distancia destes ao robô.

2.1 O processo de baixo nível (CMVision)

Este processo desenrola-se em várias partes, inicialmente a imagem é segmentada de acordo com uma tabela de cores existente, a forma como a calibração das cores da tabela é feita será discutida mais à frente. O objectivo é detectar qual a cor de cada *pixel* de acordo com as cores conhecidas na tabela. Seguidamente é efectuado um processo de redução do tamanho das imagens, este processo é designado *Run Lenght Encoding* (RLE). A fase seguinte pretende agrupar regiões da mesma cor que sejam vizinhas.

O processo de segmentação usa uma tabela para segmentar a cor da imagem. Essa tabela tem definido os valores máximos das variáveis Y, U e V para cada *pixel* da imagem. O resultado da segmentação é a construção de um mapa de cor para a imagem, que define na imagem quais os *pixeis* de uma determinada cor. Este mapa é disponibilizado ao processo de alto nível para que este verifique a cor de cada *pixel*.

Seguidamente é efectuado o RLE, sobre o mapa obtido anteriormente. Este processo consiste em substituir conjuntos de *pixeis* por um *run*. Os *pixeis* de cada *run* têm de pertencer à mesma linha da imagem. Um *run* é uma estrutura de dados que guarda a cor a posição *x*, *y* do primeiro *pixel* com essa cor e o tamanho ou seja o número de *pixeis* da mesma cor. O objectivo desta redução de tamanho da imagem é a redução do tempo de processador de pesquisa sobre ela.

O passo seguinte consiste em agrupar *run´s* que sejam vizinhos e que tenham a mesma cor. Este passo associa cada *run* a um pai. O pai consiste no *run* da mesma cor que esteja mais à esquerda e mais no topo da imagem e que esteja ligado directa ou indirectamente, através de outros *run´s* da mesma cor, ao pai. A estrutura que define um *run* tem assim mais um valor que representa o pai do *run*. Inicialmente cada *run* é o seu próprio pai. No final obtêm-se listas com os *run´s* que constituem cada região da mesma cor.

2.2 O processo de alto nível

Este processo tem o objectivo de encontrar objectos na imagem obtida da câmara e em estimar a distância deste à posição do robô. Para tomar decisões este processo tem de ter acesso à imagem original, ao mapa de cor, à imagem de RLE e às listas de regiões. Esta função obtém ainda dados dos sensores do robô para poder calcular a distância aos objectos. A localização de cada objecto em relação ao robô é uma estrutura de dados que define a sua posição em 3d, em relação a um ponto conhecido pelo robô. Para cada objecto existem funções próprias para efectuar a sua localização.

3 O processo de calibração de cor

Como foi dito anteriormente é necessário efectuar uma calibração de cor para construir a tabela que permite detectar as cores das imagens obtidas da câmara do robô. Este processo de calibração consiste em obter um conjunto de imagens do ambiente de jogo (campo e respectivos acessórios) e em processar essas imagens de forma a detectar as variações de cores possíveis no ambiente de jogo.

A calibração de cor é muito importante pois permite definir intervalos de valores que definem cada uma das cores do campo de jogo. Nas imagens obtidas da câmara do robô um mesmo objecto pode ter várias cores. Esta variação de cor é principalmente devida à iluminação ambiente, a possíveis sombras sobre o objecto, ao ângulo da câmera com o objecto e a distância entre estes. A figura 1-c apresenta um exemplo desta variação de cor, repare-se que as cores reais do poste são rosa e amarelo, muito diferente do que se visualiza na figura.

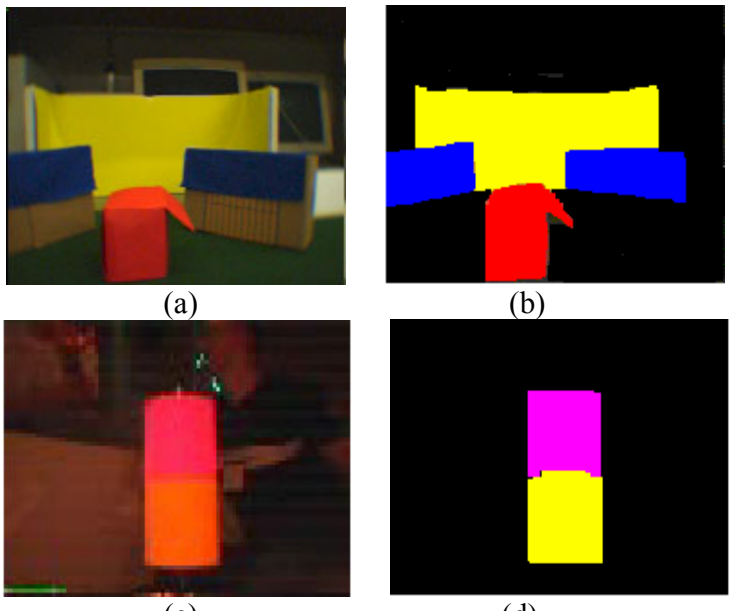


Figura 1*- (a, c) – Fotografias em RGB (d, b) - Imagens classificadas manualmente.*

O processo de calibração de cor da equipa *CMPack02* é efectuado antes do início dos jogos permitindo assim calibrar as cores do terreno de jogo para cada um dos jogos efectuados. A equipa da CMU utiliza o processo seguidamente descrito para efectuar a calibração de cores. Inicialmente são tiradas fotografias, com a câmara do robô, do terreno de jogo. Estas fotografias devem ser tiradas de diferentes ângulos e distancias permitindo assim identificar o maior nível possível de variações de uma mesma cor. As fotografias devem ainda contemplar todos os objectos relacionados com o jogo: chão (verde), delimitação do campo (branco), duas balizas (amarelo e *cyan*), adversários e colegas de equipa (azul e vermelho) marcador do lado da baliza amarela (combinação amarelo rosa de um lado e combinação rosa amarelo do lado oposto), marcador do lado da baliza *cyan* (combinação *cyan* rosa de um lado e combinação rosa *cyan* do lado opôs),

marcador do centro do campo (combinação verde rosa de um lado e combinação rosa verde do lado oposto). A figura 2 apresenta um campo de jogo, permitindo visualizar os acessórios ao campo de jogo e a sua disposição.

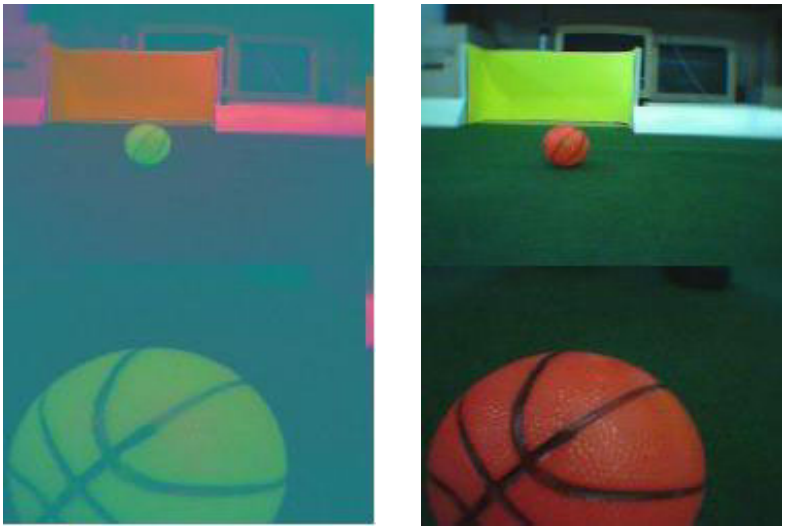
Figura 2 *Campo de Jogo.*

Seguidamente dá-se início ao processo de tratamento das fotografias obtidas. Estas fotografias são obtidas do robô em ficheiros (do tipo: i<X>.ppm; X numero inteiro), estando estas imagens em formato YUV, ver figura 3-a. Seguidamente é necessário detectar as áreas correspondentes a cada cor nas diferentes imagens. Este processo é feito manualmente, ver figura 1 (b, d), o que torna necessário transformar estas imagens em código RGB. O programa *tile*, incluído em *util/thresh* do *CMPack02*, permite transformar as imagens em RGB, agrupando-as ainda numa imagem apenas de forma a facilitar o seu tratamento, ver figura 3 (b). Este programa gera ainda um ficheiro com as imagens agrupadas mas em código YUV, figura 3 (a).

Na pintura manual feita sobre as imagens devem ter-se em conta as seguintes normas.

- São usadas cores base para todos os objectos, estas estão definidas no ficheiro *colors.txt* localizado no *CMPack* em *agent/config/config*.
- O verde dos marcadores e do campo é o mesmo.
- Se um *pixel* não corresponder a nenhuma objecto deve ser pintado de preto.
- *Pixeis* que não sejam relevantes, devem ser pintados de cinzento, em RGB (64,64,64) - (192,192,192).

O processo de pintura manual tem o objectivo de definir quais as regiões de cada cor nas imagens. Permitindo assim ao processo de construção da tabela detectar quais os *pixeis* da imagem original que devem ser tomados em conta para definir o intervalo de valores de cada cor. O resultado deve ser guardado num ficheiro com o nome *label.ppm*.



(a) (b)

Figura 3 *– (a) - Resultado do tile em YUV. (b) – Resultado do tile em RGB.*

Seguidamente deve modificar-se o ficheiro *util/thresh/colors.txt* para que este passe a referenciar o ficheiro criado (*label.ppm*) e o respectivo ficheiro original codificado em YUV. Este ficheiro permite também definir ficheiros de teste para que após a execução do programa se verifique se as cores geradas com a tabela criada correspondem ao esperado.

O passo seguinte consiste em correr o programa *thresh* localizado em *util/thresh*. Este programa gera um conjunto de ficheiros entre os quais o ficheiro *out.tmap*. Este ficheiro representa uma tabela em 3d de dimensão 16*64*64 que representam os valores de YUV para cada uma das 9 cores possíveis.

O programa *thresh* começa por ler o ficheiro *colors.txt* e por obter deste as cores base que foram utilizadas na pintura manual, construindo um vector com estes dados. Seguidamente procede-se à leitura dos nomes dos ficheiros em YUV e respectivos ficheiros *label* associados. Após esta leitura são construídas estruturas de dados com a informação destes ficheiros, dimensões e informação da cor de todos os *pixeis* associando a cada pixel a cor base que este deve satisfazer. É construída uma lista com esta informação que inclui todos os ficheiros de entrada especificados no ficheiro *colors.txt*.

O passo seguinte constrói uma estrutura de dados que especifica as variações de cor encontradas nas diferentes imagens. Este processo tem que agrupar para cada cor base os *pixeis* associados e depois identifica os diferentes níveis de variação dos valores de Y, U e V.

Para efeitos de teste é possível incluir no ficheiro *colors.txt* um conjunto de imagens em formato YUV e respectivos nomes de ficheiros de saída associados. O programa *thresh* possibilita executar testes classificando as imagens de entrada de acordo com a tabela construída anteriormente. Como saída é gerado um ficheiro com as regiões identificadas de cada cor na imagem original.

4 Trabalho a realizar

Nesta secção será descrita o trabalho a ser realizado pela nossa equipa. Para a sua concepção, achou-se conveniente dividi-lo em duas fases, uma vez que consistem em 2 módulos distintos. O objectivo deste trabalho é automatizar o processo de calibração de cor anteriormente descrito. Desta forma pretende-se que o robô tire as fotografias sem intervenção humana e que seguidamente tenha a capacidade de processar as imagens construindo a tabela sem se ter de fazer a pintura manual de imagens, automatizando este processo.

- **Primeira Fase**

  Na primeira fase o robô *AIBO* deve ser capaz de efectuar com sucesso as seguintes três etapas: - orientação, planeamento, fotografar.

  Pretende-se com a orientação, que o robô após ser colocado no terreno de jogo seja capaz de se orientar e saber a sua posição. Esta orientação consiste na tentativa de localização da baliza, que é o objecto de maiores dimensões e com maior probabilidade de ser detectado pelo *AIBO*. Após a sua localização deve ser capaz de efectuar um planeamento que o permitirá tirar fotografias de forma autónoma aos restantes objectos que compõem o terreno de jogo, isto é a baliza e os seis postes de delimitação do campo. Para tal, assumimos que o robô é colocado numa posição inicial frontal à baliza amarela (ver figura 4). Desta posição serão tiradas fotografias à baliza, ao poste P5, P6, P1, e P2. O robô deslocar-se-á então até à posição 2 onde com maior pormenor tirará nova foto ao poste P1 e à baliza. De seguida fará um longo caminho até a posição 3 onde fará mais duas fotografias, uma ao poste P1 e outra à baliza. Na posição 4, que será o mais central à baliza possível, será tirada mais uma foto a baliza do lado oposto. Na posição 5 será feito o mesmo que na posição 3 e na posição 8.

  Nesta fase sempre que uma foto for tirada o robô *AIBO* deve ser capaz de identificar que objecto esta a fotografar, indicando o seu nome com recurso ao *speaker*. Será ainda guardada uma estrutura de dados referente às fotografias tiradas, que permitirão saber que objecto e qual a sua distancia a este, com recurso ao infravermelho que equipa o robô. As fotografias serão então armazenadas em memória, para serem usadas na segunda fase.

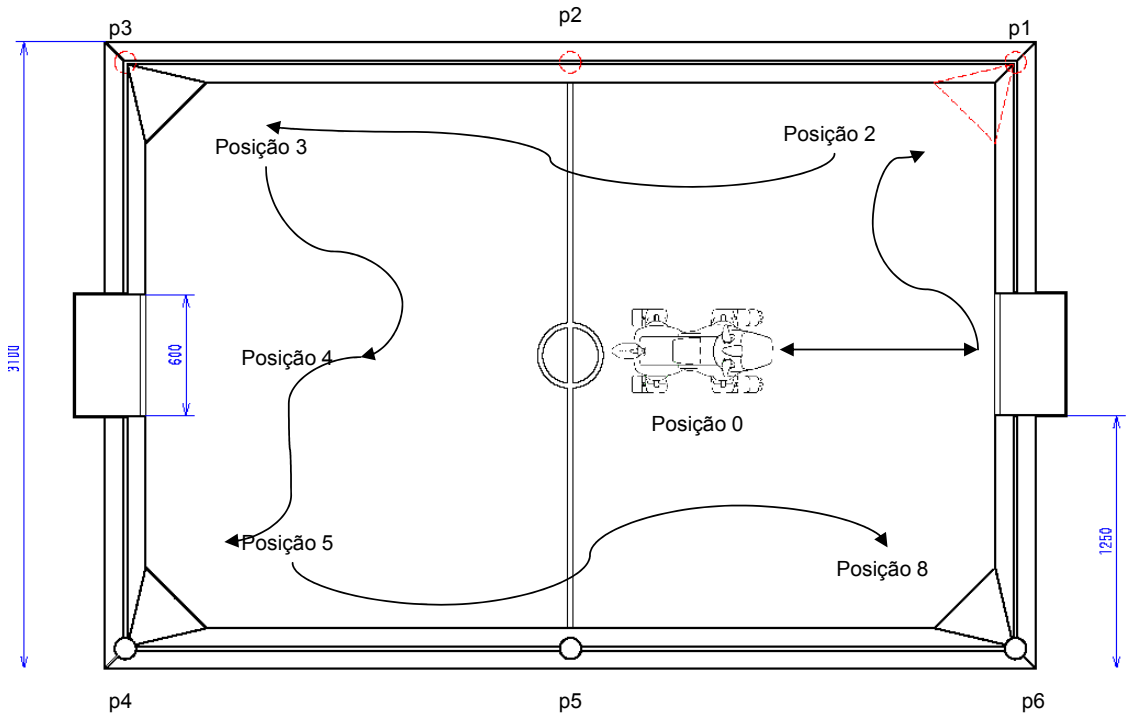


Figura 4 *– Trajecto do robô.*

- **Segunda Fase**

  Nesta fase será efectuado o processamento das fotografias tiradas na fase transacta. Pretende-se com este processamento efectuar uma calibração automática das cores em campo. Para tal iremos recorrer às fotos tiradas anteriormente.

  Devido a estas serem guardadas no formato YUV, associado ao facto de estarmos mais habituados a trabalhar com RGB, torna-se necessário usar o programa *tile,* que não só converte as imagens para RGB como também as agrupa todas numa só, o que facilita posteriormente o tratamento. Com a imagem retornada pelo *tile* irá ser feito o processamento automático de detecção de cores. Como cada imagem tem uma estrutura de dados associada, nomeadamente a distancia a que foi tirada a fotografia ao objecto e a identificação desse objecto, sabemos não só que cores procurar como também a quantidade de *pixeis* dessa cor a encontrar na imagem, de modo a obtermos com alguma precisão a área correcta a "pintar". O algoritmo irá então analisar *pixel* a *pixel* as imagens e deve ser capaz de encontrar as cores pretendidas, que se encontram numa tabela. Após este processamento ter sido concluído será necessário construir a tabela de cores. Para tal recorremos ao programa *thresh*.

5 Bibliografia

[1] http://www-2.cs.cmu.edu/~coral/main/index.html

[2] http://www.sony.net/Products/aibo/aiboflash.html

[3] http://www.joemaller.com/fcp/fxscript_yuv_color.shtml

[4] J. Bruce, T. Balch e M. Veloso. Fast and inexpensive color image segmentation for interactive robots.

[5] http://www-2.cs.cmu.edu/~jbruce/cmvision/

[6] M. Veloso, S. Lenser, D. Vail. CMU´s Legged Robot Soccer Team.