Comandare gli Aibo

Come gestire i sottosistemi

- I passi standard da seguire
  1. Aprire le primitive e convertirle
     - *Telecamera: PRM:/r1/c1/c2/c3/i1-FbkImageSensor:F1*
     - Le primitive vanno convertite in oggetti **OPrimitiveID**
  2. Comunicare con gli oggetti che gestiscono la primitiva (mandare comandi, ricevere valori).
- Per informazioni sulle specifiche e gli indirizzi delle primitive dei vari sottosistemi: **ModelInformation_7_E.pdf** (nella documentazione di Open-R)

Come gestire i sottosistemi

- Il calcolo del tempo negli Aibo è discreto. L'unità base è il **frame** (8ms).
- I valori dei sensori, i comandi ai Joint e alla Telecamera sono discretizzati dai frame
- L'accesso ai sottosistemi avviene a blocchi di frame.
- I sensori ritornano un `osensorMAX_FRAME` (16) numero di frame alla volta
- I Joint si comandano con un `ocommandMAX_FRAME` (16) numero di frame alla volta
- La temporizzazione dei sottosistemi non è necessariamente sincronizzata con il sistema nel suo insieme (*p.es la telecamera può essere al frame 15 mentre il sensore di distanza sul 25*).

OVirtualRobotComm

- Oggetto Open-R di Sistema
- Gestisce: joint, sensori, LED e Telecamera
- Offre vari servizi in relazione al sottosistema interessato

Servizi di OVirtualRobotComm

- servizio che gestisce i Joint e i LED:
  - OVirtualRobotComm.Effector.OCommandVectorData.O
- servizio che gestisce i Sensori:
  - OVirtualRobotComm.Sensor.OSensorFrameVectorData.S
- servizio che gestisce la Telecamera:
  - OVirtualRobotComm.FbkImageSensor.OFbkImageVector Data.S

OVirtualRobotAudioComm

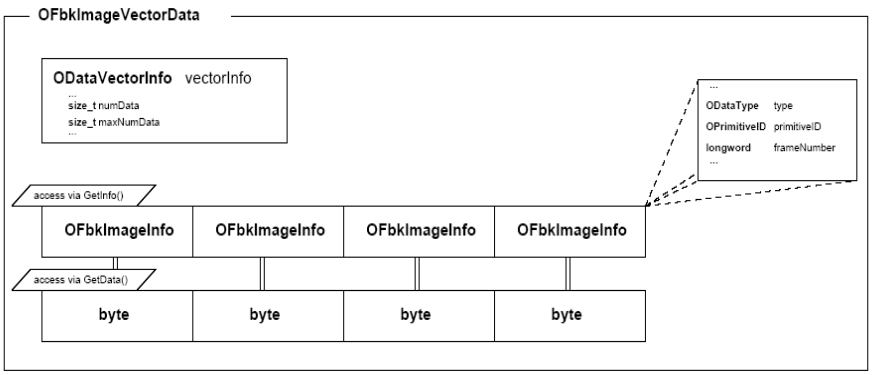
- Oggetto Open-R di sistema
- Gestisce il sottosistema audio
- Offre i seguenti servizi:
  - OVirtualRobotAudioComm.Speaker.OSoundVect orData.O
  - OVirtualRobotAudioComm.Mic.OSoundVectorDa ta.S

Gestire la Telecamera

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.

Struttura dei dati della telecamera



- ofbkimageLAYER_H, ofbkimageLAYER_M, ofbkimageLAYER_L, ofbkimageLAYER_C
- Oggett OFbkImage per gestire le immagini

OFbkImage

- Costruttore: esempio
  - `OFbkImage* img(vector->GetInfo(ofbkimageLAYER_M), vector->GetData(ofbkimageLAYER_M), ofbkimageBAND_Y)`
  - `IsValid()`
  - `Pointer()`
  - `Width()`
  - `Height()`
  - `Pixel(x,y)`
  - `...`

Alcune impostazioni

- Si impostano attraverso la funzione statica OPENR::ControlPrimitive()
- Possibili impostazioni:
  - oprmreqCAM_SET_WHITE_BALANCE (per "calibrare" i colori in base alla luce)
  - oprmreqCAM_SET_GAIN (più alto = migliore qualità dell'immagine con scarsa illuminazione)
  - oprmreqCAM_SET_SHUTTER_SPEED (velocità dell'otturatore)

Esempio

- Sample ImageCapture
  - Cattura le immagini ogni volta che viene toccato l'Aibo sul dorso o sulla testa.
  - Immagini in: bmp High resolution, e Raw in High, Med, e Low resolution

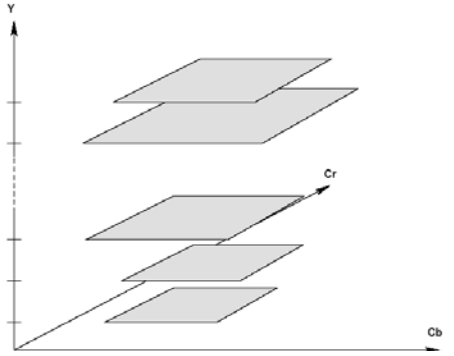
Gestione del Color Detection

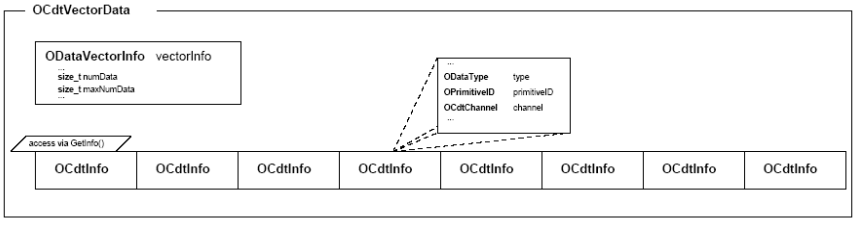
- Si impostano i valori dei canali (colori), sono possibili 8 canali.
- Si recupera il Layer Color Detection
- Per ogni pixel (1byte) del layer si applica una maschera:

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

Impostare i colori

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






Gestire il movimento

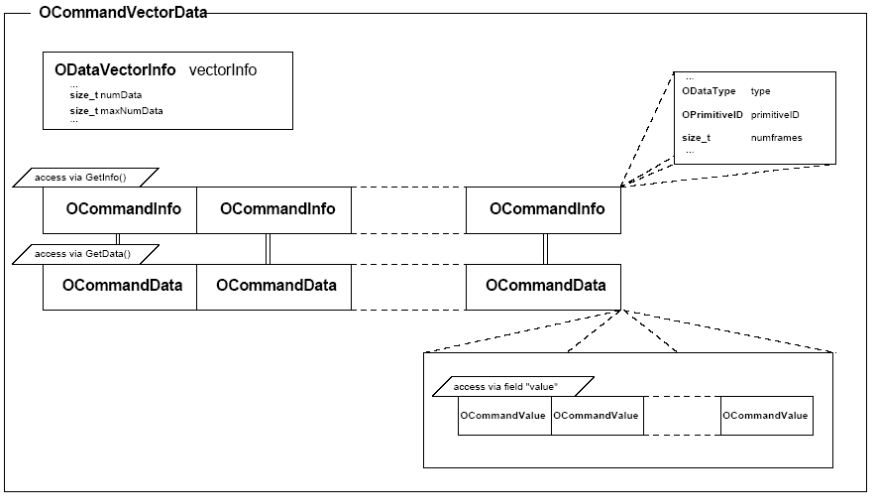
I Joint

- **Left front leg**
  - PRM:/r2/c1-Joint2:21 Left front legJ1
  - PRM:/r2/c1/c2-Joint2:22 Left front legJ2
  - PRM:/r2/c1/c2/c3-Joint2:23 Left front legJ3
- **Left rear leg**
  - PRM:/r3/c1-Joint2:31 Left rear legJ1
  - PRM:/r3/c1/c2-Joint2:32 Left rear legJ2
  - PRM:/r3/c1/c2/c3-Joint2:33 Left rear legJ3

I Joint

- **Right front leg**
  - PRM:/r4/c1-Joint2:41 Right front legJ1
  - PRM:/r4/c1/c2-Joint2:42 Right front legJ2
  - PRM:/r4/c1/c2/c3-Joint2:43 Right front legJ3
- **Right rear leg**
  - PRM:/r5/c1-Joint2:51 Right rear legJ1
  - PRM:/r5/c1/c2-Joint2:52 Right rear legJ2
  - PRM:/r5/c1/c2/c3-Joint2:53 Right rear legJ3

Struttura dei dati dei Joint



Comunicazione con OVirtualRobotComm

- Per ridurre la dimensione dei dati passati con la comunicazione inter-object e velocizzare il processo spesso si usa la memoria condivisa: **RCRegion**.
- I passi da seguire:
  1. Creare un vettore dei comandi
  2. Mappare tale vettore in una zona di memoria condivisa dell'oggetto.

Esempio di codice

```
void MySampleClass::NewCommandVectorData( ) {
   OStatus result ;
   MemoryRegionID cmdVecDataID ;
   OCommandVectorData□ cmdVecData ;
   OCommandInfo□ info ;
   result=OPENR::NewCommandVectorData (NUM_JOINTS,
                        &cmdVecDataID ,
                        &cmdVecData ) ;
   if ( result == oSUCCESS)
   {
        region= new RCRegion( cmdVecData->vectorInfo.memRegionID ,
                cmdVecData->vectorInfo. offset,
                (void □) cmdVecData ,
                cmdVecData->vectorInfo.totalSize);
        cmdVecData->SetNumData(NUM_JOINTS) ;
        for ( int j = 0 ; j < NUM_JOINTS; j++)
        {
                info = cmdVecData->GetInfo( j ) ;
                info->Set(odataJOINT_COMMAND2,
                        jointID [ j ] ,
                        ocommandMAX_FRAMES) ;
        }
   }
}
```

I passi da seguire per gestire i Joint

- Per gestire i Joint degli Aibo bisogna seguire alcuni passi standard:
  1. Conversione primitive
  2. Creazione zona di memoria condivisa
  3. Impostare i gain
  4. Calibrare i Joint
  5. Impostare gli angoli dei Joint

Conversione delle primitive

- Convertire le primitive dei Joint in primitiveID
  - Funzione *openPrimitive();*
- Accendere i motori
  - Funzione *OPENR::SetMotorPower(opowerON);*

Creare la zona di memoria condivisa

```
void MySampleClass::NewCommandVectorData( ) {
   OStatus result ;
   MemoryRegionID cmdVecDataID ;
   OCommandVectorData cmdVecData ;
   OCommandInfo info ;
   result=OPENR::NewCommandVectorData (NUM_JOINTS,
                        &cmdVecDataID ,
                        &cmdVecData ) ;
   if ( result == oSUCCESS)
   {
        region= new RCRegion( cmdVecData->vectorInfo.memRegionID ,
                cmdVecData->vectorInfo. offset,
                (void ) cmdVecData ,
                cmdVecData->vectorInfo.totalSize);
        cmdVecData->SetNumData(NUM_JOINTS) ;
        for ( int j = 0 ; j < NUM_JOINTS; j++)
        {
                info = cmdVecData->GetInfo( j ) ;
                info->Set(odataJOINT_COMMAND2,
                        jointID [ j ] ,
                        ocommandMAX_FRAMES) ;
        }
   }
}
```


Impostare Joint gain

- Impostare i limiti dei motori.
- La documentazione consiglia di impostare i valori standard
  - Funzione *setJointGain();*

Calibrare i Joint

- All'avvio può succedere che ci sia una differenza tra la posizione reale dei Joint e la posizione letta dai sensori.
- E' sufficiente leggere i valori dei sensori e posizionare i motori in base ai valori letti
  - Funzione *adjustDiffJointValue();*

Impostare gli angoli dei Joint

- Ora si possono mandare i comandi ai Joint.
- I Joint accettano angoli espressi in micro radianti (10^{-6} rad).
  - Funzione *setJointValue();*


Gestire i Sensori

Come gestire i Sensori

- Come gli altri sottosistemi si accede al loro attraverso il servizio di OVirtualRobotComm:
  - OVirtualRobotComm.Sensor.OSensorFrameVectorData.S
- Si aprono le primitive
- Si ricevono i messaggi.
- Esempio:
  - ./ers7/SensorObserver7

Struttura dei dati dei Sensori