Manuale d'uso degli script Pollicino^{TM}

Marco Ghidinelli

February 8, 1999

Abstract

Questo documento descrive il funzionamento del sistema di stima della posizione "Pollicino^{TM}" realizzato da Davide Grana basato sull'istruzione di una rete neurale con immagini acquisite su uno specchio conico.

1 Implementazione generale.

La rete neurale viene istruita partendo da un set di immagini codificate in vettori monodimensionali contenenti i dati delle "acquisizioni" statiche eseguite da una telecamera che riprende lo specchio conico nell'ambiente dove si vuole stimare la posizione. Una parte di queste immagini viene utilizzata per istruire la rete neurale, e quando tale rete risulta istruita, ne viene testata l'accuratezza facendole stimare la posizione nel caso di altre immagini acquisite.

Gli script per il funzionamento del sistema Pollicino sono fondamentalmente delle interfaccie verso la rete neurale `SNNS`. Tali script trasformano le immagini provenienti dallo specchio conico in pattern per istruire tale rete neurale, eseguono il processo di training e di testing della rete e permettono una valutazione grafica dei risultati ottenuti. Per ogni ambiente di acquisizione si possono predisporre più test istruendo con più o meno informazioni la rete neurale. Tutti gli script vengono implementati su una macchina UNIX e richiedono di funzionare in determinate directory e necessitano di numerosi file di configurazione.

Il sistema specchio conico-telecamera viene portato in un ambiente dove si vogliono eseguire delle stime e vengono acquisite tutte le immmagini considerate necessarie. Tali immagini possono essere prese da posizioni appartenenti ad una griglia cartesiana o da qualsiasi insieme di punti della stanza.

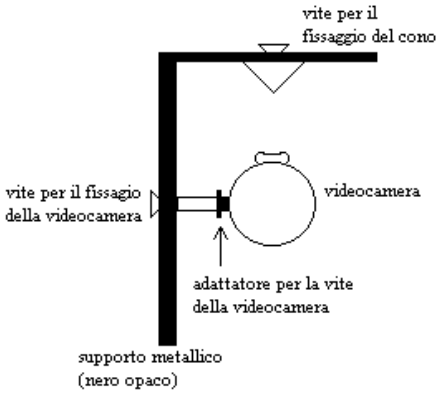
Per una buona comprensione del funzionamento del software è necessario leggere completamente questo documento, in quanto nei primi tre capitoli vengono spiegate i modi di funzionamento e le modalità di configurazione. I più ansiosi possono comunque iniziare a leggere dal capitolo quattro per poi tornare a leggere i primi capitoli in seguito.

2 Hardware

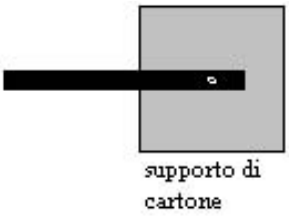
E' qui descritta l'implementazione hardware del sistema POLLICINO. E' dapprima presentato il supporto che permette di mantenere in asse tra di loro lo specchio conico e la telecamera; seguono le caratteristiche dello specchio conico e della telecamera utilizzati.

2.1 Struttura per l'acquisizione

Il sistema POLLICINO prevede l'utilizzo di una videocamera ed uno specchio conico in asse tra di loro. POLLICINO è stato realizzato con un supporto di tipo metallico, come mostrato in figura:



Esso è costituito da un supporto metallico di colore nero opaco a forma di L. L'opacità ha la funzione di evitare riflessioni che agiscono da disturbo sull'immagine acquisita. Sul lato più corto è fissato il cono attraverso una vite; sul lato più lungo è fissata la telecamera. Si è reso necessario l'utilizzo di un adattatore per la vite da inserire nella telecamera in modo da fissarla. La vista dall'alto si presenta nel modo indicato dalla figura seguente.



Un supporto quadrato di colore grigio opaco è inserito tra il supporto e la base del cono, per evitare che fonti di luce situate sul soffitto negli ambienti di utilizzo possano mandare in saturazione la telecamera, che acquisirebbe in questo modo una immagine con la presenza di grandi macchie bianche, corrispondenti alle zone direttamente colpite dalla luce.

Sul lato lungo del supporto sono presenti fori di diverse dimensioni e ampiezze per permettere attraverso delle viti il fissaggio del sistema al robot mobile che lo utilizza.

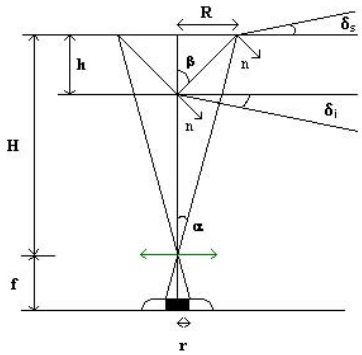
Il pregio di una realizzazione di questo tipo è la robustezza del supporto metallico che permette di avere telecamera e cono in asse tra di loro anche quando la struttura è sollecitata da vibrazioni.

Un apparente svantaggio è costituito dal fatto che il supporto occlude parzialmente la vista dell'ambiente circostante, ma in realtà questo avviene per pochissimi settori angolari e non costituisce un problema ai fini dell'autolocalizzazione, come mostrato nel cap.6, alla luce dei risultati sperimentali.

2.1.1 Lo specchio conico

In questo paragrafo si discutono le caratteristiche geometriche della implementazione proposta, seguite dalla presentazione del modello utilizzato per descrivere la rugosità della superficie.

I parametri che determinano la geometria del sistema costituito dallo specchio conico e dalla videocamera sono mostrati nella figura alla pagina seguente.



con:

α semiampiezza dell'angolo sotto il quale il cono è visto dalla telecamera

β semiampiezza dell'angolo alla sommità del cono

δ_i ampiezza dell'angolo compreso tra la base del cono ed il raggio limite inferiore del campo di visione corrispondente al vertice

δ_s ampiezza dell'angolo compreso tra la base del cono ed il raggio limite del campo di visione corrispondente alla base del cono

h altezza del cono

H distanza fra la base del cono e l'obiettivo della telecamera

R raggio della base del cono

f distanza focale dell'obiettivo

r raggio massimo della zona sensibile del sensore CCD.

Le principali relazioni che intercorrono tra i parametri considerati sono le seguenti:
Angolo di visione:

$$tan(\alpha) = \frac{r}{f}$$

Pendenza dello specchio:

$$tan(\beta) = tan(\alpha) * \frac{H}{h} = \frac{R}{h}$$

Angolo limite inferiore:

$$\delta_i = \frac{\pi}{2} - 2\beta$$

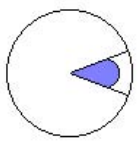
Angolo limite superiore:

$$\delta_s = \frac{\pi}{2} + \alpha - 2\beta$$

$$\delta_s - \delta_i = \alpha$$

Si suppone che la geometria del sistema permetta di utilizzare al massimo il sensore della telecamera, ovvero che il cono occupi tutta l'immagine. L'ampiezza dell'angolo α allora dipende solamente dalle dimensioni del sensore CCD e dalla focale della lente obiettivo. Si noti inoltre che α corrisponde all'angolo di visione $\delta_s - \delta_i$. Una volta fissati α e β in base alle caratteristiche della telecamera utilizzata e al campo di visione desiderato risultano noti i rapporti $\frac{H}{h}$e $\frac{R}{h}$tramite i quali è possibile scegliere le dimensioni del cono e la distanza tra cono e telecamera.

Lo specchio conico permette, tramite l'acquisizione di una sola immagine, la visione su tutto lo spazio circostante. L'immagine è costituita da settori angolari, che corrispondono agli elementi verticali che rientrano nel campo di visione; gli elementi con bassa componente verticale tendono ad essere eliminati nell'immagine. L'immagine di un oggetto parallelo all'asse del cono, a distanza opportuna in modo da intercettare il campo di visione solo per una parte di quest'ultimo, genera un settore suddiviso in due zone delimitate da una curva corrispondente al lato orizzontale. La curva è inviluppata dal settore e tende a chiudersi nel centro del cono, come si può vedere nella figura seguente:



Tanto più l'oggetto si allontana dal cono, tanto più il settore corrispondente si stringe e la curva risulta sempre più schiacciata.

La presenza di rugosità sulla superficie dello specchio produce effetti di diffusione della luce riflessa che privilegiano la percezione delle caratteristiche che occupano tutto il raggio del settore rispetto a quelle che lo occupano solo in parte. Ciò rende più uniforme il colore del settore e attenua gli effetti degli elementi non verticali intercettati dal campo di visione.

Il modello per la riflettività prevede tre componenti per la luce riflessa dalla superficie:

- Diffusione Lambertiana
- Riflessione speculare (specular spike)
- Riflessione speculare diffusa (specular lobe).

La componente Lambertiana è distribuita con la medesima intensità in tutte le direzioni ammissibili. E' imputabile a processi di diffusione dovuti a disomogeneità del materiale costituente gli strati superficiali [SPA, 68]. Nel caso dello specchio conico la componente Lambertiana può essere trascurata (essendo costante per ogni direzione, essa determina principalmente un effetto di diminuzione del contrasto).

La componente di riflessione speculare è distribuita come un picco di radianza orientato nella direzione teorica di riflessione della luce incidente secondo la normale alla superficie nel punto di incidenza considerato. Il picco è molto accentuato per superfici molto lisce.

Il lobo di riflessione speculare diffusa è presente, insieme alla componente di riflessione speculare, in uno stretto intervallo di rugosità. L'intensità e l'angolo solido occupato dal lobo di riflessione diffusa diminuiscono all'aumentare delle componente di riflessione speculare, e aumentano in corrispondenza dell'aumento delle dimensioni delle irregolarità superficiali. L'asse del lobo risulta orientato secondo una direzione diversa da quella teorica di riflessione, deviando verso la superficie.

Una superficie molto liscia si comporta come uno specchio perfetto in quanto il lobo speculare diffuso è trascurabile rispetto al picco speculare.

2.1.2 La videocamera

Sono qui di seguito riportate le specifiche della telecamera utilizzata fornite dal costruttore e la prova di ripetibilità effettuata con i relativi risultati.

E' stata utilizzata la telecamera Color QuickCam per PC prodotta dalla Connectix Corporation. Le specifiche tecniche fornite dal costruttore sono le seguenti:

- Fino a 640*480 pixel di risoluzione
- Fino a 30 fps (fotogrammi per secondo)
- Alimentata dalla porta della tastiera, consumo inferiore a 2 Watt
- Angolo di visione circa 48ř (equivalente ad una lente da 50 mm su una macchina fotografica da 35 mm)
- Fuoco regolabile da 35 mm all'infinito
- Diaframma $f = 1,6$

La connessione al calcolatore avviene secondo le modalità specificate nel manuale annesso alla videocamera.

Si è voluta testare la ripetibilità della telecamera nella fase di acquisizione. L'obiettivo è quello di verificare quanto siano simili tra loro i pattern ottenuti da immagini apprese a pochi istanti l'una dall'altra nello stesso punto e con le stesse condizioni di illuminazione. A questo scopo sono state apprese dieci immagini nella stessa posizione senza variare l'ambiente circostante per quanto riguarda gli oggetti presenti e l'intensità delle fonti di luce. Si sono confrontati, per ogni componente cromatica, i vettori forniti in uscita dalla pre-elaborazione dell'immagine. I risultati sono visibili nei tre grafici riportati alla pagina seguente che riportano per ogni componente cromatica i primi 8 elementi del vettore ottenuto dalla pre-elaborazione. I valori sono stati ottenuti per interpolazione.

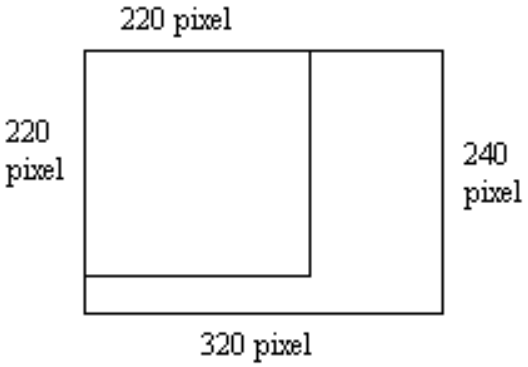
Come si può notare, i risultati sono molto buoni: il vettore dato in uscita è praticamente sempre lo stesso, di volta in volta cambiano solo alcune componenti, e il discostamento massimo risulta essere quello mostrato nella tabella riportata alla pagina seguente (i dati sono normalizzati a 255).

3 Software

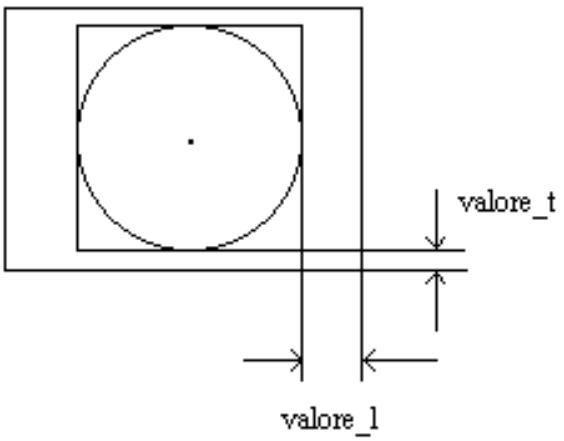
3.1 Setup della telecamera

Le immagini acquisite dalla telecamera hanno una risoluzione di 240*320 pixel. Per fotografare opportunamente il cono si è scelta una finestra quadrata di dimensioni 220*220 pixel all'interno di tale immagine.

Tramite `qcread` è possibile acquisire l'immagine: lanciando l'eseguibile `qcread -x 220 -y 220 -s 1` viene acquisito un frame in scala 1:1 di larghezza 220 pixel e altezza 220 pixel che si posiziona rispetto al frame completo di dimensioni 240*320 nel modo illustrato alla pagina seguente.



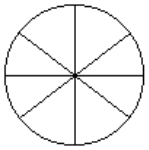
La prima operazione da effettuare durante il setup è quella di centrare, per lo meno in prima approssimazione, l'immagine del cono. Questo è possibile tramite le opzioni `-l` e `-t` di `qcread`: `qcread -x 220 -y 220 -s 1 -l valore_l -t valore_t` preleva dall'immagine originale 240*320 pixel una immagine quadrata di dimensioni 220*220 pixel posizionata come indicato in figura:



Per centrare opportunamente l'immagine del cono è necessario effettuare delle prove variando di volta in volta `valore_l` e `valore_t` attorno al loro valore 'ideale', che è `valore_l` 50 `valore_t` 10, ovvero il valore che i due parametri dovrebbero assumere se cono e telecamera fossero perfettamente in asse tra di loro.

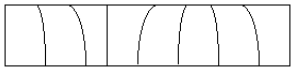
Per vedere sul monitor del calcolatore tali immagini si può utilizzare ad esempio il programma xv, cosicché la riga di comando è ora `qcread -x 220 -y 220 -s 1 -l valore_l -t valore_t | xv -&`.

Dopo avere centrato approssimativamente l'immagine del cono si cerca ora un allineamento asse cono-asse telecamera più preciso. Questo avviene osservando il risultato della trasformazione da sistema di riferimento cartesiano a polare. Se infatti telecamera e cono sono perfettamente in asse tra di loro il frame ottenuto tramite `qcread` ha l'aspetto della figura seguente:



e il risultato del cambiamento del sistema di riferimento da cartesiano a polare produce l'immagine seguente:

Se invece telecamera e cono non sono perfettamente in asse tra di loro, in uscita è prodotta una immagine del tipo seguente:



A questo punto si cerca di ottenere in uscita una immagine il più possibile vicina a quella 'ideale' di perfetto allineamento. Vengono quindi variati i parametri `valore_l` e `valore_t` ed effettuata la trasformazione cartesiano polare tramite il programma `quick`. La riga di comando ora è `qcread -x 220 -y 220 -s 1 -b 160 -l valore_l -t valore_t | quick | xv *.pgm &` dove `-b 160` è l'opzione per la luminosità, da settare necessariamente se l'ambiente di acquisizione risulta poco illuminato. Questo permette di distinguere bene i vari settori angolari tra di loro, soprattutto una volta cambiato il sistema di riferimento.

`xv *.pgm` permette di visualizzare sul monitor i file `blu.pgm`, `rosso.pgm`, `verde.pgm`, aventi forma simile alla fig. 6.13, contenenti le tre componenti cromatiche dell'immagine in coordinate polari.

Una volta settati i parametri è possibile effettuare l'acquisizione di immagini.

3.2 Acquisizione dell'immagine

È effettuata lanciando da riga di comando `qcread -x 220 -y 220 -s 1 -b 160 -l valore_l -t valore_t | quick > nomeimmagine.`

In questo modo il file nomeimmagine contiene ora 1080 valori, compresi tra 0 e 255, corrispondenti alle 360 componenti angolari rosse, 360 verdi, 360 blu.

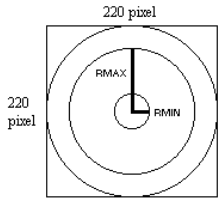
L'acquisizione può essere fatta anche lanciando `testall`, script nel linguaggio di comandi della shell del sistema operativo Linux che riassume la linea di comando con i parametri opportunamente fissati:

```
#!bin/sh
qcread -x 220 -y 220 -s 1 -b 160 -l 50 -t 10 | quick
```

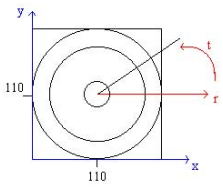
cosicchè l'acquisizione si effettua ora con `testall > nomeimmagine.`

3.3 Elaborazione dell'immagine

Questa fase ha lo scopo di estrarre dal frame acquisito dalla telecamera 360 componenti angolari per ogni canale cromatico ed avviene essenzialmente attraverso una trasformazione cartesiano-polare. Per ogni colore della terna R, G, B l'immagine del cono viene innanzitutto mascherata in modo tale che solo una corona circolare venga considerata nei passi successivi, scartando così la parte centrale, dove è difficile distinguere bene i vari settori tra di loro, e la parte ai bordi che presenta rumore. La corona circolare è presa come in figura:



RMIN e RMAX sono rispettivamente i raggi interno ed esterno della corona, fissati di default in 60 e 90 pixel rispettivamente. Sulla corona circolare così ottenuta è applicata una trasformazione del sistema di riferimento da cartesiano a polare. Gli assi cartesiani x e y e il riferimento polare r e t sono posizionati rispetto alla corona circolare nel modo indicato dalla figura seguente:

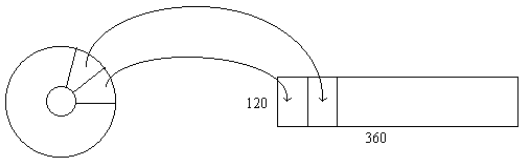


La corona circolare viene così mappata in una matrice di dimensioni 120*360 per ogni componente cromatica, indipendentemente dai valori di RMIN e RMAX. Per ogni pixel di coordinate (r,t) in questa matrice sono calcolate le coordinate (x,y) del pixel nella immagine originale del cono che gli corrisponde secondo le relazioni:

$$x = 110 + [\frac{RMAX - RMIN}{120} * r + RMIN] * cos(\frac{2\pi * t}{360})$$

$$y = 110 + [\frac{RMAX - RMIN}{120} * r + RMIN] * sin(\frac{2\pi * t}{360})$$

La mappatura avviene perciò nel modo indicato dalla figura seguente:



Per ogni canale cromatico si ha ora una matrice 120*360.

Lo scopo è ottenere per ogni componente di colore 360 valori, ognuno rappresentativo di un settore angolare di 1ř.

Per ogni settore angolare si è scelto di considerare la media di tutti i pixel che vi appartengono. In pratica ciò significa che per ognuna delle 360 colonne della matrice ottenuta viene calcolata la media dei 120 valori che la compongono.

Per ogni colore sono ora ottenuti 360 elementi, compresi tra 0 e 255, rappresentativi di 360 settori angolari.

In appendice è riportato il listato del programma quick.c realizzato in linguaggio C++ che effettua la elaborazione dell'immagine appena descritta. Si noti il fatto che vengono fornite dalla telecamera, nell'ordine, la componente blu, rossa, e verde, mentre la trasformazione in coordinate polari fornisce nel consueto ordine le componenti R, G, B.

3.4 Stima della posizione

Sono stati implementati algoritmi di stima della posizione mediante rete neurale e metodi statistici su PC con sistema operativo Linux versione 2.1.43. La scelta del sistema operativo è stata dettata dalla possibilità di utilizzare un simulatore di rete neurale.

3.4.1 Implementazione della rete neurale

La definizione, le fasi di apprendimento e di test della rete neurale sono stati effettuati tramite il simulatore di rete neurale dell'Università di Stoccarda, SNNSStuttgart Neural Network simulator, versione 4.1, 1995, ottenuto gratuitamente via anonymous ftp all'indirizzo ftp://ftp.informatik.uni-stuttgart.de (129.69.211.2) assieme al relativo manuale[SNNS, 95].

Mediante il simulatore è stata effettuata la definizione degli strati e dei linktra le varie celle della rete, seguita dalle fasi di learning e di test.

Sono stati implementati degli script nel linguaggio dei comandi della shell del sistema operativo per creare i pattern e i file utilizzati dal simulatore e per effettuare la stima della posizione con risultati di tipo grafico in uscita.

Per convenzione d'ora in avanti i comandi editati dall'utente nelle varie fasi saranno indicati con il carattere tipografico `Courier`.

3.4.2 Gli eseguibili.

Gli script Pollicino per funzionare correttamente devono trovare un albero di sottodirectory ben definito costruito secondo regole fissate. Gli script devono inoltre essere eseguiti dalla directory CONE e questo necessita quindi una variazione delle variabili d'ambiente che devono contenere il percorso degli eseguibili che si trovano in CONE/bin. Per settare questa variabile si deve identificare la shell utilizzata e poi eseguire:

```
bash-sh:  export PATH=$PATH:/dove_si_trova_CONE/CONE/bin

csh:      setenv PATH $PATH:/dove_si_trova_CONE/CONE/bin
```

E' necessario inoltre avere installato sulla macchina su cui gli script devono funzionare i programmi `bash`, `awk`, `ghostscript`, `gnuplot`, che sono programmi free prodotti dalla GNU.

3.4.3 La rete neurale.

Come si è già detto gli script servono fondamentalmente come interfaccia per la rete neurale `SNNS`, che deve essere installata nel sistema. La versione richiesta dagli script è la 4.1 e può essere ottenuta via ftp al sito `ftp.informatik.uni-stuttgart.de` nella directory `SNNS`. Anche la directory degli eseguibili della rete neurale devono essere inseriti nel `PATH` per poter essere eseguiti dagli script.

3.5 Gli ambienti.

Nella directory `CONE` sono presenti delle directory con una estensione `.env`: queste directory rappresentano tutti gli ambienti di test degli script e contengono:

`data` tutti i campioni provenienti dallo specchio conico sotto forma di componenti di rosso, blu, verde, o tutte e tre le componenti.

`maps` tutte le informazioni relative alla loro posizione posizione,

`test` tutti i test preparati.

Il contenuto di queste directory verrà analizzato ora singolarmente.

3.5.1 Directory `data`.

In questa directory sono contenute tutte le immagini acquisite in un dato ambiente. Il loro nome identifica l'ordine con il quale le immagini vengono acquisite mantre la loro estensione rappresenta quali parti dello spettro *RGB* di una immagine sono contenute nel campione.

Queste immagini rappresentano il vettore di 360 gradi elaborato dall'immagine ricevuta dalla telecamera puntata sullo specchio conico e contengono una lista di 360 o 1080 elementi a seconda che rappresentino una sola componente di colore o tutte e tre. I valori dei vettori qui contenuti rappresentano tutti i 360 valori scalati di uno stesso fattore comune a tutte le altre immagini acquisite per permettere di mantenerne la dinamica tra -1 e 1. Nel caso di un vettore tricromatico le tre componenti sono inserite una dopo l'altra.

3.5.2 Directory `maps`.

Questa directory contiene le mappe di immagini che devono essere utilizzate per testare e/o istruire la rete neurale: esse sono formate da una sequenza di righe contenenti il nome dell'immagine (contenuta nella directory `data`) seguita dalle coordinate cartesiane del punto dove tale immagine è stata acquisita. Queste mappe possono essere utilizzate sia in fase di learn che in fase di test della rete.

3.5.3 Directory `test`.

Questa directory contiene delle sottodirectory che definiscono i vari tipi di test preparati per un determinato ambiente. Un test è costituito dalla definizione delle mappe utilizzate per addestrare e testare la rete, dal numero di iterazioni usate per l'addestramento e dalla rete neurale utilizzata.

Il nome del test viene spesso utilizzato come identificativo del tipo di test: le prime lettere corrispondono alla densità con cui si sono acquisiti i campioni, cioè bassa, media o alta; mentre le lettere finali identificano su quali componenti di colore ci si è basati per il test: `r_g_b` corrisponde a tutte le componenti di colore, `g` corrisponde solo al verde.

In una sottodirectory di test vi sono numerosi file che devono essere analizzati separatamente:

`batch.cfb` E' lo file batch fondamentale che viene lanciato dagli script: viene mandato al programma `snnsbat` che esegue il suo contenuto. Al suo interno vi sono informazioni fondamentali su quello che deve eseguire il test:

`NetworkFile` Determina la rete neurale che deve essere usata per lo svolgimento di un test.

`InitFunction` Determina la funzione di inizializzazione dei pesi dei nodi della rete neurale

`NoOfInitParam` Definisce il numero di parametri da passare alla funzione di inizializzazione dei pesi.

`InitParam` Definisce i parametri (che devono essere in numero pari a `NoOfInitParam` definito sopra) da passare alla funzione di inizializzazione dei pesi.

`LearnPatternFile` Definisce quale pattern utilizzare per addestrare la rete neurale: questo deve indicare il file learn.pat nella sottodirectory di test corrente.

`NoOfLearnParam` Determina il numero di parametri da dare alla funzione di addestramento.

`LearnParam` Definisce i parametri da dare alla funzione di addestramento.

`MaxLearnCycles` Definisce il numero massimo di cicli per addestrare la rete.

`MaxErrorToStop` Definisce l'errore che si vuole raggiungere in fase di addestramento.

`Shuffle` Determina se il pattern di learn deve essere mescolato per evitare che la correlazione tra i vari campioni influenzi l'apprendimento.

`TestPatternFile` Definisce quale pattern utilizzare per testare la rete già addestrata.

`ResultFile` File contenente i risultati del test: questo file è temporaneo e viene eliminato dagli script quando la procedura è terminata.

`ResultMinMaxPattern`

`ResultIncludeInput` Determina se anche il learn pattern deve essere testato dalla rete.

`ResultIncludeOutput` Determina se il file di output deve contenere anche i pattern di learn oltre a quelli di test.

`TrainedNetworkFile` Definisce il nome della rete neurale addestrata dalla procedura.

`learn.def/test.def` Indica quali sono le mappe che vengono utilizzate per testare e addestrare la rete. Gli script provvedono a creare i pattern di learn e di test a seconda di quanto viene definito in questi file.

`learn.map/test.map` Questi file contengono la copia della mappa (della directory maps) che viene utilizzata nel test e nell'addestramento della rete. Queste mappe sono generate automaticamente dagli script

`learn.pat/test.pat` Sono file temporanei contenenti i pattern effettivi che vanno ad istruire la rete. Normalmente vongono cancellati dopo essere stati utilizzati, ma possono essere non cancellati in fase di debug.

`gnuplot.ps` Questo è un file postscript creato dagli script e permette di valutare visivamente l'efficacia di un test: le losanghe rappresentano la posizione corretta mentre i segmenti rappresentano l'errore che viene commesso dalla rete neurale. Questo file non è sempre presente nella directory test e viene sempre sovrascritto quando il test viene eseguito.

4 L'uso degli script.

Gli script funzionano prendendo da tutti i campioni disponibili per un determinato ambiente due sottoinsiemi: uno viene utilizzato per addestrare la rete neurale, l'altro viene usato per testarla. La generazione dei pattern è guidata da quanto dettato nei vari file di configurazione visti sopra. Gli script fondamentali sono pochi e permettono di testare il funzionamento della rete in varie modalità di funzionamento:

`make1test nome_env nome_test` Permette di testare il funzionamento della rete in modo normale senza inserire rotazioni o occlusioni. il risoltato del test viene visualizzato a video e salvato nella directory relativa al test eseguito. Nella directory del test vi è inoltre il file `gnuplot.ps` che rappresenta graficamente il risoltato del test.

`make1testocc nome_env nome_test inizio fine valore` Esegue il test creando un'occlusione fissa in tutti i vettori di acquisizione. `Inizio` e `fine` indicano i settori che devono essere occlusi mentre `valore` indica ???

`make1testoccrnd nome_env nome_test ampiezza num_set valore` Esegue una occlusione casuale dei vettori di acquisizione.

`make1testrot nome_env nome_test rotmin rotmax` Esegue delle rotazioni dei vettori di acquisizione comprese tra rotmin e rotmax.

Vi è inoltre uno script che premette di testare una rete precedentemente addetrata in "real time": per ogni nuova immagine acquisita, viene stimata immediatamente la posizione del robot: si accede a questa modalità modiante il programma `realtimetest`.