# RoboWii 2.1

Andrea Pontecorvo

# Indice:

**Capitolo 1**

# Introduzione

RoboWii 2.0.1 si basa sull'interazione tra un giocatore umano e il robot mobile Spykee tramite l'utilizzo di un controller Wiimote e di marker colorati indossati dal giocatore.

In questa versione del RoboWii, che deriva da RoboWii 1.0 si è scelto di utilizzare un robot commerciale ( Spykee ) per sviluppare un nuovo tipo di gioco basato sull'interazione tra robot e un giocatore umano.

# Capitolo 2

*Obiettivi*

Il gioco RoboWii 2.0.1 si tratta di una variante del classico gioco del nascondino: in questo caso è il robot che cerca di nascondersi dal giocatore per un tempo sufficiente a fargli guadagnare un punto. Il giocatore dovrà invece cercare di "colpirlo" utilizzando il WiiMote guadagnando in questo modo un punto a suo favore.
Il gioco termina quando uno dei due partecipanti ( umano o robot ) raggiunge un punteggio prestabilito.

*Descrizione generale & regole*

Come detto precedentemente il gioco è una variante del nascondino, quindi le regole di base sono le stesse: il robot cerca di individuare il giocatore, tramite i marker che quest'ultimo indossa sul petto e sulla schiena, o rilevando di essere stato puntato con il WiiMote.
Quando il giocatore individua il robot dovrà puntarlo tramite il WiiMote e, tenendolo sotto tiro, tenere premuto il pulsante di fuoco per un tempo sufficiente a "caricare" il colpo vincendo il match e ottenendo un punto; in questo caso il robot inizierà un nuovo match, in caso contrario cercherà di nascondersi.
Il gioco è progettato per essere svolto principalmente al chiuso, in un ambiente che offra al robot diversi possibili nascondigli ( ad esempio mobili ), ben illuminato e non eccessivamente vasto.

## Comportamenti del robot

In questa sezione vengono descritti nel dettaglio i comportamenti del robot durante il gioco e la sua interazione con giocatore e ambiente circostante

*Fuga per contato visivo*
Quando il giocatore si trova davanti al robot, questo sarà in grado di vederlo ( grazie al marker indossato ) e reagirà girandosi per allontanarsi dal giocatore ed eventualmente nascondersi, cercando di tenere il giocatore fuori dal proprio campo visivo.

*Fuga per rilevazione IR*
Quando il giocatore punta Spykee con il WiiMote, il robot cercherà di scappare dando origine a un comportamento uguale a quello della regola precedente.

*Ricerca giocatore*
Quando il gioco viene avviato il robot si pone in uno stato di ricerca: durante questa fase compie delle lente rotazioni sul posto per individuare, grazie ai marker, il giocatore. Se durante la rotazione il robot si trova ad avere la visuale ostruita da un ostacolo invertirà la direzione di ricerca. Questa fase si interrompe quando il giocatore viene individuato oppure quando il robot viene puntato tramite gli IR.

*Il robot viene colpito ( punto per il giocatore )*
Quando il giocatore riesce a tenere sotto tiro il robot per un tempo sufficiente a caricare il colpo, il robot risulta colpito e di conseguenza il giocatore ottiene un punto, che viene indicato tramite i led blu del WiiMote.
Una volta segnalato il punteggio e dopo un tempo di attesa di qualche secondo il robot si rimette alla ricerca del giocatore e il gioco riprende per il match successivo.

*Spykee si nasconde*
Quando il robot rileva il giocatore ( tramite contatto visivo ) o si accorge di essere puntato ( rilevazione IR ) cercherà di allontanarsi in una direzione casuale ( O NO? ). Se durante la fuga, che avviene in linea retta, dovesse rilevare delle cavità ( per la presenza di mobili o di altri ostacoli ) sul lato destro o sinistro si fermerà e si dirigerà verso di esse in modo da essere almeno parzialmente coperto alla vista del giocatore.
Una volta “nascosto” ricomincerà a cercare il giocatore.

*Ricerca del giocatore*
Questo comportamento viene eseguito all’avvio della partita, dopo che il robot viene colpito e ogni volta che ha raggiunto un nascondiglio. Spykee gira su se stesso in modo da individuare il giocatore tramite la telecamera. Se girando dovesse rilevare ( tramite i sonar ) che la telecamera andrà a guardare un ostacolo troppo vicino fermerà la rotazione e la riprenderà in senso opposto, in modo da osservare solo spazio libero.
Se durante la ricerca Spykee individua il giocatore tramite contatto visivo o se dovesse venire puntato cercherà di fuggire e nascondersi.

# Capitolo 3

## 3.1 Hardware

In questa sezione verrà illustrato l'hardware utilizzato per la realizzazione del progetto.

### *3.1.1 Spykee*

Il robot utilizzato è un modello commerciale ( modificato, sez. 3.1.2 ) prodotto dalla Meccano: Spykee.

Figura 1: Spekee

Spykee è un robot autonomo mobile di forma umanoide dotato di due braccia fisse, di una luce bianca e di una telecamera a colori con una risoluzione di 320x240 px ( entrambe montate sulla tesa ). Il robot dispone inoltre di alcune luci colorate diffuse tramite fibre ottiche ( non utilizzate per il progetto ). E' possibile inoltre far emettere al robot alcuni suoni, ad esempio una sirena d'allarme o un esplosione.
Il sistema di movimento del robot è di tipo differential drive: i due cingoli sono controllati indipendentemente l'uno dall'altro permettendo al robot di girare su se stesso o di effettuare curve con raggio variabile. La velocità massima in linea retta è di circa 30 cm al secondo.
Il robot ha un autonomia di circa 30 minuti e viene ricaricato tramite una base collegata alla rete elettrica.
Per collegarsi al computer Spykee crea una connessione wi-fi di tipo ad hoc ed è controllabile anche tramite il software proprietario fornito col robot.

### *3.1.2 Sonar e led*

Per poter essere utilizzato in diversi progetti sono state necessarie diverse aggiunte al robot Spykee.
Il robot è stato equipaggiato con due led infrarossi, posizionati sulle spalle, in modo da poter essere individuato dal wiimote, inoltre essendo nota la distanza tra i due led è possibile stimare la distanza robot-giocatore.
Oltre ai led IR sono presenti un led verde, che indica quando il robot viene colpito, e un gruppo di quattro led rossi ( montati frontalmente ) che indicano il livello di carica del colpo quando il giocatore tiene Spykee puntato.
Per rilevare le distanze ( e quindi gli ostacoli ) sono stati montati diversi sensori sonar sul robot con un raggio di circa 7 metri e un ampiezza di circa 30°. Questi sensori indicano la distanza dell'oggetto più vicino che si trova nel loro raggio d'azione.
Attualmente sono utilizzati sei sensori:
tre frontali: utilizzati principalmente per rilevare gli ostacoli di fronte a Spykee e per eventualmente essere utilizzati insieme alla telecamera.
due sensori laterali, uno a destra e uno a sinistra: utilizzati per la rilevazione delle cavità e quindi di possibili nascondigli
uno posteriore: utilizzato per rilevare gli ostacoli.

Tutte le aggiunte sono alimentate tramite tre batterie stilo ( indipendenti quindi dalla ricarica del robot ) e comunicano i dati al computer tramite una scheda XBee ( sez. 3.1.3 ).

### *3.1.3 XBee*

Per poter trasmettere i dati dei sensori al computer e per poter comandare i led di Spykee è stata utilizzata una coppia di schede XBee ( una collegata al computer e una a Spykee ). Queste schede permettono di stabilire una connessione punto-punto tra di loro e si comportano come se fossero una porta seriale.
I dati vengono inviati e ricevuti tramite SonarExpert (sez. X.Y.Z ).

### 3.1.4 Wiimote

Il WII Remote ( wiimote ) è il controller standard della WII. Esso è dotato di una telecamera infrarossi, un accelerometro a tre assi, di quattro led blu controllabili indipendentemente e di diversi pulsanti. Al momento solo alcuni di essi sono utilizzati:

Pulsante B: viene utilizzato per caricare il colpo e sparare al robot.

Pulsante A: arresto di emergenza del robot.

Pulsante -: passa al controllo manuale del robot e interrompe il gioco.

Pulsante +: passa al gioco.

Tasti direzionali: permettono il controllo del robot in modalità manuale.

Tramite la telecamera è possibile individuare i due led infrarossi presenti sul robot e quindi sapere quando esso è puntato oppure no. Le coordinate dei due led sono fornite come punti su un "immagine" di 1024x768px.


Figura 2: Il Wii Remote (WiiMote)

I led invece sono utilizzati per indicare il punteggio del giocatore e del robot, l'inizio e la fine di una partita.

*3.1.5 Computer*

L'intero gioco viene gestito tramite un computer dotato di connettività Wi-Fi ( per connettersi a Spykee ) e bluetooth ( per il Wiimote ). E' inoltre necessario disporre di almeno una porta usb per poter collegare la scheda XBee.

*3.1.6 Marker*

Per poter essere riconosciuto dal robot il giocatore deve indossare almeno due marker colorati: uno sul torace e uno sulla schiena. Al momento il colore utilizzato è un rosa acceso in quanto facilmente riconoscibile dal robot.

## 3.2 Software

In questa sezione viene spiegato parte del codice sorgente del gioco e le librerie che sono state utilizzate per realizzarlo.

### *3.2.1 Librerie utilizzate*

*3.2.1.1 Wiiuse*

Wiiuse è una libreria scritta in C/C++ che permette di interfacciare il computer con diversi controller wiimote, tramite collegamento Bluetooth.
Essa permette di rilevare la pressione dei tasti, controllare i led blu e inoltre dispone di funzioni per ottenere i valori registrati dall'accelerometro ( permettendo cosi di conoscere l'orientamento del wiimote ) e anche di ottenere, sotto forma di coordinate XY. E' inoltre possibile ottenere una stima della distanza dalla sorgente infrarossi ( coordinata Z ).

Per rilevare un wiimote bisogna utilizzare il seguente codice:

```
wiimotes = wiiuse_init(MAX_WIIMOTES);
found = wiiuse_find(wiimotes, MAX_WIIMOTES, CONNECTION_TIME);
if (!found) {
  printf ("No wiimotes found.");
  return 0;
}
connected = wiiuse_connect(wiimotes, MAX_WIIMOTES);
if (connected)
  printf("Connected to %i wiimotes (of %i found).\n", connected, found);
else {
  printf("Failed to connect to any wiimote.\n");
return 0; }
```

```
wiiuse_motion_sensing(wiimotes[0],1);
wiiuse_set_ir(wiimotes[0], 1);
```

Mentre per rilevare gli eventi:

```
while (1) {
   if (wiiuse_poll(wiimotes, MAX_WIIMOTES)) {
      /* ... */
   }
}
wiiuse_cleanup(wiimotes, MAX_WIIMOTES);
```

Quindi il codice principale che controlla il robot e il gioco va inserito all'interno di questo ciclo while.
Per una documentazione completa delle Wiiuse si veda [5].

*3.2.1.2 Libreria Spykee.*

Questa libreria permette di controllare il robot Spykee; permette inoltre di ricevere il video della telecamera sotto forma di una sequenza di immagini jpeg con una risoluzione 320x240px con una profondità di colore di 24bpp.

Le funzioni principali sono:

- Spykee( char*username, char *password );
- void Move ( int right, int left )
- void startCamera();
- static byte *pharseMessage( byte *m, int *lm, byte **image, enum operations op )

Per una documentazione completa della libreria si veda [1].

*3.2.1.3 OpenCV*

OpenCV è una libreria open source [2] che consiste in un insieme di funzioni e classi scritte in C/C++ sviluppate e supportate da Intel per la visione artificiale. Questa libreria è orientata alla visione artificiale in tempo reale utilizzando quindi funzioni ad alte prestazione, ottimizzate soprattutto per architettura Intel.
OpenCV comprende numerose funzioni di alto livello: tecniche di calibrazione, features detection, tracciamento tramite flusso ottico, analisi sulla geometria dei contorni, identificazione del movimento, ricostruzione 3D, riconoscimento dei gesti della mano e segmentazione dell'immagine. Permette inoltre di lavorare sui colori delle immagini riconoscendo aree con un determinato colore, permettendone il tracciamento.

*3.2.1.4 libjpeg*

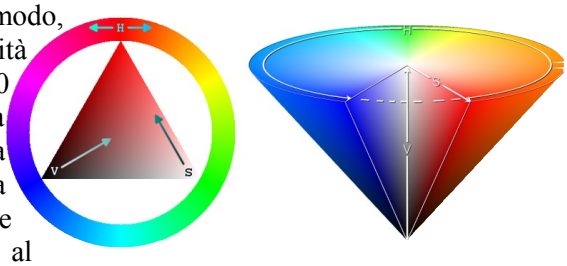
Questa è la libreria standard per la lettura e scrittura di immagini in formato jpeg. E' stato necessario utilizzarla per poter convertire il flusso di immagini in formato jpeg acquisito da Spykee e passarlo alle OpenCV come sequenza di immagini non compresse ( RGB ).

### *3.2.3 Visione*

#### *3.2.3.1 Istogramma immagine*

L'istogramma immagine è una rappresentazione della distribuzione delle diverse tonalità di colore in una immagine. Sull'asse orizzontale rappresenta le differenti tonalità di colore, mentre quello verticale vi è il numero di pixel per un particolare tono.

#### *3.2.3.2 Spazio colore HSV*




Lo spazio del colore HSV ( Hue, Saturation, Value ) è un modo, differente dal RGB, per rappresentare il colore. La tonalità ( hue ) indica il colore stesso ed ha un valore compreso tra 0 ( rosso ) e 360 ( ancora rosso ), infatti nella rappresentazione "a cono" corrisponde all'angolo lungo l'asse verticale. La saturazione corrisponde alla distanza dall'asse del cono e varia tra 0 e 1. La luminosità ( Value ) varia anch'essa tra 0 e 1 e corrisponde all'altezza del cono ( 1 in cima corrisponde al bianco, 0 al vertice inferiore al nero ).
L'utilizzo di questa rappresentazione del colore è molto comoda in quanto permette di tenere separati la tonalità di una determinata regione dell'immagine da altre sue caratteristiche che in questo caso risultano meno utili ( la saturazione e la luminosità ).
Nell'utilizzare questa rappresentazione con le OpenCV bisogna porre attenzione al fatto che l'hue varia tra 0 e 180, mentre Saturation e Value variano tra 0 e 255.

#### *3.2.3.3 Algoritmo di visione*

Il modulo che si occupa del riconoscimento visivo è VisionExpert, che composto dalle seguenti funzioni ausiliarie:

- *Costruttore*: qui sono inizializzate le variabili e i buffer necessari all'acquisizione e all'elaborazione delle immagini ( si vedano i commenti del codice per la descrizione delle singole variabili ).
- *jpeg2bmp*: si occupa di convertire un immagine in formato jpeg, rappresentata come un vettore di byte, in un immagine non compressa in formato bmp anch'essa rappresentata come un array di byte ( in cui ogni tripletta di byte indica il contenuto di rosso, verde e blu di un singolo pixel).
- *CreateMatchingHist* si occupa di creare l'istogramma del colore di un immagine campione fornita dall'utente. Questo istogramma verrà poi utilizzato dall'algoritmo per identificare il giocatore.

L'algoritmo vero e proprio è contenuto nella funzione RunDuty ( eseguita in loop dall'MRT ); i passi principali dell'algoritmo sono:

1. *righe 109-118:* qui l'immagine viene acquisita, tramite la funzione Spykee::pharseMessage, che restituisce l'immagine in jpeg come array di byte e che memorizza in imgLen la dimensione del vettore.
2. *riga 123:* l'immagine viene convertita in formato bmp. Questo è stato necessario dato che le openCV non permettono la decodifica di un immagine jpeg già caricata in memoria.
3. *riga 126:* il vettore dell'immagine bmp viene "incollato" sul campo imageData di m_image. In questo modo alle openCv sembrerà di avere caricato e decodificato un immagine jpeg.
4. *righe 128-130*: l'immagine viene convertita nello spazio di colore hsv. Viene anche effettuata una sogliatura della saturazione e della luminosità dell'immagine per eliminare le zone troppo chiare e troppo scure che disturberebbero il riconoscimento del colore. Il risultato di queste operazioni è un immagine in bianco e nero ( m_mask ), in cui le zone nere sono le parti di immagine che non soddisfano i valori di soglia ( quindi o troppo chiare o troppo scure ).
5. *riga 132:* il canale hue viene separato dall'immagine.

6. *righe 134-138:* viene calcolato il backproject utilizzando l'istogramma del colore dell'immagine di riferimento ( calcolato con CreateMatchingHist ). Il backproject è una rappresentazione probabilistica dell'immagine originale che mostra per ogni pixel la probabilità di appartenere o meno alla gamma di colori contenuti nell'istogramma. Per una migliore precisione il backproject viene poi messo in and con la maschera e sogliato per ottenere solo la zona in cui il colore è più "evidente".

7. *riga 145:* vengono conteggiati i pixel bianchi nel backproject.

8. *righe 165-184:* se il numero di pixel bianchi è superiore a una certa soglia, i pixel bianchi vengono poi sommati per colonne e inseriti in un vettore. Alla righe 174-175 viene calcolata la posizione sull'asse delle ascisse dove sono presenti più pixel bianchi e conoscendo l'apertura della telecamera ( 46° ) viene convertito nella posizione angolare del giocatore rispetto alla direzione del robot. In questo caso viene inviato a MrBrian un messaggio che indica che la posizione del giocatore.

9. *righe 185-194:* nel caso in cui il giocatore esca dal campo visivo non verrà inviato a MrBrian il messaggio corrispondente a questo evento.

Inizialmente si era pensato di utilizzare l'algoritmo camschift per il tracking del giocatore. Tuttavia questo algoritmo funziona bene per tracciare il giocatore senza che questo esca dal campo visivo. Se questo dovesse avvenire c'è il rischio che venga "agganciato" un oggetto sullo sfondo che ha un colore simile a quello del marker del giocatore. Quindi alla fine si è deciso di utilizzare l'algoritmo precedente per ottenere la posizione del giocatore,

### 3.2.4 MRT

Il software del gioco è sviluppato intorno al DCDT, un framework che permette di eseguire in parallelo diversi moduli, in questo classi derivate da una classe base chiamata Expert. I vari moduli possono comunicare tra loro, sia che siano in esecuzione su una stessa macchina o in una rete, tramite lo scambio di messaggi in formato XML.

I moduli principali sono:

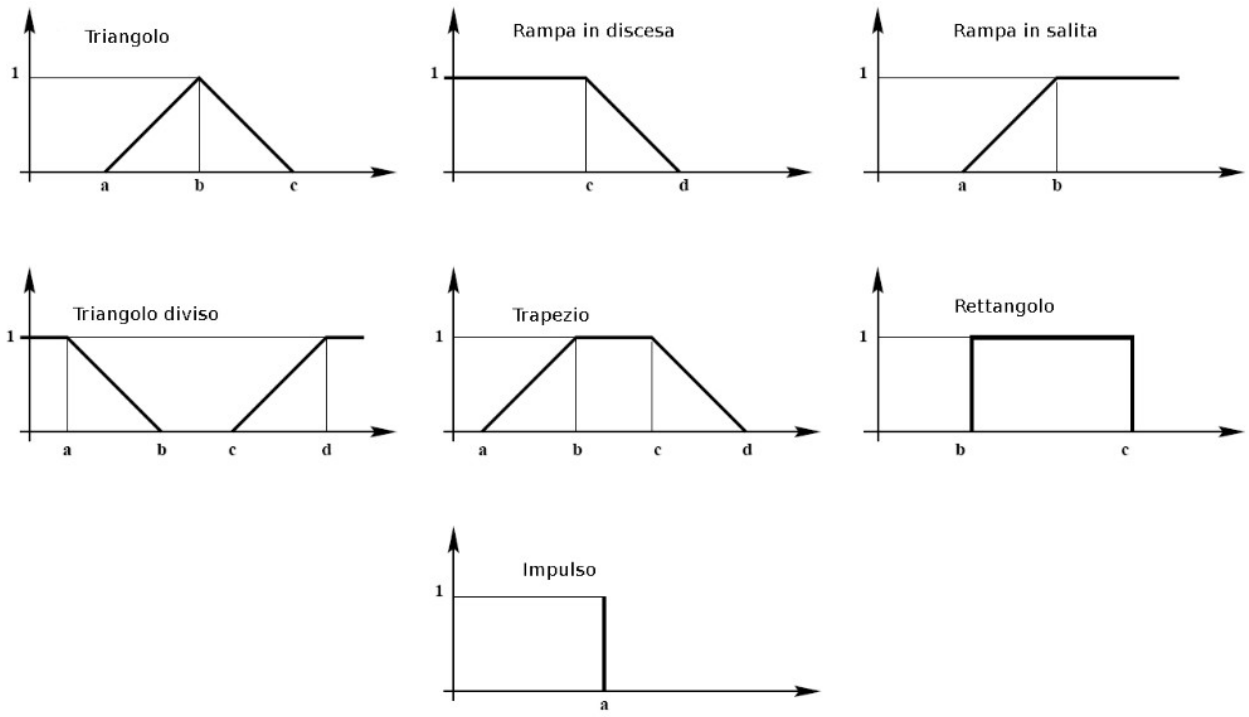
- VisionExpert: si occupa di gestire la telecamera di Spykee e contiene l'algoritmo di visione.
  - SonarExpert: si occupa di ricevere e interpretare i dati dei sonar. Permette inoltre di controllare i led.
  - MotorExpert: invia i comandi di movimento al robot.
  - WiimoteExpert: contiene il codice principale del gioco e si occupa di gestire la comunicazione con il Wiimote.
  - BrianExpert: si occupa di interpretare i dati che gli arrivano dagli altri moduli, generare le regole fuzzy e inviare i comandi agli altri moduli.

*Logica fuzzy e regole*

I comportamenti del robot sono decisi utilizzando la logica fuzzy: un tipo di logica in cui le proposizioni non assumono solo il valore vero o falso, ma hanno un grado di verità compreso tra 0 (falso) e 1 (vero). Il software che si occupa di gestire la logica fuzzy è Mr. Brian sviluppato dal Politecnico di Milano per il progetto MRT ( Milano RoboCup Team ).

Per prima cosa è necessario definire i valori in ingresso e la loro forma come variabili fuzzy. Queste vano definite nel file shape_ctof.txt ( Appendice B ).

Le forme principali utilizzate sono quelle rappresentate in figura.




Una volta definite le variabili bisogna definire i predicati ( contenuti nel file Predicate.ini ) che assumono valori di verità compresi tra 0 e 1. Essi possono essere definiti a partire da una o più variabili fuzzy o anche da altri predicati, combinati utilizzando gli operatori logici AND, OR e NOT.

In seguito i moduli CANDO e WANT si occupano di decidere quali comportamenti vanno attivati valutando i predicati. In particolare il modulo CANDO si occupa di decidere se un comportamento può venire attivato ( ad esempio se il robot rileva che dietro di se ha spazio libero può indietreggiare ), mentre il modulo WANT si occupa di stabile se un comportamento deve essere attivato ( se ad esempio il giocatore viene visto dal robot, questo vuole indietreggiare ).
I vari comportamenti sono definiti nel file behaviour.ini e nei rispettivi file .rul.

I comportamenti possono inoltre impostare i valori delle variabili fuzzy in uscita, che sono tutte definite come impulsi nel file s_shape.txt. Queste vengono poi convertite in valori numerici e inviate dal modulo che gestisce Mr. Brian agli altri moduli interessati.

# Capitolo 4

*4.1 Analisi qualitativo dell'algoritmo di visione*

L'algoritmo utilizzato è in grado di distinguere un giocatore che indossa un marker di un colore che “risalta” rispetto allo sfondo; in questa situazione il giocatore risulta facilmente distinguibile ed è quindi possibile determinarne con buona approssimazione l'angolo ( e di conseguenza la posizione ) del giocatore rispetto al robot.

Se nell'ambiente di gioco sono presenti oggetti di un colore uguale ( o anche molto simili ) al marker, l'algoritmo non è in grado di distinguere il giocatore ( che indossa il marker ) dall'oggetto esterno. Di conseguenza si possono verificare due situazioni:

1. Se l'oggetto esterno entra nel campo visivo del robot, verrà scambiato per il giocatore e il robot agirà di conseguenza.
2. Se sia l'oggetto esterno che il marker si trovano nel campo visivo del robot, non si può sapere a priori quale dei due verrà identificato come giocatore.

Eseguendo delle prove si è riscontrato che un buon marker deve avere le seguenti caratteristiche:

- Un colore che risalta rispetto all'ambiente circostante
- Non deve essere composto d un materiale che rifletta eccessivamente la luce, in quanto esso apparirebbe troppo chiaro alla telecamera e ignorato ( Fig. 5 )
- Deve essere il più possibile una figura piena di concavità in modo da risultare un unico oggetto uniforme agli occhi del robot.

Ad esempio una maglietta colorata, o un grosso rettangolo di carta ( sempre colorata ) incollata sul petto del giocatore rappresentano dei buoni marker ( il giubbino utilizzato in figura non è un ottimo marker ).

Una limitazione imposta invece dalla telecamera utilizzata è quella di giocare in una stanza ben illuminata; questo problema potrebbe tuttavia essere risolto utilizzando una diversa telecamera.

E' inoltre possibile che in alcune situazioni sia presente un "rumore di fondo" che potrebbe provocare false letture nell'algoritmo ( Fig. 6 )

Fig. 5: immagine della telecamera ( sopra ) e relativo backprject

Fig 6: Rumore di fondo ( falsi positivi )

*4.2 Miglioramenti*

Ci sono diverse alternative per rendere l'algoritmo più robusto:

- Utilizzare un marker di una forma ben definita e di due colori diversi e modificare l'algoritmo in modo che identifichi un oggetto di quella particolare forma e che abbia esattamente quei due colori.
- Effettuare, prima di iniziare una partita, una fase di "training" del sistema di visione, durante la quale si "insegna" al robot a riconoscere il marker del giocatore utilizzando tecniche di template-matching.
- Un ulteriore miglioramento che si potrebbe fare è insegnare al robot a riconoscere la forma di una persona ( in particolare il viso ) in corrispondenza del marker, di modo da ridurre il rischio di confusione con oggetti esterni al gioco.

**Riferimenti**

[1] Antonio Micali, Documentazione libreria Spykee ( AirWiki – Spykee )

[2] Sito web OpenCV: http://sourceforge.net/projects/opencvlibrary/

[3] Documentazione MRT ( AirWiki )

[4] Documentazione MrBrian ( AirWiki )

[5] Wiiuse: http://www.wiiuse.net