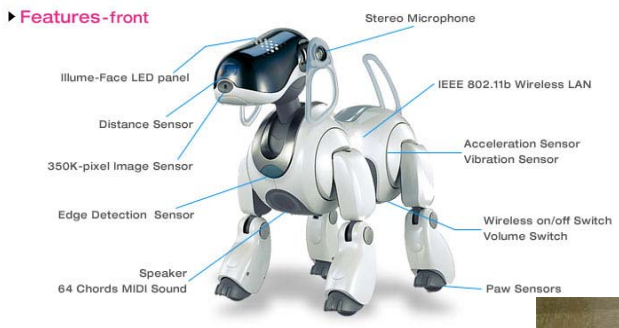
Visione - Esercitazione

autori Gilberto Decaro e Alberto Borghese

Aibo – Sony – ultima versione:



ERS7

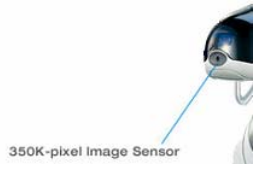
- Visione
- Movimento
- Coordinazione, flotta di robot (Wifi 802.11b).
- MIPS R7000, 576Mhz, 64Mbyte RAM

http://openr.aibo.com

Sensori di distanza

- Tecnologia a raggi infrarossi
- Sul muso: da 0,1 a 0,9 metri
- Sul corpo:
  - Near: da 0,05 a 0,5 metri
  - Far: da 0.2 a 1.5 metri
- Misure attendibili

Telecamera



- Tecnologia CMOS
- Dimensione massima immagine 416 x 320 (decompressa)
- Le immagini sono fornite compresse con il metodo “trasformata di Haar”
- 30 fps
- Offre quattro layers di immagini: tre layer di immagini compresse a bassa, media, alta risoluzione ed un layer di color detection.
- Immagini in formato YCrCb
- Color Detection hardwired, riconosce fino a 8 colori contemporaneamente

Telecamera (bassa risoluzione)

52 x 40

Telecamera (media risoluzione)

104 x 80

Telecamera (alta risoluzione)

208 x 160

Telecamera (alta risoluzione

de

416 x 320

Telecamera (color detection)

Immagine originale

Immagine segmentata

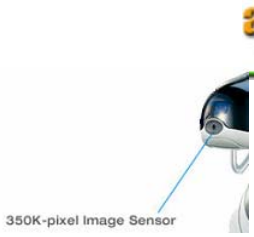
Clustering dei colori in un numero ridotto di colori campione

Telecamera (bianco e nero)

416 x 320

Alta risoluzione, immagine decompressa

Telecamera



Tecnologia CMOS

Dimensione massima immagine 416 x 320 (decompressa)

Le immagini sono fornite compresse con il metodo “trasformata di Haar”

30 fps

Offre quattro layers di immagini: tre layer di immagini compresse a bassa, media, alta risoluzione ed un layer di color detection.

Color Detection hardwired, riconosce fino a 8 colori contemporaneamente

Formato immagini

- Le immagini sono codificate in **YCrCb**
  - Y = luminosità
  - Cr = rosso meno la luminosità
  - Cb = blu meno la luminosità
  - Conversione da RGB a YCrCb:
    - **Y**= 0.299**R** + 0.587**G** + 0.114**B**
    - **Cb**= -0.169**R** - 0.331**G** + 0.5**B** + 128
    - **Cr**= 0.5**R** - 0.419**G** - 0.081**B** + 128
- Quattro **layer** compressi = Alta, media, bassa e Color Detection.
- Per ogni layer si puo’ accedere solo alle singole componenti Y, Cr e Cb che vanno poi ricomposte per generare l’immagine a colori; per il layer Color Detection si accede solo alla componente color detection.

I Layer

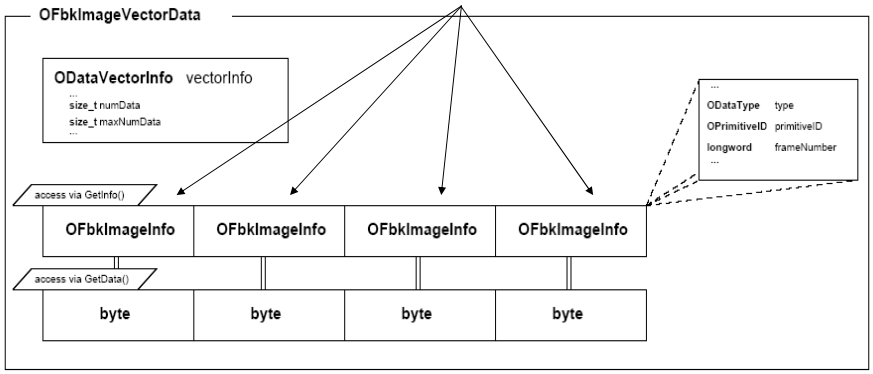
- I quatto layer hanno dimensione:
  - **Layer h**: 208x160 (recostructed 416x320)
  - **Layer m**: 104x80 (recostructed 208x160)
  - **Layer l**: 52x40 (recostructed 104x80)
  - **Layer cdt**: 104x80

OVirtualRobotComm

- Servizio utilizzato per accedere alla telecamera:
  - OvirtualRobotComm.FbkImageSensor.OFbkImageVectorData.S

Struttura dati telecamera

4 elementi: uno per layer



OFbkImage

- OFbkImage: oggetto per la gestione delle immagine della telecamera:
  - Costruttore di OFbkImage: OFbkImage* img(

    imageVector->GetInfo(ofbkimageLAYER_M),

    imageVector->GetData(ofbkimageLAYER_M),

    ofbkimageBAND_Y);

    - Oggetto per la gestione della banda Y del layer m.
  - Alcuni metodi utili di OFbkImage: IsValid(), Pointer(), Width(), Height(), Pixel(x,y)...

Trasformata di Haar 1/3

- Le immagini che ritorna OVRC sono compresse con la trasformata di Haar (compressione loss-less).
- E' possibile utilizzarle senza decomprimerle.
- Sono disponibili altri tre layer: ofbkimageBAND_Y_LH , ofbkimageBAND_Y_HL , ofbkimageBAND_Y_HL.
- Dati quindi i coefficienti LL (ofbkimageBAND_Y), LH, HL e LL è possibile ricostruire l'immagine quadruplicando la dimensione originale.

Trasformata di Haar 2/3

- ALGORITMO:
  - Per ogni pixel p dell'immagine:
    - Recupero i coefficienti LL, LH, HL, HH di p
    - Calcolo le quattro componenti y di p:
      - a = ll + lh + hl + hh
      - b = ll + lh - hl - hh
      - c = ll - lh + hl - hh
      - d = ll - lh - hl + hh
    - Recupero i valori cr e cb di p

Trasformata di Haar 3/3

- A questo punto ho le componenti YCrCb di quattro pixel:

	0	1
0	D	C
1	B	A

- Ho utilizzato 6 byte (LL, LH, HL, HH, CR, CB) al posto di 12 byte (Y Cr Cb * 4).
- Guardate il codice di W3AIBO (metodo *ReconstructAndConvertYCbCr()*) per il codice Open-R

Gestione del Color Detection

1. Color **D**etection **T**able hardware:

- Eseguito in hardware: non vengono utilizzate risorse di calcolo.
- Massimo 8 colori riconoscibili.
- Risoluzione 104x80.

2.Color Detection Software utilizzando la classe

AISAiboYCrCbColorSeg

- Comsuma risorse di calcolo: su Aibo ERS7 circa 2msec per immagine (Layer H).
- Massimo 32 colori riconoscibili
- Risoluzione arbitraria: dipende dalla dimensione dell'immagine originale

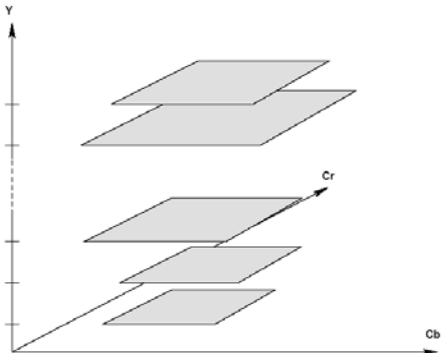
CDT Hardware

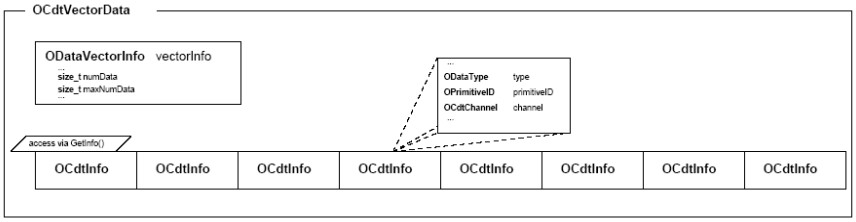
- Si impostano i valori dei canali (colori), sono possibili 8 canali.
- Si recupera il Layer Color Detection
- Per ogni pixel (1byte) del layer si applica una maschera:

Channel number	bytemask	bytemask in hexadecimal
1	00000001	0x01
2	00000010	0x02
3	00000100	0x04
4	00001000	0x08
5	00010000	0x10
6	00100000	0x20
7	01000000	0x80
8	10000000	0x100

CDT: Impostare i colori

- Ogni canale è rappresentato da 32 piani CrCb lungo la componente Y nello spazio de colori.
- Struttura che gestisce i colori OCdtVectorData:





CDT: Metodi utilizzati

- Crea vettore CDT: *OPENR::NewCdtVectorData(&cdtVecID, &cdtVec);*
- Imposta il numero di canali: *SetNumData(1);*
- Recupero l'OCdtInfo: *GetInfo(0);*
- Inizializza l'OCdtInfo: *Init(fbkID, BALL_CDT_CHAN);*
- Imposta un piano: *Set(Y_segment, Cr_max, Cr_min, Cb_max, Cb_min);*
- Memorizza le impostazioni nella telecamera: *OPENR::SetCdtVectorData(cdtVecID);*
- Cancella il vettore: *OPENR::DeleteCdtVectorData(cdtVecID);*

AISAiboYCrCbColorSeg 1/4

- Classe in C++ per il riconoscimento dei colori.
- Scaricabile dal sito dell'AIS-Lab.
- Sono disponibili 32 colori impostabili: l0, l1... l31.
- Metodi da utilizzare:
  - **AISAiboYCrCbColorSeg(int width, int height)**: costruttore. E' necessario specificare la larghezza e altezza dell'immagine da elaborare per ottimizzazioni dell'algoritmo.
  - **bool setColor(Colors selectedLayer,unsigned char y,unsigned char crMin, unsigned char crMax, unsigned char cbMin, unsigned char cbMax)**: imposta i valori di Y Cr e Cb per il colore specificato in selectedLayer.

AISAiboYCrCbColorSeg 2/4

- **const unsigned int* processImage(unsigned char* pImage)**: processa l'immagine:
  - **pImage**: immagine da elaborare. E' il puntatore ritornato dal metodo OfbkImageVectorData::GetData.
  - Ritorna un array di *unsigned int* delle dimensioni in pixel di pImage. Per ogni unsigned int l'i-esimo bit identifica il riconoscimento o meno dell'i-esimo colore impostato con setColor.
- **int getNumberOfRecognizedPixel(Colors selectedLayer)**: ritorna il numero di pixel riconosciuti appartenenti al selectedLayer colore nell'utlima elaborazione.

AISAiboYCrCbColorSeg 3/4

- Di seguito un esempio di impostazione dei colori:

```
AISAiboYCrCbColorSeg segmentator(208, 160);
for(int i=0; i< 256; i++)
   {
   // giallo
   seg.setColor(AISAiboYCrCbColorSeg::l0, i,  135, 165,
                                                50, 98);
   // AiboBall
   seg.setColor(AISAiboYCrCbColorSeg::l1, i, 150, 230,
                                               120, 190);
   }
```

AISAiboYCrCbColorSeg 4/4

- Di seguito un esempio di elaborazione:

```
OFbkImageVectorData* imageVec =
  reinterpret_cast<OFbkImageVectorData*>(const_cast<void*>(event.Data(0)));
byte* data = imageVec->GetData(ofbkimageLAYER_H);
const unsigned char* ptr=segmentator.processImage( static_cast<unsigned
  char*>(data) );
if(segmentator.getNumberOfRecognizedPixel(AISAiboYCrCbColorSeg::l1) =>
  BALL_THRESHOLD)
   {
    for (int y = 0; y < height; y++)
      for (int x = 0; x < width; x++)
         {
            if (*ptr++ & AISAiboYCrCbColorSeg::l1)
               ...
               ...
         }
```


Alcune impostazioni 1/2

- Si impostano attraverso la funzione statica OPENR::ControlPrimitive()
- Possibili impostazioni:
  - oprmreqCAM_SET_WHITE_BALANCE (per impostare il bilanciamento del bianco)
  - oprmreqCAM_SET_GAIN (più alto = migliore qualità dell'immagine con scarsa illuminazione)
  - oprmreqCAM_SET_SHUTTER_SPEED (velocità dell'otturatore)

Alcune impostazioni 2/2

- E' possibile impostare l'Aibo in modo che effettui il bilanciamento automaticamente con i comandi:
  - Bilanciamento automatico del bianco:

  OPENR::ControlPrimitive (fbkID, oprmreqCAM_AWB_ON, 0, 0, 0, 0);

  - Esposizione automatica:

  OPENR::ControlPrimitive (fbkID, oprmreqCAM_AE_ON, 0, 0, 0, 0);
- Purtroppo non sono molto efficienti, la predominanza di un colore nell'immagine confonde il bilanciamento.

Semi Streaming video

- **W3Aibo** (sample di Open-R): server web minimale per Aibo.
- In ascolto sulla porta 60080
- Ad ogni richiesta W3Aibo risponde con l'immagine catturata dalla telecamera in formato **jpeg**
- Offre vari URL:
  - /layerh = layer high - /layerhr = layer high recostructed
  - /layerm = layer mid - /layermr = layer mid recostructed
  - /layerl = layer low - /lyerlr = layer low recostructed
  - /layercdt = layer CDT (solo nella versione modificata)

Semi Streaming video

- Esempio:
  - http://159.149.136.200:60080/layerhr
- Utilizzando il meta tag html "*refresh*" è possibile utilizzare W3Aibo per effettuare una sorta di streaming via WEB

```
<html> <head>
<META HTTP-EQUIV="Refresh" CONTENT="0;./<nomefile>.html">
</head>
<body>
<img src="http://192.168.254.5:60080/layerhr" width=416 height=320>
</body></html>
```


Semi Streaming video

- Vantaggi:
  - Semplice da installare sull'Aibo (un solo oggetto OPEN-R con una sola connessione a OVRC)
  - Semplice da utilizzare lato client (e' sufficiente un browser)
  - Non dipendente dal sistema operativo
- Svantaggi:
  - Streaming poco fluido.

Alcuni link utili

- Sul sito dell'Ais lab: http://ais-lab.dsi.unimi.it/
  - Documento sulla visione: **Aibo_Imaging_v1.3.pdf**
  - Documento sulla impostazione dei parametri della telecamera: **CameraParameter_v1.0.pdf**
  - **Slide della Carnegie Mellon sulla visione: Vision.pdf**

Esercitazione 4 (Descrizione)

- Applicazione per la ricerca della palla e allienamento dell'Aibo.
  - Muove la testa alla ricerca della Aibo ball
  - Una volta individuata allinea il corpo dell'Aibo nella direzione della Palla.
- Utilizza la Color Detection Table hardwired dell'Aibo per la segmentazione dei colori.
- Utilizza Motion della Carnegie Mellon per la gestione dei Joint.

Esercitazione 4 (Diagramma)