UNIVERSITÀ DI BRESCIA
FACOLTÀ DI INGEGNERIA
Dipartimento di Ingegneria dell'Informazione

Laboratorio di Robotica Avanzata
Advanced Robotics Laboratory

Corso di Robotica
(Prof. Riccardo Cassinis)

Sistema di visione Mobolab

Elaborato di esame di:

Fabio Filisetti, Michael Caldera

Consegnato il: **09 Luglio 2012**

Sommario

Il lavoro consiste nello sviluppo di un sistema software che, acquisendo in ingresso il segnale video del laboratorio Mobolab, determini la posizione del robot che in quel momento si sta muovendo, sostituisca lo sfondo dell'area di lavoro con un'immagine scelta dall'utente e invii il risultato dell'elaborazione in streaming web.

1. Introduzione

Il laboratorio di robotica mobile Mobolab è costituito da un piano di 153x175 cm di dimensione, di colore bianco, sul quale si possono muovere due robot, uno alla volta per quanto riguarda il seguente software.

Perpendicolarmente al piano, ad un'altezza di circa 170 cm da esso, è posta una webcam che riprende l'intera area.

L'illuminazione è fornita da quattro lampade alogene.

1.1. Specifiche hardware

Il sistema è costituito da una telecamera Philips SPC1000 NC, collegata ad un server con CPU Intel(R) Core(TM) i3 CPU 550 @ 3.2 GHz.

I robot, facenti parte della serie Lego MINDSTORMS, sono nominati Qui e Quo.

1.2. Specifiche software

Sul calcolatore è installato un SO Linux Debian con kernel 2.6.32-5-amd64 (Squeeze).

L'acquisizione e l'elaborazione del video sono effettuati da un software scritto in linguaggio C, che utilizza le librerie OpenCV, versione 2.3.1.

Lo streaming video è realizzato tramite Javascript.

Fig.1: L’area di lavoro

2. Il problema affrontato

Il lavoro prevede il raggiungimento di due obiettivi principali: la sostituzione via software dello sfondo dell'immagine servita in streaming e la determinazione della posizione dei robot, da effettuare in tempo reale.

2.1. Sostituzione dello sfondo e streaming web

Questa funzionalità prevede che, nel video acquisito dalla telecamera, la superficie del tavolo (sfondo) sia sostituita con un'immagine statica selezionabile dall'utente. La sostituzione deve avvenire per tutta l'area dove i robot si muovono normalmente. La sostituzione deve essere effettuata con un frame rate sufficientemente elevato da dare l'impressione di effettivo movimento (non meno di 10 frame/s). Il video così modificato deve essere reso disponibile per lo streaming in modo da consentire una facile visualizzazione all'interno di pagine web, con una risoluzione che varia a seconda della dimensione della finestra del browser. Le tre risoluzioni supportate sono: 300x225px, 400x300px, 500x375px.

È ammesso un modesto ritardo nella trasmissione delle immagini, non superiore a 0,5 secondi.

2.2. Determinazione della posizione del robot

Questa parte del lavoro prevede che i robot, opportunamente dotati di marker passivi, vengano identificati all'interno delle immagini rilevate dalla telecamera, e che la loro posa (coordinate X e Y di un punto significativo al loro interno, e orientamento Theta rispetto al tavolo) sia fornita periodicamente su un apposito socket TCP. La posizione così rilevata deve essere corretta per compensare le deformazioni introdotte dalla telecamera. La frequenza di acquisizione delle posizioni non deve essere inferiore a 2 Hz.

La precisione della determinazione della posizione non deve essere influenzata dalla risoluzione usata per lo streaming.

3. La soluzione adottata

La soluzione adottata per risolvere il problema può essere suddivisa in due parti ben distinte, implementate con due software diversi.

La prima riguarda l'acquisizione del video e l’applicazione di tutte le tecniche di computer vision necessarie per determinare la posizione del robot ed effettuare la sostituzione dello sfondo. È stata realizzata avvalendosi del linguaggio C e delle librerie OpenCV.

La seconda riguarda lo streaming web dei risultati ottenuti dalla prima. È stata realizzata tramite il linguaggio Javascript.

Per quanto riguarda l'elaborazione delle immagini, le varie sottoparti, descritte in dettaglio nei prossimi paragrafi, si avvalgono di un' unica struttura dati, contenente tutti gli elementi necessari. I più significativi sono:

- *CvCapture* capture*; Rappresenta la risorsa webcam
- *IplImage* prevFrame*; Rappresenta il frame precedente a quello attuale.
- *IplImage* currentFrame*; Rappresenta il frame soggetto all'elaborazione nel ciclo corrente.
- *IplImage* output*; Rappresenta il frame da visualizzare in output.
- *IplImage* section*; Rappresenta la più piccola immagine rettangolare di dimensioni dinamiche che contiene il robot in movimento. Questa sarà l'immagine su cui verrà effettuata l'analisi per il riconoscimento della posizione del robot.
- *int section_offset_x; int section_offset_y*; Sono gli offset della section rispetto al frame originale.

- *int x; int y; int theta*; Rappresentano le coordinate aggiornate e l'azimuth del robot in movimento.

Il main consiste nell'invocazione delle seguenti funzioni:

- *Webcam init()*; Il cui compito è quello di inizializzare l'intera struct sopra citata;
- *void end(Webcam);* Distrugge la stuct prima di terminare il programma;
- *void nextFrame(Webcam);* Questa funzione si occupa di far evolvere i componenti della struct effettuando tutte le operazioni necessarie al raggiungimento degli obiettivi;
- *void saveFrames(Webcam w);* Il cui compito è quello di salvare in una certa directory tutti i frame che dovranno essere visualizzati sul web.

3.1. Sostituzione dello sfondo

La sostituzione dello sfondo sfrutta la tecnica del background subtraction. Essa è in grado di rilevare cambiamenti tra un frame di un flusso video, ed il frame precedente. Il concetto che è alla base di questa tecnica consiste nel considerare uno sfondo fisso e ricercare delle aree, all'interno del quadro, che variano rispetto alle zone statiche. Tale ricerca è effettuata pixel per pixel.

L'uso dell'algoritmo si basa soprattutto sull'identificazione e sulla definizione di tutto quello che deve essere considerato come lo sfondo statico della scena rispetto a quello che, invece, si suppone sia in movimento. A seconda degli scenari, si tende a considerare come sfondo ogni parte della scena che è sempre immobile o che varia periodicamente; in ogni caso, le parti che compongono il background devono variare così lentamente da poter essere considerate statiche rispetto all'intero periodo di osservazione.

Questa tecnica ci permette quindi di identificare i robot soltanto in caso si muovano almeno una volta, in questo modo anche le zone di sosta vengono associate allo sfondo.

Nel caso in cui i robot, dopo essersi mossi per la prima volta, effettuino delle soste verrà visualizzato l'ultimo frame registrato soggetto a movimento.

La funzione che si occupa di ciò è: IplImage* backgroundSubtraction(IplImage*, IplImage*); che acquisisce in ingresso due frame in scala di grigi e ne effettua la differenza creando un nuovo frame e restituendolo, mentre spetta a void nextFrame(Webcam); fare tutto il resto.

Il fatto di identificare la porzione di frame che varia è utile anche per la determinazione della posizione del robot. Infatti le operazioni descritte nel prossimo paragrafo vengono effettuate solo in un intorno di quella porzione del frame, e non su tutta l'immagine, permettendo di ridurre il tempo computazionale.

Fig.2: due frame consecutivi (precedente e corrente)

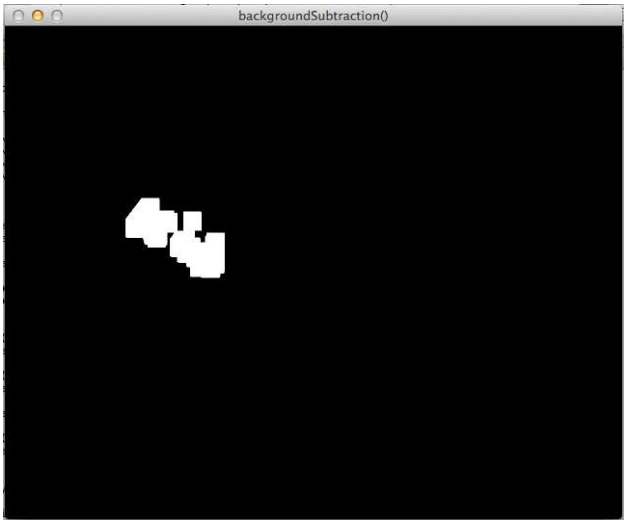

Fig.3: Background subtraction tra i 2 frame di fig.2

Fig.4: La più piccolo porzione del frame che contiene il robot.

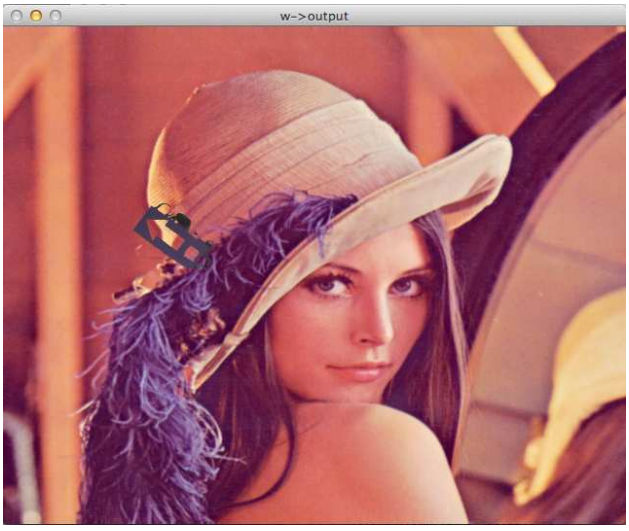

Fig.5: L'immagine di output

3.2. Determinazione della posizione dei robot

Ogni robot presente sul piano di lavoro è identificato da un marker, posto sul “dorso” dello stesso, in modo che sia inquadrato dalla webcam.

Il marker è composto da due parti: una qualsiasi immagine, detta immagine principale, che abbia determinate caratteristiche (descritte nel paragrafo 4.4) e un'altra immagine, detta retro, con una forma geometrica semplice.

Come si evince dal nome, l'immagine retro deve essere posta dietro l'immagine principale, verso la parte posteriore del robot.

Attualmente l'immagine principale è costituita da una freccia colorata, il retro da un quadrato bianco.

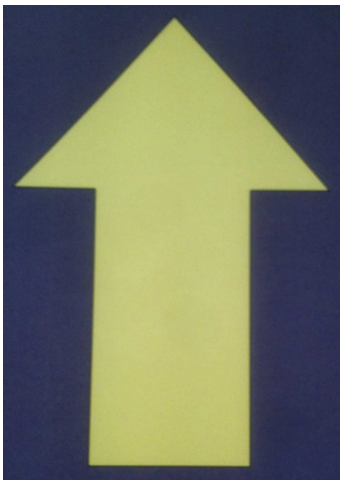

Fig.6: L’immagine principale attuale

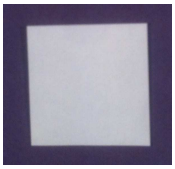

Fig.7: L’immagine retro attuale

All'avvio il software analizza il campione dell'immagine principale, tramite la funzione *void descriviCampione(IplImage *img)*, per estrarne le informazioni che saranno necessarie al riconoscimento e alla determinazione della posizione del robot.

Lo stesso fa per il campione dell'immagine retro, mediante la *funzione void descriviRetro(IplImage *img).*

Nel dettaglio, per il campione dell'immagine principale, vengono svolte le seguenti operazioni:

1. Estrazione dei bordi dell'immagine
2. Approssimazione a poligono del bordo con area maggiore
3. Memorizzazione dei vertici del poligono in una struttura dati con visibilità globale
4. Determinazione delle regioni presenti nell'immagine tramite tecniche di region growing
5. Costruzione dei tre istogrammi R,B,G della regione corrispondente al poligono determinato ai punti precedenti.

A questo punto l'immagine campione è descritta, e sarà usata per cercare immagini simili nei frame del video acquisito dalla webcam.

Per il campione dell'immagine retro vengono implementati solo i primi 3 punti. Questo per alleggerire il codice, considerando che questa immagine è una forma geometrica semplice, e dunque richiede meno informazioni per il riconoscimento, mentre quella principale può essere più complessa.

Ogni frame acquisito in input dal programma viene inizialmente elaborato dalle funzioni per il background subtraction descritte nei paragrafi precedenti. Queste funzioni determinano la porzione di video che varia da un frame all'altro ed invocano la funzione *void matching(IplImage *img)* passandogli come parametro la porzione di immagine che conterrà il robot, l'unico elemento che si muove nel video.

La funzione matching esegue le seguenti operazioni:

- Estrazione bordi dell'immagine.
- Approssimazione a poligono di tutti i bordi.

In seguito per ognuno dei poligoni trovati vengono effettuate le seguenti operazioni:

- Calcolo del perimetro.
- Se il perimetro rientra in un intervallo che indica la possibilità che appartenga all'immagine retro, vengono effettuate le operazioni per determinare se lo sia effettivamente. Se l'intervallo indica la possibilità che appartenga all'immagine principale vengono invece effettuate le operazioni per determinare la veridicità di questa ipotesi. Se il perimetro non appartiene a nessuno di questi intervalli, il poligono viene scartato.

Per determinare se un poligono rappresenta la stessa immagine del campione principale vengono svolte le seguenti operazioni:

1. Calcolo del numero dei vertici. Se questo è compreso nell'intorno [verticiCampione-1, verticiCampione+1] si prosegue, altrimenti il poligono viene scartato.
2. Determinazione delle regioni presenti nel frame tramite tecniche di region growing.
3. Costruzione dei 3 istogrammi R,B,G della regione corrispondente al poligono attuale all'interno del frame.
4. Confronto dei 3 istogrammi con quelli dell'immagine campione tramite la funzione cvCalcEMD2. Se la differenza tra essi è inferiore ad un certo intervallo (determinato sperimentalmente) si prosegue, altrimenti il poligono viene scartato.
5. Confronto della forma del poligono con quello dell'immagine campione tramite la funzione cvMatchShapes. Il risultato restituito dalla funzione viene memorizzato.

Infine, viene cercato il poligono con il risultato di cvMatchShapes minore.

Se questo valore è inferiore ad una certa soglia, determinata sperimentalmente, esso viene eletto come poligono corrispondente all'immagine campione.

Altrimenti si ipotizza che nel frame non sia presente l'immagine campione, e si passa al frame successivo.

Per determinare se un poligono rappresenta la stessa immagine del campione retro, vengono effettuati solo i punti 1 e 5, cercando quello con il risultato di cvMatchShape minore.

Una volta trovate le due immagini all'interno del frame, viene determinata la posizione del robot.

Coordinate X,Y

Per determinare queste coordinate si usa il baricentro dell'immagine principale, il quale è stato calcolato durante la fase di region growing, ed è dunque già disponibile.

Coordinata Theta:

Per calcolare la coordinata Theta è necessario trovare la direzione ed il verso del robot (dato che può muoversi in retromarcia).

Avendo a disposizione i baricentri dell'immagine principale e di quella retro, si calcola il coefficiente angolare della retta che li congiunge.

Una volta trovato il coefficiente angolare si determina l'angolo della retta rispetto all'asse orizzontale dell'immagine, tramite la nota formula: *Alpha=atan(m).*

Dato che l'immagine retro è posta sempre dietro quella principale, osservando le posizioni relative dei due baricentri, si capisce in che modo è orientato il robot. A seconda dell'orientamento si aggiunge una

costante opportuna all'angolo Alpha calcolato precedentemente. Questo porta ad avere Theta, che rappresenta l'angolo del robot rispetto all'asse verticale.

Le coordinate così trovate vengono scritte in un file, con il seguente formato: x-y-theta (es. 123.7-12.4-93.1) .

Le coordinate hanno i seguenti versi all'interno dell'area di lavoro:

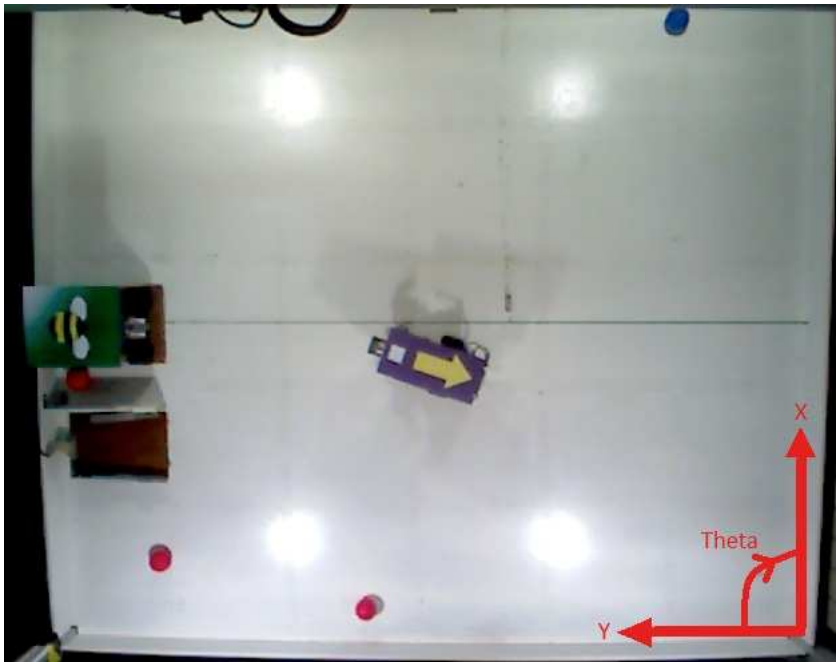

Fig.8: L'area di lavoro con il suo sistema di riferimento

Nell'immagine seguente si vede come il programma trovi all'interno del frame l'immagine principale, quella retro ed il segmento che unisce i loro baricentri.

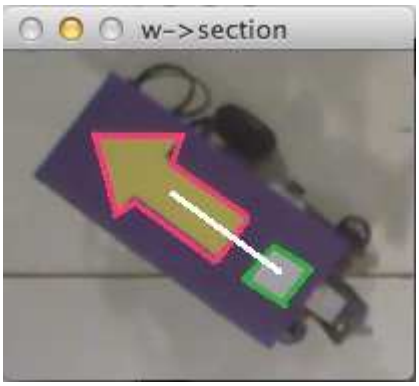

Fig.9: Matching tra il frame e le immagini campione e determinazione del segmento che unisce i loro baricentri

3.3. Streaming web

Per fare in modo che il sistema sia semplice, cross-platform e cross-browser, si è realizzato lo streaming web utilizzando Javascript. In questo modo il client non necessita di plugin per il browser, ed il server non necessita di applicativi specifici.

Il software adibito all'elaborazione del video, descritto nei paragrafi precedenti, crea ad intervalli di tempo predefiniti sei file jpeg compressi, rappresentanti il frame elaborato e il frame originale, in tre diverse dimensioni per entrambi, che vengono continuamente sovrascritti.

Spetta al browser del client continuare ad aggiornare la stessa immagine, con una frequenza tale da dare l'impressione di riprodurre un video.

L'utente può scegliere se visualizzare lo streaming del video originale oppure la versione con lo sfondo modificato.

In questo modo si ha la certezza che la sequenza di immagini (il filmato) venga visualizzato su qualsiasi piattaforma, inoltre è possibile garantire risoluzioni diverse dello stesso flusso video.

A seconda della dimensione della finestra del browser infatti, viene mostrata una versione diversa della stessa immagine.

Per implementare questo si è fatto fatto ricorso alla libreria per Javascript jQuery. Grazie alla quale è possibile identificare facilmente e dinamicamente la risoluzione del browser del client e conseguentemente variare la dimensione del flusso video.

Poichè a differenza dei video un'immagine può essere visualizzata in una pagina web anche senza essere stata completamente scaricata, sorge il problema di bufferizzare tale flusso. Per risolverlo è stato sufficiente creare un buffer composto da due immagini: gli ultimi due frame generati dal software che elabora il video.

Nello specifico è stato creato creato, nella pagina html, un , il quale contiene un tag <img> adibito alla visualizzazione di tutti i frame correnti.

Il problema sorge in due situazioni:

- quando il software scritto in C sta creando il file in output e nello stesso istante javascript lo sta aggiornando nella pagina;
- quando il browser sta cercando di visualizzare l'immagine nella pagina, ma per un qualsiasi motivo l'immagine non viene scaricata velocemente.

Nel file sorgente in HTML della pagina web visualizzata dal browser, l'url del frame corrente è contenuto nell'attributo *src* del tag <*img*>.

Per aggiornare il frame, ed ottenere così l'effetto video, tramite Javascript viene modificato l'attributo *src*, mettendo come url il nuovo frame. In questo modo il browser chiede al server la nuova immagine e la sostituisce a quella vecchia, senza che sia necessario ricaricare l'intera pagina.

Per evitare problemi dovuti al caching, il browser viene forzato, sempre tramite Javascript, a non mantenere la cache.

Inoltre nella pagina web è possibile visualizzare due tipi di filmati: quello originale e quello con sostituzione dello sfondo.

4. Modalità operative

4.1. Componenti necessari

L'eseguibile del programma per il riconoscimento è compilato per la distribuzione di Linux Debian.

È disponibile anche il file sorgente. Per compilarlo è necessario utilizzare un qualsiasi compilatore per C/C++.

Affinchè sia possibile compilare il sorgente scritto in C è necessario avere installate le librerie openCV 2.3.1 disponibili all'indirizzo web: http://sourceforge.net/projects/opencvlibrary/files/

Per lo streaming, è necessario un qualsiasi web server. Il client dovrà disporre di un browser grafico con Javascript attivato.

4.2. Modalità di installazione

I file del programma sono contenuti nelle seguenti directory:

- Il file binario in */usr/local/bin*
- Le immagini campione in */etc/mobolab_vision*
- La pagina HTML, le immagini generate dal programma ed il file delle posizioni: */var/www/mobolab/mobolab_vision/output*

Se è necessario installare le librerie OpenCV per ricompilare il tutto, queste sono le istruzioni per Linux (per altri sistemi operativi rifarsi alla guida ufficiale: http://opencv.willowgarage.com/wiki/InstallGuide/)

Aprire un terminale e digitare:

sudo apt-get install build-essential libgtk2.0-dev libavcodec-dev libavformat-dev libjpeg62-dev libtiff4-dev cmake libswscale-dev libjasper-dev

In seguito è necessario installare le JDK:

sudo add-apt-repository ppa:sun-java-community-team/sun-java6

sudo apt-get update

sudo apt-get install sun-java6-jdk

Scaricare le openCv all'indirizzo riportato nel capitolo 4.1, aprire il terminale, spostarsi nella cartella dove è stato salvato il file e digitare:

tar -xvf OpenCV-2.3.1.tar.bz2

A questo punto si creerà una cartella che nominata OpenCV-2.3.1; digitare in seguito da terminale:

cd OpenCV-2.2.0/

cmake .

Controllare che nella schermata non appaia nessun errore e procedere digitando:

make

ed infine:

sudo make install

4.3. Modalità di esecuzione

Utilizzando un sistema UNIX, il comando per compilare correttamente il file sorgente è:

g++ `pkg-config --cflags opencv` mobolab_vision.cpp -o mobolab_vision `pkg-config --libs opencv`

Il file binario va posto nella directory: /usr/local/bin

Il programma, avviato tramite riga di comando, deve ricevere come argomento il path del file usato come background. Per avviare l'eseguibile si digita dunque nel terminale:

/usr/local/bin/mobolab_vision path_background

4.4. Modalità di taratura

A seguire l'elenco dei parametri personalizzabili (sono tutti delle define) per tarare la sensibilità del programma, adattandolo eventualmente a nuove esigenze.

- OUTPUT_DIRECTORY path della directory ove verranno salvate le immagini in output. È necessario che questa directory sia raggiungibile dalla pagina html avente il compito di visualizzarla sul web
- CAMPIONE path dell'immagine campione per l'immagine principale posta sul robot
- CAMPIONE_RETRO path dell'immagine campione per l'immagine retro posta sul robot
- POSIZIONI path del file che contiene l'ultima posizione rilevata del robot
- JPEG_QUALITY serve per minimizzare le dimensioni delle immagini in output (applicando una maggiore compressione del formato) a discapito della qualità
- PIXEL_CHANGED_MIN_PERC e PIXEL_CHANGED_MAX_PERC rappresentano l'intervallo percentuale entro il quale il numero di pixel variati costituiranno un movimento significativo della scena al fine di identificare il robot
- BACKGROUND_SUBTRACTION_THRESHOLD è la soglia alla quale viene effettuata la distinzione tra sfondo e robot. Questo è il parametro fondamentale su cui agire nel caso venga cambiato il colore della piattaforma su cui i robot si muovono.
- int output_widths[] = {500, 400, 300}; è la larghezza delle immagini che verranno create in output (l'altezza viene calcolata proporzionalmente)

Durante la fase di matching, sia per l'immagine principale che per quella retro, i poligoni trovati vengono discriminati in base al perimetro, per evitare di effettuare operazioni complesse come cvMatchShape o cvCalcEMD2 su candidati che sicuramente non possono essere l'immagine cercata.

- MIN_PERIMETRO_RETRO è il valore minimo del perimetro affinchè il poligono venga preso in considerazione per la ricerca dell'immagine retro.
- MAX_PERIMETRO_RETRO è il valore massimo del perimetro affinchè il poligono venga preso in considerazione per la ricerca dell'immagine retro.
- MIN_PERIMETRO_CAMPIONE è il valore minimo del perimetro affinchè il poligono venga preso in considerazione per la ricerca dell'immagine campione.
- MAX_PERIMETRO_CAMPIONE è il valore massimo del perimetro affinchè il poligono venga preso in considerazione per la ricerca dell'immagine campione.

Se si desidera cambiare l'immagine posta sul robot bisogna tenere in considerazione questi parametri, ed eventualmente modificarli dopo averli determinati sperimentalmente con la nuova immagine.

All'interno del file sorgente, nella parte finale della funzione *matching()*, è presente una porzione di codice attualmente commentata utilizzabile per questo scopo.

Per determinare i parametri sopra elencati è sufficiente togliere i vincoli attuali, ponendo ad esempio i perimetri minimi a 0 e quelli massimi a 9999, decommentare le seguenti istruzioni:

cvShowImage("Matching", img2);

cout<<"Perimetro: "<<cvArcLength(risultato, CV_WHOLE_SEQ,1);<<"\n";

sleep(2);

e avviare il software. Durante l'esecuzione appariranno simultaneamente, in una finestra la porzione di immagine identificata come quella campione, evidenziata da un bordo rosa (come in fig.5) e stampato a console il valore del perimetro. Se il matching è corretto si prende nota del perimetro.

Dopo una decina di risultati corretti è possibile stimare i valori da inserire per il perimetro.

Un'altra discriminazione che viene effettuata, questa volta solo per l'immagine principale, è quella sull'istogramma. Viene calcolata la somiglianza tra gli istogrammi del campione dell'immagine principale e la porzione di frame contenuta nel poligono candidato.

Se questa somiglianza è superiore ad una certa soglia, il candidato viene scartato, altrimenti si prosegue con i controlli.

- MAX_DIST_ISTOGRAMMA_ACCETTABILE rappresenta questa soglia.

Sempre durante la fase di matching si cerca il poligono con il valore dato dalla funzione cvMatchShape minore, che dovrebbe rappresentare quello più somigliante all'immagine campione. Tuttavia se il valore del poligono eletto come "vincitore" è superiore ad una certa soglia si presuppone che non sia quello cercato, e quindi per quel frame non viene trovato nessun matching.

- MAX_SOMIGLIANZA_CAMPIONE_ACCETTABILE rappresenta la soglia per il poligono che deve rappresentare l'immagine principale.
- MAX_SOMIGLIANZA_RETRO_ACCETTABILE rappresenta la soglia per il poligono che deve rappresentare l'immagine retro.

Non dovrebbe essere necessario modificare questi parametri, anche in caso di modifica dell'immagine. In ogni caso la procedura da seguire è simile a quella per il perimetro:

Togliere i vincoli attuali, ponendoli ad esempio a 9999.

Decommentare le seguenti istruzioni:

cvShowImage("Matching", img2);

cout<<"somiglianza: "<<somiglianzaMin<<"\n";

cout<<"somiglianzaRetro: "<<valMS_min<<"\n";

cout<<"Dist Istogramma: "<<distanzaFinale<<"\n";

sleep(2);

e avviare il software. Durante l'esecuzione appariranno simultaneamente, in una finestra la porzione di immagine identificata come quella campione, evidenziata da un bordo rosa (come in fig.5) e stampati a console i valori trovati per le soglie sopra descritte. Se il matching è corretto si prende nota delle soglie.

Dopo una decina di risultati corretti è possibile stimare i valori da inserire.

4.5. Avvertenze

Per il corretto funzionamento del software è necessario che i robot si muovano uno alla volta.

Caratteristiche delle immagini campione

L'immagine principale posta sul robot deve avere le seguenti caratteristiche:

- Essere monocromatica e continua.
- Avere un colore che contrasti con lo sfondo anche in scala di grigi. Ad esempio un'immagine rossa (0,255,0) su sfondo blu (255,0,0) non va bene, poiché in scala di grigi immagine e sfondo avrebbero la stessa luminosità.
- Essere approssimabile con un poligono abbastanza bene.
- Essere simmetrica, rispetto al proprio asse verticale, in modo che il suo baricentro sia all'incirca sulla metà della sua larghezza

L'immagine digitale usata come campione per l'immagine principale deve avere le seguenti caratteristiche:

- Essere in formato JPG.
- Contenere solo ed esclusivamente l'immagine (e ovviamente una porzione del suo sfondo).
- Avere il lato più corto di almeno 500 pixel, per permettere una buona descrizione.
- Essere una fotografia dell'immagine posta sul robot, scattata nelle stesse condizioni di luminosità a cui lavora il software.
- Essere orientata in modo che la parte di immagine che “guarda” la prua del robot sia verso l'alto nella fotografia.
- Deve chiamarsi “campione.jpg” ed essere contenuta nella directory */etc/mobolab_vision/campioni*

L'immagine retro posta sul robot deve avere le seguenti caratteristiche:

- Essere una forma geometrica semplice e poligonale (es. quadrato).
- Avere un buon contrasto rispetto allo sfondo.
- Essere posizionata verso la poppa del robot, in modo che la retta che unisce il suo baricentro con quello dell'immagine principale abbia lo stesso verso del robot.

L'immagine digitale usata come campione per l'immagine retro deve avere le seguenti caratteristiche:

- Essere in formato JPG.
- Contenere solo ed esclusivamente l'immagine (e ovviamente una porzione del suo sfondo).
- Avere il lato più corto di almeno 400 pixel, per permettere una buona descrizione.
- Essere una fotografia dell'immagine posta sul robot, scattata nelle stesse condizioni di luminosità a cui lavora il software.
- Deve chiamarsi “campioneRetro.jpg” ed essere contenuta nella directory */etc/mobolab_vision/campioni*

5. Conclusioni e sviluppi futuri

Riassumendo, si è realizzato un sistema software che, acquisito in input il video di un robot che si muove, sostituisce lo sfondo originale con un’immagine statica, riconosce la posizione del robot in un sistema di riferimento, rendendo disponibile l’informazione tramite web e invia in streaming il video del robot che si muove, permettendo all’utente di scegliere se visualizzare il video originale o la versione con lo sfondo modificato.

Alcuni sviluppi futuri potrebbero essere:

- Gestire il movimento contemporaneo di entrambi i robot nell'ambiente di lavoro
- Usare le informazioni prodotte dal software in merito alla posizione dei robot per implementare ulteriori controlli, oppure effettuare la correzione della posizione, oppure gestire la guida manuale ecc.
- Implementare un sistema di calcolo automatico dei parametri.
- Implementare un sistema di generazione automatica delle immagini campione.

Indice