AIBO Sony – Introduzione alla gestione delle immagini

Laura D'Angelo e Igor Colombo
Laboratory of Applied Intelligent Systems (AIS-Lab)
http://ais-lab.dsi.unimi.it
Versione 1.3 – 14th December 2004

Sommario

Il documento si propone di essere una rielaborazione della documentazione reperita su internet, sia proveniente da Sony che non.

	Pagina
Indice	2
1. Acquisire immagini – Il formato OfbkImageVectorData	3
1.1 Metodologia di compressione	4
1.2 Riconoscimento del colore	5
1.3 Parametrizzare la CTD	5

1. Aquisire immagini - Il formato OFbkImageVectorData

Aibo fornisce informazioni dalla camera attraverso quattro differenti layer. I primi tre layer rappresentano **immagini a colori** con una differente risoluzione (alta, media e bassa), il quarto rappresenta **un'immagine di riconoscimento dei colori** (Aibo possiede un algoritmo di riconoscimento dei colori embedded ma parametrizzabile).

Le immagini a colori sono nel formato YCrCb, il che significa che sono codificate usando le componenti: Y che rappresenta la luminosità, Cr la componente rossa meno Y, e Cb la componente blu meno Y. OPEN-R fornisce separatamente l'accesso al valore di un pixel nella banda Y, un pixel nella banda Cr, e un pixel nella banda Cb, mai come un unico dato nel formato YCrCb. In altre parole le immagini a colori sono trattate come tre immagini monocromatiche. In questo modo per scrivere sul disco le immagini a colori, dobbiamo recuperare le varie componenti Y, Cr, Cb separatamente e poi fonderle insieme.

Come abbiamo citato sopra, tramite uno dei servizi messi a disposizione dal sistema per mezzo dell oggetto **OVirtualRobotComm** è possibile aquisire informazioni dalla camera nel **formato OFbkImageVectorData**. In linea pratica è suffciente collegare il servizio **subject OVirtualRobotComm.FbkImageSensor.OFbkImageVectorData.S** ad un **obsever dichiarato in un proprio oggetto** che **accetti tipi di dato OfbkImageVectorData** tramite il file conntect.cfg.

L'oggetto OFbkImageVectorData è una struttura così composta da:

- ODataVectorInfo vectorInfo: l'header comune agli oggetti di sistema predisposti per la comunicazione con i dispositivi.
- OFbkImageInfo info[<n>]: I metodi a disposizione sono getData(int index) che restituisce il byte puntatore all area di memoria dove risiede limmagine e getInfo(int index) che restituisce l'oggetto OFbkImageInfo, l'indice a parametro può essere indicato tramite le costanti:
- ofbkimageLAYER H (alta risoluzione)
- ofbkimageLAYER M (media risoluzione)
- ofbkimageLAYER L (bassa risoluzione)
- ofbkimageLAYER C (riconoscimento colore)

Ad esempio per accedere al layer immagine a media risoluzione si usa: GetInfo(ofbkimageLAYER M) e per accedere al dato GetData(ofbkimageLAYER M).

L' OPEN-R SDK fornisce una **classe per la manipolazione delle immagini**, questa classe è chiamata **OFbkImage**.

Per creare un'istanza della classe OFbkImage è necessario avere un puntatore alle informazioni e un puntatore al dato vero e proprio. Questi puntatori vengono recuperati attraverso l'uso delle funzioni GetInfo() e GetData() di un OFbkImageVectorData come detto in precedenza.

Il costruttore di OFbkImage necessita anche di un argomento che specifica in quale canale dovrà manipolare l'immagine. Per la selezione della banda si usano le costanti:

```
ofbkimageBAND_Y
ofbkimageBAND_Cr
ofbkimageBAND_Cb
ofbkimageBAND_CDT.
```

Per esempio se fbkIVD è di tipo OFbkImageVectorData, allora avremo:

```
OFbkImage(fbkIVD->GetInfo(ofbkimageLAYER_M),  fbkIVD->GetData(ofbkimageLAYER_M),
ofbkimageBAND_Y).
```

Verrà così creato un OFbkImage che si riferisce alla componente Y di media risoluzione.

Per ottenere un immagine a colori è necessario recuperare tutti e tre i componenti Y,Cr,Cb e convertirli secondo il formato scelto, ad esempio RGB per il formato file BMP. L'SDK Open-R non fornisce alcun metodo per manipolare le immagini e acuisirlein in formati conosciuti (BMP, GIF, JPEG:...).

1.1 Metodologia di compressione

La camera in dotazione Aibo permette di riprendere immagini fino a 350.000 pixel. La risoluzione raggiungibile con il LAYER_H e' di 208x160. In effetti AIBO applica un metodo di compressione sulle immagini, si tratta della trasformata di Haar. Questa trasformata perette una scomposizione multi-livello/multi-sottobanda, ovvero ad ogni livello di compressione che e' possibile applicare iterativamente, le informazioni vengono rimpiazzate da 4 sottobande, nel caso AIBO il livello di compressione è 1. Il sistema AIBO mette quindi a disposizione 3 ulteriori bande(sottobande),oltre alle spora citate:

```
ofbkimageBAND_YLH
ofbkimageBAND_YHL
ofbkimageBAND_YHH
```

che rappresentano i coefficienti del livello di compressione sottostante ovvero dell'immagine originale (avendo un unico livello di compressione).

La trasformata di Haar, a partire da un input in pixel di 2x2 genera 4 coefficienti, LL, HL, LH, HH rispettivamente:low-pass per righe e per colonne LL, high-pass per righe e low-pass per colonne HL, low-pass per righe e high-pass per colonne LH, high-pass per righe e high-pass per colonne HH. Algebricamente la generazione dei coefficienti avviene in questo modo dato un input di a,b,c,d pixels:

ll = [(a+b)+(c+d)]/4
lh=[(a+b)-(c+d)]/4
hl=[(a-b)+(c-d)]/4
hh=[(a-b)-(c-d)]/4

In LL viene posto l'informazioe piu' significatica ed infatti corrisponde alla banda ofbkimageBAND_Y quella utilizzata per il valore di luminosita' nel formato 208x160. Applicando l'antitrasformata dati i 4 coefficienti definiti nelle 4 bande Y (LL),LH, HL e HH si possono ricalcolare i pixel originali:

a = ll + lh + hl + hh
b = ll + lh - hl - hh
c = ll - lh + hl - hh
d = ll - lh - hl + hh

Applicando l'inversa della trasformata di Haar a partire dai coefficienti ciati e' possibile creare immagini in bianco e nero a 416x360 essendo i coefficienti calcolati sulla banda Y(luminosità). Per approssimazione le immagini a colori possono essere ingrandite della stessa dimansione associando ad ogni componente Y decompresso il valore di Cr e Cb relativo al pixel dal quale si sta' effettuando la decompressione.

1.2 Riconoscimento del colore

L'algoritmo codificato a livello hardware per li riconoscimento dei colori che AIBO mette a disposizione si basa sulla seguente strutturazione. E' possibile riconoscere 8 colori o meglio 8 configurazioni di colorazione, le caratterisriche di ciascuna configurazione in termini di Y, Cr e Cb vengono registrate in quelli che il sistema definisce canali, 8 per l'appunto.

Tramite il layer CDT (Color Detection Table), è possibile valutare per ciascun byte di informazione il rilevamento o meno dei colori codificati nella CDT, questo è eseguito con una sola operazione in and tra il pixel di informazione e una maschera di byte legata a ciascun canale.

Dunque il sistema mette a disposizione una tabella di 8 canali contenete per ciascun canale una maschera di 1 byte. I canali sono parametrizzabili secondo la configurzione di colorazione che si desidera far riconoscere all AIBO.

Un esempio di codice per verificare il matching di un pixel con uno dei canali definiti è il seguente:

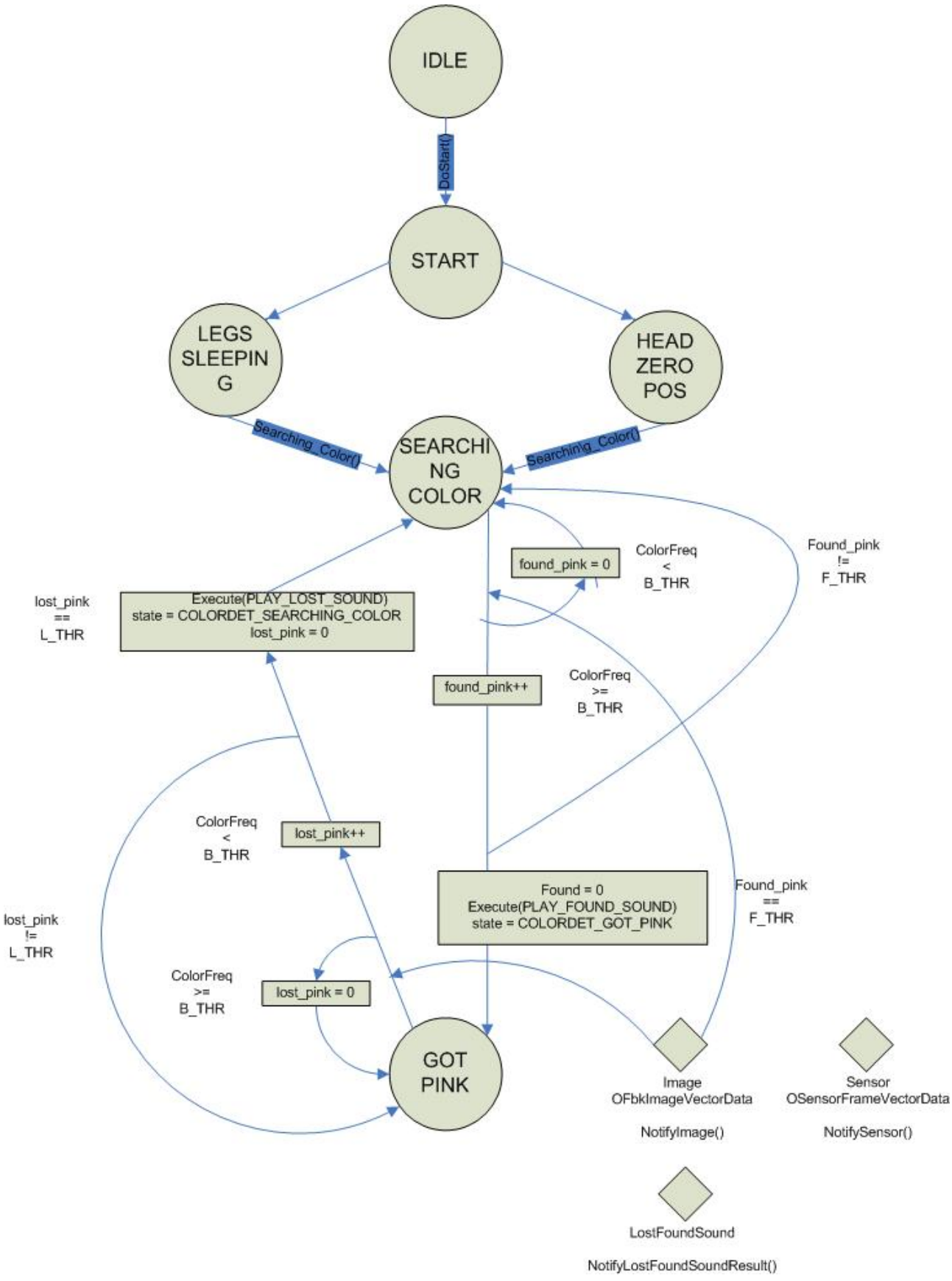
```
OFbkImage image ( fbkIVD-> Get Info (ofbkimageLAYER_C) , fbkIVD-> GetData
  (ofbkimageLAYER_C) , ofbkimageBAND_CDT ) ;
if ( ( image - > Pixel (10 ,10) & 0 x10 ) == 0x01 ) //il riconoscimento è
positivo per il pixel in posizione (10, 10) sulla maschera 0x10
```

1.3 Parametrizzare la CTD

La descrizione di una colorazione su un canale e' configurata come un insieme di 32 sezioni in cui e' suddivisa la luminosità Y per ciascuna di queste sezioni sono indicati i valori massimi e minimi che Cr e Cb possono assumere. Dunque per ciascuna delle 32 sezioni in cui Y(da 0 a 255) e' suddiviso, è configurabile un volume che descrive il range dei valori CrCb dentro il quale il colore viene caratterizzato per quella sezione.

Durante il processo di riconoscimento del colore, l'hardware confronta la componente Y per ciascun pixel e la mette in rapporto alla relativa sezione Y verificando che il valore Cr e Cb del pixel ricada in un punto sul volume CrCb parametrizzato.

OPEN-R mette a disposizione l'oggetto **OCdtVectorData per inviare (parametrizzare)** le informazioni relative agli 8 canali di riconoscimento. L’oggetto è composto dal consueto ODataVectorInfo e un array di 8 OCdtInfo, quest’ultimo mette a disposizione il metodo set(Y_segment , Cr_max, Cr_min, Cb_max, Cb_min) che può essere eseguito per ciascuna delle 32 disponibili sezioni di Y.


IDLE
DoStart()
START
LEGS SLEEPIN G
HEAD ZERO POS
Searching_Color()
Searching_Color()
SEARCHI NG COLOR
found_pink = 0
ColorFreq < B_THR
Found_pink != F_THR
lost_pink == L_THR
Execute(PLAY_LOST_SOUND)
state = COLORDET_SEARCHING_COLOR
lost_pink = 0
found_pink++
ColorFreq >= B_THR
ColorFreq < B_THR
lost_pink++
Found = 0
Execute(PLAY_FOUND_SOUND)
state = COLORDET_GOT_PINK
Found_pink == F_THR
lost_pink != L_THR
ColorFreq >= B_THR
lost_pink = 0
GOT PINK
Image
OFbkImageVectorData
NotifyImage()
Sensor
OSensorFrameVectorData
NotifySensor()
LostFoundSound
NotifyLostFoundSoundResult()

1.4.2 Oggetto MovingHead

L' oggetto MovingHead ha come scopo quello di portare la testa dell' Aibo alla posizione zero (posizione iniziale), è molto utile perché mostra i passi base per inizializzare e muovere le giunture. Si parte sempre dallo stato IDLE, che però differisce dai precedenti in quanto l'oggetto è sempre in questo stato e solo quando arriva un messaggio dalla porta Command l'oggetto esegue una transizione di stato. Per prima cosa se la porta Move(OVirtualRobot Comm observer) è pronta viene chiamata la funzione AdjustDiffJointValue, viene spedito un messaggio e l'oggetto esegue una transizione nello stato ADJ DIFF JOINT VALUE, viceversa se la porta Move non è pronta resta nello stato START e aspetta.
Lo scopo della funzione AdjustDiffJointValue è quello di calibrare i sensori delle giunture e del motore, la funzione legge prima la posizione di ciascuna giuntura, che può differire da quella reale che non è calibrata, a questo punto viene spedito un comando che risetta le giunture nelle posizioni lette inizialmente. In questo modo è stata eseguita la calibrazione.
Per finire viene settato il PID di ogni giuntura con la funzione SetJoinGin() quando l'oggetto passa nello stato MOVING ZERO POS, facendo cosi muovere la testa nella posizione zero.
Questo viene fatto in ZP_MAX_COUNTER passi dopo di che torna nello stato IDLE.
Si sottolinea che prima di fare muovere qualunque giuntura bisogna eseguire una sequenza di inizializzazione.

1.4.3 Oggetto MovingLegs

E'molto simile all'oggetto precedente, eccetto il fatto che dopo aver portato le zampe nella posizione base, le si devono portare in una posizione standard chiamata "sleeping position".
Esiste un array che contiene tutti i valori da assegnare alle giunture per le posizioni "stand" "sleep", "rest".