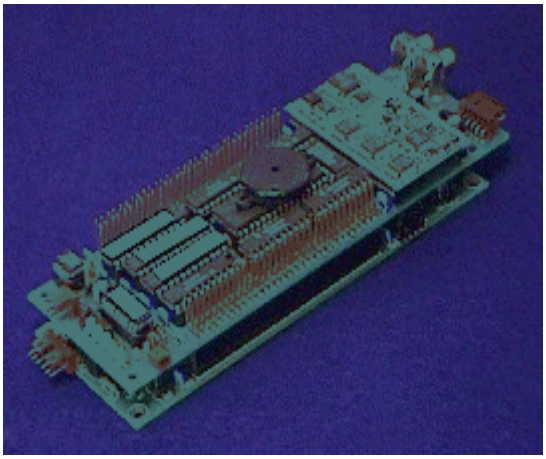
Esercitazione di Robotica

Docente Prof. Riccardo Cassinis



"Cognachrome vision system"

FOLLOW MY "RABBIT"

Carlo Gatta

Giuseppe Medeghini

Gianluca Vezzoli

1. INTRODUZIONE

1.1 OBIETTIVO

L'obiettivo del nostro lavoro è la realizzazione di un robot in grado di inseguire una macchinina radiocomandata. Sulla macchinina è stato montato un cilindro di un vivido colore arancio; il robot inseguirà in pratica questo cilindro colorato.

1.2 MATERIALE UTILIZZATO

Per la realizzazione del progetto abbiamo utilizzato:
- robot Speedy di tipo Pioneer1;
- sistema di visione Cognacrhome installato sul robot;
- sistema PC-Linux;
- sistema mini-Arc per la gestione del Cognachrome.

2. UTILIZZO DEL SISTEMA DI VISIONE COGNACRHOME

2.1 UTILIZZO DEL COGNACHROME

Il cognachrome è un sistema hardware in grado di acquisire ed elaborare immagini video provenienti dalla telecamera montata su Speedy. Le funzionalità del cognachrome sono limitate ma hanno il vantaggio di essere implementate direttamente sul 68332 montato su scheda. Questo consente di utilizzare la potenza di calcolo dell'elaboratore che "guida" il robot senza dover fare calcoli sull'intera immagine.
L'utilizzo del Cognachrome è semplice e necessita di poca conoscenza per essere sfruttato al meglio. Per avere un insieme di informazioni sull'utilizzo e le funzionalità si consulti la relazione dal titolo "TEST SU PRESUNTI MALFUNZIONAMENTI DEL SISTEMA DI VISIONE COGNACHROME". Per le istruzioni fornite dal Saphira riguardanti il cognachrome consultare il manuale Saphira oppure vedere il codice commentato a pag. 8.

2.2 TARATURA DEL COGNACHROME

La taratura del cognachrome consiste semplicemente nell'"addestrare" il sistema di visione a riconoscere un determinato colore (nella modalità blob o bb_blob); il sistema in seguito fornisce un insieme di informazioni via seriale. Il Saphira fornisce inoltre un insieme di funzioni e strutture dati per il controllo e la lettura dei dati forniti dal Cognachrome. Per sfruttare al meglio e più in dettaglio le funzioni offerte dal Cognachrome è preferibile fare riferimento al manuale.

2.3 DATI FORNITI DAL COGNACRHOME E ORGANIZZATI DA SAPHIRA

Il sistema Cognachrome fornisce dati riguardanti tre canali (a,b,c), ognuno dei quali può essere "addestrato" a riconoscere un colore. Saphira predefinisce tre variabili di tipo *vinfo* ognuna delle quali contiene i dati riguardanti un canale.
La struttura *vinfo* è così definita:

```
struct vinfo
        {
        int type;               /* indica in che modalità sta funzionando il canale */
                                /* BLOB, BLOB_BB, LINE MODE */
        int x,y;        /* Sono le coordinate del centro di massa del blob individuato */
        int area;               /* E' l'area in pixel del blob individuato */
        int h,w;                /* Sono altezza e larghezza apparenti del riquardo del blob */
        int first, num;         /* Sono la prima e il numero di linee individuate nella modalità */
                                /* LINE MODE */
        }
```

In particolare le tre variabili si chiamano sfVaInfo, sfVbInfo, sfVcInfo rispettivamente per i canali a, b, c;

3 PROGRAMMAZIONE

3.1 FUNZIONALITÀ DEL SAPHIRA UTILIZZATE

Le funzioni di Saphira che abbiamo utilizzato sono:

sfSetRVelocity (int RVEL)

sfSetVelocity (int VEL)

settano rispettivamente la velocità di rotazione (in gradi/ sec) e di movimento (in mm/sec) nella struttura che contiene i dati del robot.

sfSetMaxVelocity(int MAXVEL)

setta la massima velocità raggiungibile dal robot (in mm/sec).

sfRobotComStr(int com, char *str)

questa funzione saphira manda una stringa di comando al robot;

com e *str assumono diversi significati a seconda del tipo di comando che si vuole utilizzare; nel nostro caso li usiamo per inizializzare il cognachrome.

sfAddEvalFn(char *nome,void *fn, int Rtype, int Nargs,....)

è la funzione che rende accessibile al colbert le funzioni scritte in C;

*nome è il nome che si vedrà nell'ambiente Saphira;

fn è il nome della funzione;

Rtype è il tipo ritornato dalla funzione;

Nargs è il numero di argomenti passati alla funzione;

dopo Nargs vanno messi gli argomenti della funzione.

3.2 ALTRE LIBRERIE

Per gentile concessione dei tesisti dell'LDRA abbiamo usato la libreria pio.h, che ci mette a disposizione le funzioni pioConnetRobot() e pioDisconnectRobot() che gestiscono il lancio del programma Saphira connesso direttamente con il Pioneer via seriale (la libreria fornisce anche alcune funzioni di movimentazione elementare da noi non utilizzate).
All' interno di pio.c si trovano i programmi sorgenti di pio.h che permettono, tramite opportune modifiche, di far partire Saphira collegato al robot o al simulatore.

3.3 ESEGUIBILE E SHARED OBJECT

Per la nostra esercitazione abbiamo sviluppato due programmi dal risultato praticamente identico (l'inseguimento della macchina).
Nel primo caso il programma ha il pieno controllo del robot ed esegue solo la funzionalità di inseguimento richiamando le funzioni descritte in 2.4, viene compilato tramite un makefile appropriato che include le librerie e crea l'eseguibile.
Nel secondo caso invece con un programma strutturato in modo diverso creiamo uno shared object che viene caricato in Saphira e che definisce la funzione insegui invocabile come una normale activities. In questo caso insegui deve però condividere il controllo del robot con gli altri normali behavior (es: avoid collision).

4 FUNZIONAMENTO

Il robot è controllato in velocità, sia di traslazione che di rotazione. Per determinarle facciamo delle considerazioni sull'area apparente del blob in quell'istante e sulla sua posizione apparente. Per migliorarne il comportamento ed evitare che "scodi" o si comporti in maniera anomala abbiamo tracciato le seguenti funzioni non lineari per determinare la velocità di rotazione e di traslazione:
_il grafico n°1 rappresenta la funzione per la velocità di tralsazione,
_il grafico n°2 rappresenta la funzione per la velocità di rotazione.

Queste funzioni sono un compromesso tra reattività e stabilità, risultato di lunghi test atti a migliorare questi due
obiettivi.

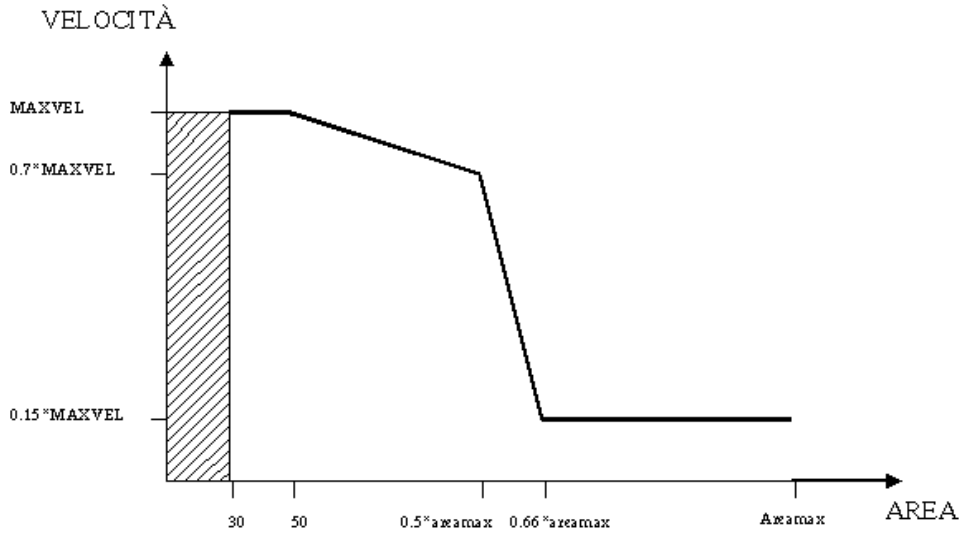


Grafico n°1

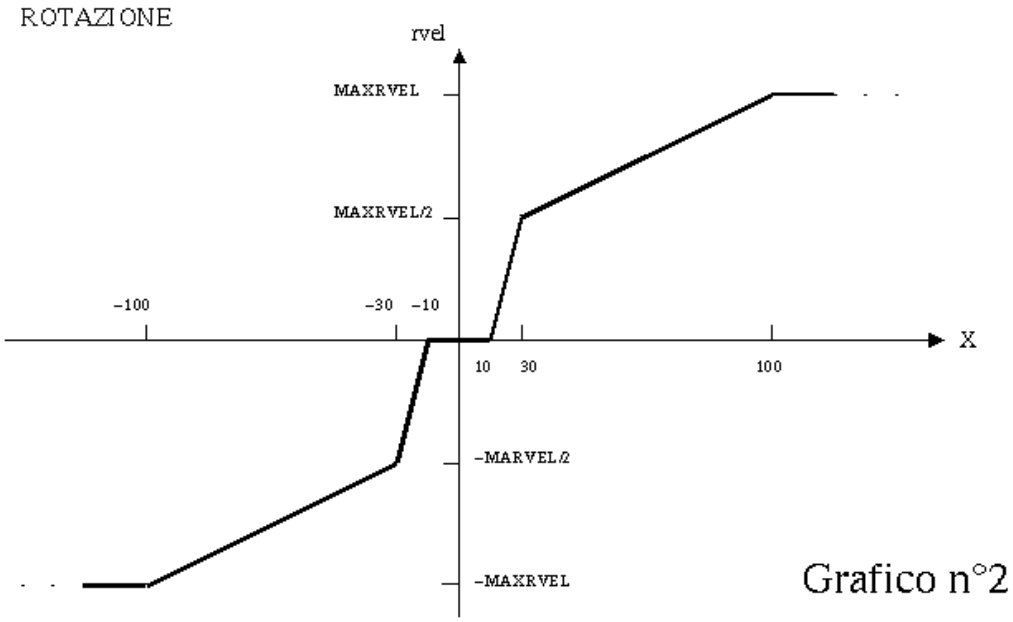


Grafico n°2

5 LIMITI

1. Se l'ambiente è male illuminato o in caso di abbagliamento il sistema di visione Cognachrome risponde in maniera anomala e non restituisce valori di posizione ed area corrispondenti a quelli reali.
2. se nell'ambiente ci sono oggetti di colore simile al colore del blob considerato, il Cognachrome li può confondere e deviare il robot verso questi.
3. l'area del blob non deve essere troppo piccola, perchè la telecamera montata su Speedy ha un obiettivo a focale corta, che rimpicciolisce la dimensione degli oggetti già ad una breve distanza.

6. APPENDICE

A Codice "Autonomo"

```
#include "saphira.h"

#define MAXVEL 1000
#define MAXRVEL 120
#define AREAMAX 550

/*************************************************************/
/* MAXVEL   definisce la velocità massima del robot (fino a 3000)*/
/*          espressa in mm/sec                                */
/* MAXRVEL  velocità di rotazione (max 180 °/sec)              */
/* AREAMAX  area apparente del blob alla distanza di inseguimento*/
/*************************************************************/

/* Definizione delle variabili globali                       */
/* v= velocità di traslazione del robot                      */
/* g= velocità di rotazione                                  */
/* vecchiog= ultima direzione relativa della macchina        */
/* area= area apparente del blob vista dal Cognachrome       */

int v,x,y,g,vecchiog,area;

/*************************************************************/
/* area_to_vel calcola la velocità da assumere per inseguire      */
/* l'oggetto. Area è il valore dell'area apparente restituita     */
/* dal Cognachrome. La funzione restituisce la velocità che il    */
/* robot deve mantenere                                      */
/*************************************************************/
int area_to_vel(int area)
{
/*se l'area è minore di 30 abbiamo perso il blob e lo cerchiamo      */
/*ruotando nell'ultima direzione in cui stava andando la macchinina  */

if(area<30)
  {
    sfSetRVelocity(abs(vecchiog)/(vecchiog+1)*MAXRVEL);
    printf("LA STO CERCANDO\n");
    return(0);
  }

/* implementa la funzione disegnata  nel grafico 1                  */

 if(area<50)return(MAXVEL);
 if(area<0.5*AREAMAX) return(MAXVEL-((area-50)*0.3*MAXVEL/(0.5*AREAMAX-50)));
 if(area<0.66*AREAMAX) return(0.7*MAXVEL-((area-300)*0.55*MAXVEL/(0.16*AREAMAX)));
 if(area<AREAMAX) return(0.15*MAXVEL);
return(0);
}


/*************************************************************/
```

```
/* x_to_rvel è la funzione che restituisce la velocità di rotazione */
/* per inseguire il blob                                            */
/* a questa funzione viene passata la posizione apparente del blob  */
/* nell'immagine ottenuta da Cognachrome e restituisce la velocità  */
/* rotazione del robot                                              */
/********************************************************************/

int x_to_rvel(int x)
{
int sign;

/*ricaviamo il segno della rotazione (x espresso rispetto all'asse  */
/* verticale dell'immagine                                          */

if (x>0)sign=1;
 else
   {
     x=-x;
     sign=-1;
   }

/* implementa la funzione disegnata  nel grafico 2                   */

if (area<30) return(g);
if(x<10)return(0);
if(x<30)return(sign*(x-10)*MAXRVEL/40);
if(x<100)return(sign*(MAXRVEL/2+(x-30)*MAXRVEL/140));
return(sign*MAXRVEL);
}

/********************************************************************/
/* settavel() usa i due comandi base della libreria Saphira per settare*/
/* la velocità di traslazione e rotazione                            */
/********************************************************************/
void settavel()
{
  sfSetRVelocity(g);
  sfSetVelocity(v);
}


/********************************************************************/
/* setup()connette il robot, setta le velocità di punta e pone la sua  */
/* velocità iniziale a zero                                         */
/********************************************************************/
void setup()
{
  pioConnectRobot();
  sfRobotComStr(VISION_COM,"pioneer_a_mode=0");

  sfSetMaxVelocity(MAXVEL);
  sfSetMaxRVelocity(MAXRVEL);

  sfSetVelocity(0);
  sfSetRVelocity(0);
```

```
}

/*************************************************************************/
/*insegui() è la funzione principale che legge i valori della struttura */
/*sfVaInfo, calcola g e v con le funzioni precedenti e le imposta con   */
/*settavel()                                                             */
/*************************************************************************/

void insegui()
{
  x=0;
  y=0;

  while(TRUE==TRUE)
    {
      x=sfVaInfo.x;
      y=sfVaInfo.y;

      area=sfVaInfo.area;
      vecchiog=g;
      g=x_to_rvel(x);
      v=area_to_vel(area);
      settavel();
      printf("V:%d   R:%d      \n",v,g);
    }
}
void main(void)
{
  setup();
  insegui();
  pioDisconnectRobot();
};
```

B Makefile per il listato autonomo

Questo è il makefile necessario per compilare il listato autonomo in modo da ottenere un file eseguibile prova.

Lanciando prova viene aperta una finestra con saphira ed inizia l'inseguimento della macchinina.

ATTENZIONE: con questa versione non sono attivi i behavior.

NB: /fwm è la directory che contiene i nostri sorgenti ovvero il codice autonomo e tutte le altre librerie usate.

```
#-----------------------------------------------------------------
#
#  Follow My Rabbit      :              makefile per compilato autonomo
#
#  Last update 2000 May 11
#
#-----------------------------------------------------------------

CC = gcc
OFLAGS = -O -g -DUNIX -DLINUX

X11D = /usr/X11
BIND = $(HOME)/fwm/
SRCD = $(HOME)/fwm/
OBJD = $(HOME)/obj/
INCD = -I$(SAPHIRA)/handler/include/ -I$(X11D)include/ -Iinclude/
LIBD = -L$(SAPHIRA)/handler/obj/ -L$(X11D)lib/ -L/lib/ -Llib/
COLBERT = $(SAPHIRA)/colbert/
OAALIB = $(SAPHIRA)/oaa/agents/lib/
LIBS = -ldl  -lm -lc -lXm -lXt -lX11 -lXext -lXpm  -lsf

include $(SAPHIRA)/handler/include/os.h

CFLAGS = -g -D$(CONFIG) $(OFLAGS) $(INCD)
INCLUDE = -I$(INCD) -I$(X11D)include
# executable file
all: $(BIND)fwm
        touch all
# sources
$(OBJD)prova.o : $(SRCD)prova.c
        $(CC) -c $(CFLAGS) $(SRCD)prova.c  $(INCLUDE) -o $(OBJD)prova.o
$(OBJD)pio.o : $(SRCD)pio.c
        $(CC) -c $(CFLAGS) $(SRCD)pio.c  $(INCLUDE) -o $(OBJD)pio.o
# if  more .c files must be compiled, copy the two rows above and change the file name,
# then update the obj files raw below.
# object files
OBJS = $(OBJD)prova.o $(OBJD)pio.o $(OBJD)mychroma.o
# don't touch this!!!
$(BIND)fwm : $(OBJS)
        $(CC) $(OBJS) -o $(BIND)prova  \
-L$(MOTIFD)lib $(LIBD) -lsf $(LIBS) -lc -lm

clean:
        -rm -f $(OBJD)* *~ $(SRCD)*~
        -rm -f $(BIND)fwm  $(BIND)*.pgm $(BIND)*.ppm
```

C Codice "Shared Object"

```
#include "saphira.h"

int v,x,y,g,vecchiog,area,areamax,maxvel,maxrvel;

int area_to_vel(int area) // Funzione che calcola la velocità da assumere per
inseguire l'oggetto
{
if(area<30)
  {
    sfSetRVelocity(abs(vecchiog)/(vecchiog+1)*maxrvel);
    printf("LA STO CERCANDO\n");
    return(0);
  }

 if(area<50)return(maxvel);
 if(area<0.5*areamax) return(maxvel-((area-50)*0.3*maxvel/(0.5*areamax-50)));
 if(area<0.66*areamax) return(0.7*maxvel-((area-300)*0.55*maxvel/(0.16*areamax)));
 if(area<areamax) return(0.15*maxvel);

return(0);
}

int x_to_rvel(int x) // Funzione che calcola la velocità di rotazione da assumere
per inseguire l'oggetto
{
int sign;

if (x>0)sign=1;
 else
   {
     x=-x;
     sign=-1;
   }

if (area<30) return(g);
if(x<10)return(0);
if(x<30)return(sign*(x-10)*maxrvel/40);
if(x<100)return(sign*(maxrvel/2+(x-30)*maxrvel/140));
return(sign*maxrvel);
}

void settavel()
{
  sfRobotComInt(sfCOMVEL,v);
  sfRobotComInt(sfCOMRVEL,g);
}
/*********************************************************************/
/* la funzione insegui ha subito dei cambiamenti:                    */
/* è sparito il ciclo, perchè viene richiamata ogni 100 msecondi dal  */
/* microtask di saphira                                              */
/* quando viene richiamata controlla la posizione del blob, calcola  */
/* le velocità e le setta                                            */
/*********************************************************************/
```

```
void insegui(int a, int mv, int mrv)
{
  areamax=a;
  maxvel=mv;
  maxrvel=mrv;

  x=0;
  y=0;

  x=sfVaInfo.x;
  y=sfVaInfo.y;

  area=sfVaInfo.area;
  vecchiog=g;
  g=x_to_rvel(x);
  v=area_to_vel(area);

  settavel();

}

/*****************************************************************/
/* questa è una parte fondamentale; con questo export rendiamo  */
/* visibile a saphira la funzione insegui e lanciamo i behavior */
/* necessari (come avoidcollision)                               */
/*****************************************************************/

EXPORT void sfLoadInit(void)
{
sfRobotComStr(VISION_COM,pioneer_a_mode=BLOB_MODE);
sfStartBehavior(sfAvoidCollision,"NostroAvoid",0,0,0,4,4,5,400,sfLEFTTURN);
sfAddEvalFn("insegui",insegui,sfVOID,3,sfINT,sfINT,sfINT);

}

/*********************************************************************/
/* in questo listato la funzione main non serve, per cui rimane vuota */
/*********************************************************************/
void main(void)
{
};
```

D Makefile per listato shared object

```
#-------------------------------------------------------------
#
#  Follow My Rabbit
#
#  Last update 2000 May 11
#
#-------------------------------------------------------------

CC = gcc
OFLAGS = -O -g -DUNIX -DLINUX

X11D = /usr/X11
BIND = $(HOME)/fwmbeh/
SRCD = $(HOME)/fwmbeh/
OBJD = $(HOME)/obj/
INCD = -I$(SAPHIRA)/handler/include/ -I$(X11D)include/ -Iinclude/
LIBD = -L$(SAPHIRA)/handler/obj/ -L$(X11D)lib/ -L/lib/ -Llib/
COLBERT = $(SAPHIRA)/colbert/
OAALIB = $(SAPHIRA)/oaa/agents/lib/
LIBS = -ldl  -lm -lc -lXm -lXt -lX11 -lXext -lXpm  -lsf

include $(SAPHIRA)/handler/include/os.h

CFLAGS = -g -D$(CONFIG) $(OFLAGS) $(INCD)
INCLUDE = -I$(INCD) -I$(X11D)include

# executable file
all: $(BIND)fwmbeh
        touch all

# sources
$(OBJD)prova.o : $(SRCD)prova.c
        $(CC) -c $(CFLAGS) $(SRCD)prova.c  $(INCLUDE) -o $(OBJD)prova.o

# if  more .c files must be compiled, copy the two rows above and change the file name,
# then update the obj files raw below.

# object files
OBJS = $(OBJD)prova.o

# don't touch this!!!
$(BIND)fwmbeh : $(OBJS)
        $(CC) $(OBJS) -o $(BIND)prova  \
-L$(MOTIFD)lib $(LIBD) -lsf $(LIBS) -lc -lm

clean:
        -rm -f $(OBJD)* *~ $(SRCD)*~
        -rm -f $(BIND)fwmbeh  $(BIND)*.pgm $(BIND)*.ppm
```

NB: questo makefile genera il file prova.o; per ottenere il file prova.so lanciare il comando "ld prova.o -shared -o prova.so"

INDICE