Relazione al progetto: Naruto Hand Seals Fighting del corso di Sistemi e Applicazioni Multimediali

Michele Tamburini
mtamburi@cs.unibo.it

3 gennaio 2011

1 Abstract

Il progetto nasce con lo scopo di esplorare uno dei campi della *Augmented Reality*, che prevede l'uso delle mani nell'interazione uomo-macchina. La realizzazione ultima consiste di un gioco di combattimento in modalità picchiaduro che prende ambientazione e scenografia dall'ormai noto anime giapponese Naruto. Le mosse per così dire speciali, dovranno essere attivate non con semplici combinazioni di tasti, ma attraverso le posizioni ed i gesti delle mani dell'utente, acquisiti attraverso una webcam, che saranno poi analizzati da un motore di riconoscimento per verificarne l'accuratezza. L'implementazione del presente prevede il motore di riconoscimento e una semplice sezione di addestramento per l'utente, che lo coinvolga nell'apprendimento di alcune mosse.
Vengono utilizzate librerie *OpenCV* per l'analisi delle immagini, *Guichan* per l'interfaccia grafica, *SDL* ed il linguaggio C++.

2 Stato dell'arte

La ricerca di tecniche che favoriscano metodologie di utilizzo dei calcolatori attraverso forme di interazione sempre piú innovative, quali ad esempio l'uso del proprio corpo, ha aperto la strada giá da qualche tempo a nuove sfide per il mondo informatico. Il campo di ricerca dell'*"hand tracking"* vuole dare all'utilizzatore del calcolatore la possibilitá di un controllo, completo o parziale, di questo, attraverso i complicati movimenti o stimoli che la mano umana é in grado di fornire. L'acquisione di questi, sia essa agevolata dall'ausiglio di sensori particolari, guanti o addirittura con la mano nuda, risulta essere la prima problematica da affrontare, comune a tutte le proposte fino ad ora osservate ([11] [16] [17] [20] [19]. Gli spunti proposti nella bibliografia, lontano dall'essere esaustiva sull'argomento, vogliono semplicemente offrire una panoramica ad ampio spettro di alcune soluzioni a tale problema.

3 Progetto: introduzione

3.1 Ambientazione

L'ormai celeberrimo anime "Naruto" ha portato anche in Italia una storia di fanstasia dalle forti connotazioni orientali. Tra gli ingredienti maggiormente apprezzati dal grande pubblico risiede il fatto che i personaggi utilizzano movimenti delle mani estremamente complicati per lanciare mosse di arti marziali speciali. Ognuna di queste è composta da un numero variabile di "sigilli" (posizioni degli arti superiori) che corrispondo ai 12 segni zodiacali cinesi.
Nel seguente progetto ho inserito tutti e soli tali gesti, ovvero non si tengono in considerazione movimenti diversi (come il battito delle mani) che pur compaiono nell'anime. Lo scopo del giocatore sarà quello di riprodurre alcune posisioni che vedrà raffigurate sottoforma di immagini. Ciascuna di queste è denominata segno, o sigillo (signs, seals). Una

mossa (Move) è dunque costituita da un numero variabile di segni presi tra i 12 sopra descritti.

3.2 Il nostro problema nel dettaglio

In questo lavoro non ho certo la pretesa di affrontare un tema cosí imponente come quello dell'hand tracking. Il problema si riduce infatti ad una analisi di immagini seppur non priva di compromessi e difficoltà. A differenza di alcuni lavori di spicco che usano un database e oggetti esterni per l'inferenza della reale posizione della mano [19], in questo caso non è necessario avere una precisa cognizione del punto di partenza e di fine dell'arto. Il giocatore è libero di eseguire qualunque movimento e, una volta pronto, aziona l'acquisione del gesto, che si traduce per il calcolatore con: l'acquisizione di alcuni frame di immagine, trasformazione di questi con tecniche di processazione dell'immagine e confronto con un template precedentemente preparato. Un punto che abbatte notevolmente la dimensione del problema, ed evita quindi l'uso di un database, si basa proprio su una carattestica derivata dall'ambientazione: sebbene le combinazioni che possono avere luogo siano infinite (permutazioni di 12 elementi di lunghezza arbitraria) i diversi gesti da riconoscere rimangono comunque 12.

3.3 Pensando in grande

Fondamentalmente questo gioco risiede nella categoria "picchiaduro". L'ingrediente della *Augmented Reality* emerge nel momento in cui l'utente abbandona la tastiera per interagire con le proprie mani attraverso la webcam ed eseguire dal vivo i gesti necessari per la mossa selezionata.
Per ciascuno gli verrà notificato il livello di accuratezza calcolato in una scala da 0 a 100. Al termine dell'ultimo viene poi restituito il voto medio.

4 Introduzione alla Libreria OpenCV

OpenCV è una libreria per l'analisi delle immagini estremamente versatile e completa. Nonostante la mia completa ignoranza a riguardo, mi sono convito a sfruttarla fin dalle fasi di studio di fattibilità del progetto, spinto sia dal suggerimento avuto dalle lezioni del corso di Sistemi e Applicazioni Multimediali [13] che dall'innumerevole ammontare di commenti positivi reperibili dalla rete.
Sono necessarie solo poche nozioni per prendere dimestichezza con questa. Prima fra tutte la gerarchia delle strutture dati (CvArr, CvMat), ed in particolare quella usata per la memorizzazione dell'immagine: IplImage. Quindi l'uso delle principali funzioni di analisi delle immagini: cominciando con l'applicazione dei filtri basilari (Laplaciano, Sobel, di Smoothing...) per arrivare al dominio delle frequenze, rilevazione dei contorni, utilizzo della webcam (o addirittura multi-cam). Estremamente apprezzabile è la possibilità di un approccio "lazy", per usare un termine informatico. In un certo senso è possibile iniziare la fase di programmazione senza preoccuparsi dell'uso preciso di una determinata funzione, utilizzandola dapprima con impostazioni di default, per poi scendere nei dettagli solo in un secondo momento.
Altro pregio della libreria è la facilità della visualizzazione del risultato grafico a schermo (attraverso le funzioni cvNamedWindow(), cvShowImage()) che consentono una veloce analisi del risultato delle operazione all'interno di ogni singola parte del progetto.
Bisogna osservare tuttavia che, sebbene i primi capitoli del manuale "Learning OpenCV" [14], collocato probabilmente tra i più consigliati nella rete, siano davvero utili per approcciarsi alla libreria, lo stesso non si possa dire delle sezioni più specifiche, che non sostituiscono un libro di testo per l'analisi delle immagini. Durante tutto il periodo di sviluppo risulta però necessario armarsi del manuale ufficiale [9], disponibile anche online.
Un secondo punto a favore di tale libreria è la reperibilità di numerosi frammenti di codice ed esempi che si possono trovare in rete, ma che chiaramente vanno affrontati con la giusta ponderazione prima di essere utilizzati nel proprio lavoro.

5 Descrizione del progetto

Questa sezione contiene la descrizione della parte centrale del lavoro svolto.

5.1 Sezione di Acquisizione

Con il termine acquisione ho inteso la cattura di frames attraverso la webcam. Webcam pensata in-

tegrata all'interno del pc portatile, anche se nulla esclude l'utilizzo di un dispositivo diverso, purchè riconosciuto dal sistema, e impostato per essere quello di default.
L'oggetto Camera si compone semplicemente di una classe in grado di inizializzare una struttura di tipo Capture (della libreria OpenCV), e sfruttarla per reperire un immagine di tipo IplImage*. Su questa avranno quindi luogo la fase di processazione e valutazione della macchina di riconoscimento (vedi di seguito).

5.2 Logica di Gioco

Per consentire lo sviluppo di un'applicazione che mostri i connotati per una stesura pressoché completa, è necessario immergerla nello scenario dell'ambientazione almeno dal punto di vista logico. Ho volutamente evitato tediosi dettagli, come ad esempio l'occorrenza nella storia di una particolare mossa o il personaggio che l'ha effettuata per la prima volta, che rendono magari l'anime estremamente interessante, ma che ai fini dell'attuale lavoro non sono poi così rilevanti.
Tuttavia ciascuna mossa viene catalogata in base al proprio tipo, rango di difficoltà e livello, in maniera il più possibile fedele all'anime. Per un compito così strutturato mi sono affidato alla versalitità del linguaggio XML, e l'uso della libreria "TinyXML" [7] per la sua manipolazione. Grazie a questi accorgimenti risulterà forse più agevole l'espansione del codice in futuro.
A questo si è poi aggiunto un forte apporto ingegneristico per rendere fin dal principio tale sezione manutenibile senza troppi sforzi. In figura 2 è riportato il diagramma UML della logica di gioco.

5.3 Stack di Gioco

Non è difficile osservare come questa sezione, così come pure l'interfaccia grafica, non sono state l'oggetto di cure maniacali. In particolare ho voluto create strutture architetturalmente valide che dessero il massimo riuso di codice, ma allo stesso tempo sufficientemente versatili in termini di espandibilità.
Il risultato di tali ragionamenti è sintetizzato dal diagramma 3. La macchina di gioco (*GameMachine*) altro non è che un semplice contenitore di elementi, come ad esempio i menu o la sezione di addestramento (*TrainingSection*), ciascuno dei quali deve provvedere la funzione che ricoprirà il ruolo di ciclo principale. Infatti la GameMachine dispone tutti i moduli contenuti in una struttura a stack ed eseguirà sempre la funzione di quello in cima.
A ciascuno di questi, sia esso un menu, o un modulo differente, viene affidato un controller, che deve essere in grado di gestire eventi conformi all'interfaccia SDL.

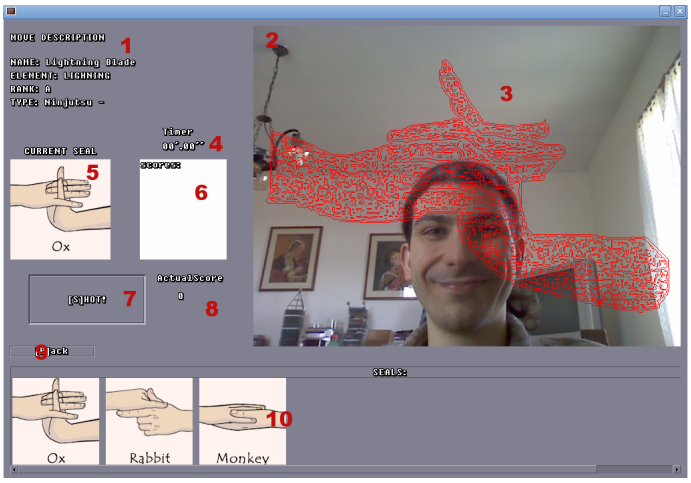

Figura 1: Screenshot dell'applicazione durante una esecuzione

5.4 L'interfaccia Grafica

Ciascun elemento di gioco (*GameElement*) ha la necessità di mostrastri all'utente attraverso un'interfaccia grafica. La libreria Guichan [2] mi è sembrato il compromesso migliore tra quelle che erano le potenzialità utili a ricoprire le richieste grafiche, tutt'altro che esigenti, e compatibilità con le librerie nativamente previste per il progetto. Questa si trova perfettamente compatibilie con SDL. Il punto di contatto tra OpenCV, per l'acquisizione delle immagini dalla webcam, e l'integrazione con la sezione grafica è reso possibile grazie alla convesione dall'immagine IplImage di OpenCV alla superficie SDL (SDL_Surface) e quindi alla visualizzazione tramite l'opportuno oggetto Guichan. Il risultato del procedimento è quanto mostrato dal punto 2 di figura 1.

Di seguito elenco le sezioni della finestra, mostrata in 1, descrivendole brevemente:

1. descrizione della mossa: oltre al nome vengono visualizzate anche il tipo, il grado di difficoltà (non di esecuzione) ed il nome reale giapponese;
2. schermo che mostra l'immagine acquisita istantaneamente attraverso la webcam;
3. sono la traccia della corretta posizione delle mani e delle dita: più il giocatore riuscirà ad esservi fedele e migliore sarà il punteggio ottenuto;
4. timer: appena premuto il pulsante "Shot!" (o usata la relativa abbreviazione), viene attivato un cronometro di 3 secondi per dare modo al giocatore di riposizionarsi correttamente;
5. mostra la posizione delle mani per eseguire il sigillo attuale;
6. contiene la lista dei punteggi ottenuti per ciacun sigillo fino al momento attuale;
7. è il pulsante che attiva il cronometro e quindi la fase di acquisizione;
8. mostra in tempo reale il punteggio che si otterrebbe nel caso scattasse l'acquisione;
9. pulsante che retrocede al menu della lista delle mosse per la selezione di una nuova;
10. sigilli necessari al compimento della mossa attuale.

6 Riconoscimento

Il cuore del progetto è costituito dalla macchina di riconoscimento del gesto effettuato. Come già accennato la qui presente soluzione applica combinazioni di semplici tecniche di analisi delle immagini. Per quanto riguarda la realizzazione della sezione di riconoscimento, fin dalla primordiale idea, ho voluto mantenere un approccio il più possibile modulare e versatile ai cambiamenti, in modo da poter poi provare agevolmente diverse metodologie di elaborazione. La macchina corrisponde ad un contenitore di moduli, ciascuno dei quali esegue una ben precisa operazione sull'immagine.
Il settaggio completo completo di questa prevede: la scelta dei moduli da inserirvi ovvero la strategia, una funzione per il processing ed una per la valutazione. Una volta applicato questo, se si volesse scendere nel dettaglio di ogni singolo passo per ogni frame il si osserverebbe un procedimento che consiste nel:

- acquisire il frame attraverso la webcam,
- richiamare la funzione di **processing** della macchina di riconoscimento,
- sfruttare poi la funzione di **valutazione** fornendo come parametri il risultato della fase precedente ed il template di riferimento per il sigillo corrente,
- raccogliere il risultato ottenuto.

I **template**, di cui si è parlato anche in precedenza, corrispondo alle immagini dei sigilli prive di sfondo o forme di rumore, processate attraverso una particolare strategia e sfruttate quindi dalla funzione di valutazione. Nella sezione *Template Creation* si trova infatti l'occorrente a questo scopo: lanciando lo script che vi si trova vengono convertite le immagini orginali "grezze" in template e poste nella directory di destinazione dalla quale si lancerà l'eseguibile.
È bene tenere a fianco il diagramma di figura 4 per avere chiare le diverse parti di questa sezione.

6.1 Moduli e Strategie

Indicherò con il termine **Modulo** una classe ereditata da *EngineModule* che implementa la funzione

```
int compute(const IplImage* src,
  IplImage* dst)
```

e può quindi essere aggiunta nell'insieme dei moduli (*ModulesPool*) della macchina di riconoscimento. L'implementazione di questa dovrà prevedere dunque l'azione di processing, per la quale il modulo stesso è pensato, sull'immagine in ingresso src, e la restituzione del risultato in dst. Bisogna tener presente che è rilevante la posizione di ciascun modulo. Dal momento che l'unità di dialogo tra questi sono le immagini, dunque matrici di pixel, l'applicazione in ordine diverso di filtri o altre operazioni quali l'estrazione dei contorni, può condurre a risultati estremamente diversi. Un esempio canonico

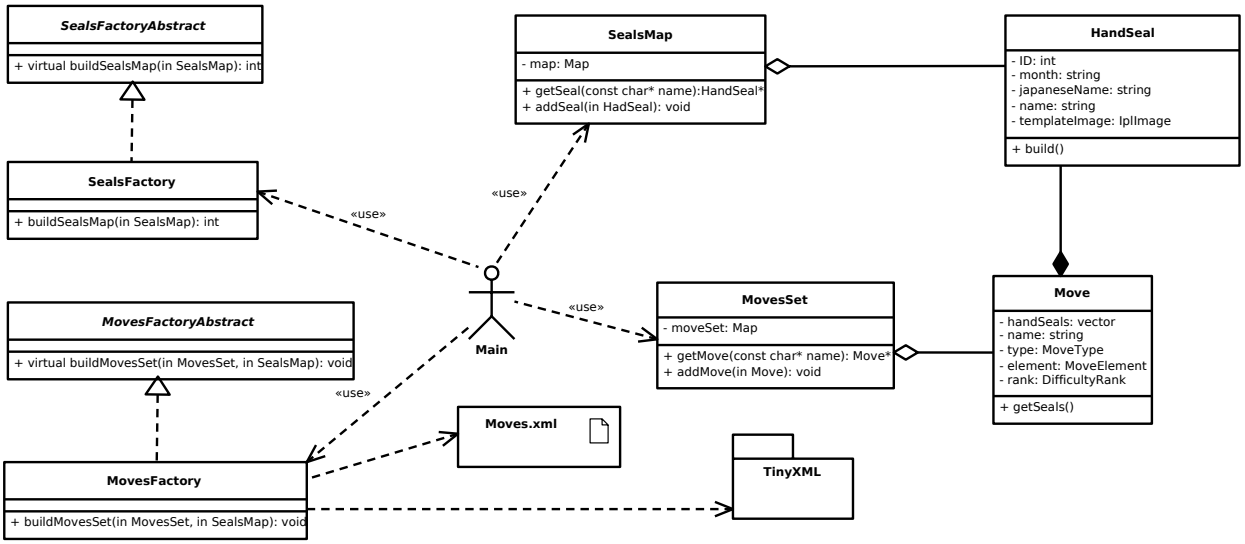

Figura 2: Uml diagram of the GameLogic section.

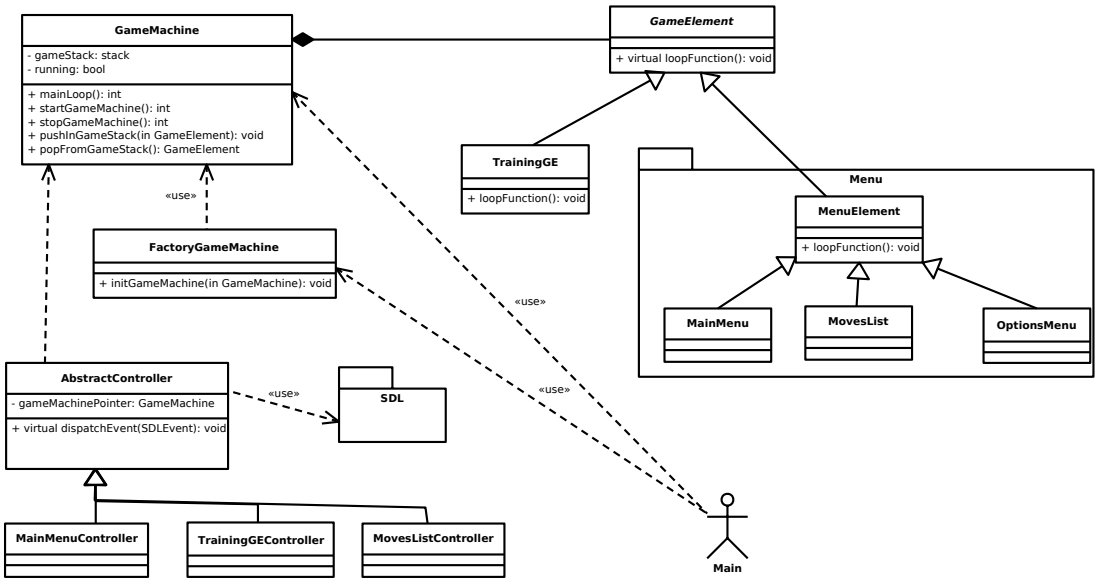

Figura 3: Uml diagram of the GameStack section.

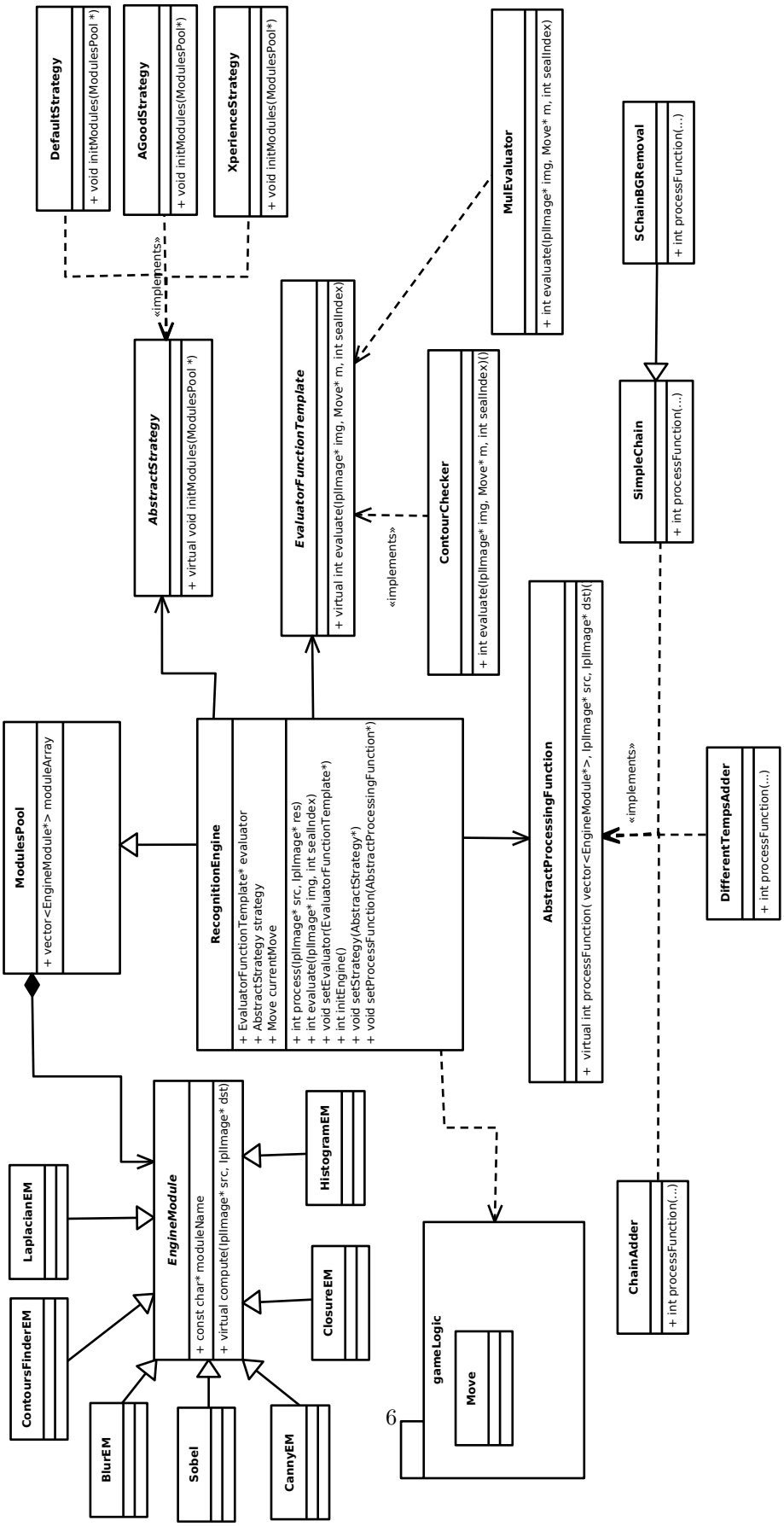

Figura 4: Uml diagram of the Recognition Engine Section.

riguarda il filtro di smoothing posto in genere sempre prima di un operatore Laplaciano.
Un **strategia** invece definisce quali moduli inserire nel bacino della macchina, in che ordine, e con quali carattestiche. Ciascuna classe appartenente alla gerarchia che si estende dalla classe *AbstractStrategy*, adempie a questo compito.

6.2 Rimozione dello sfondo

Un modulo di estrema importanza è quello della rimozione dello sfondo. La capacità di individuazione delle forme viene fortemente condizionata dalla situazione luminosa della stanza e dalla complessità e dal numero di oggetti presenti alle spalle del giocatore. Con complessità si intendono sia la forma stessa che le caratteristiche riflessive (superfici lucide o opache).
Per fare un esempio che aiuti alla comprensione ritorniamo per un istante allo scenario mostrato dalla camera in figura 1. Le fonti luminose di forte intensità sono tra le peggiori cause di rumore. Nel lampadario posto alle spalle del giocatore sono poste due lampadine a basso consumo a luce bianca: esso rappresenta un ottimo esempio di disturbo dell'acquisizione se testato di sera e con luci accese. Un ulteoriore esempio, che porta facilmente a falsi positivi, è rappresentato dai due quadri posti sul muro: essi spiccano senza alcuna difficoltà nell'analisi di rilevazione dei contorni, in particolare se vi si aggiunge il fatto che tendono a riflettere buona parte dei raggi di luce che li colpiscono (vedi angolo inferiore del quadro a destra).
La rimozione dello sfondo adottata sfrutta un principio di facile intuizione: prevede cioè l'acquisizione di alcuni frame, utili a catturare le parti fisse della scena, prima della fase di gioco, momento in cui il giocatore posiziona e muove le mani. Il modello di sfondo proposto fa poi uso della statistica e si individuano la media e la correlazione delle immagini catturate.
Sia $p(x, y)$ un'immagine, si calcolano:

$$S(x, y) = \sum_{f=1}^{N} p(x, y) \tag{1}$$

$$Sq(x, y) = \sum_{f=1}^{N} p(x, y)^2 \tag{2}$$

dove $f = frame$ e $N = numero\ totale\ di\ frame$.

Dunque la media sarà:

$$M(x, y) = \frac{S(x, y)}{N} \tag{3}$$

e la deviazione standard:

$$\sigma(x, y) = \sqrt{\frac{Sq(x, y)}{N} - M(x, y)^2} \tag{4}$$

Questo definisce il nostro modello per la rilevazione degli oggetti fissi nella scena. Data una immagine acquisita in un secondo momento si può sfruttare la quantità appena ricavata per dividere gli elementi in primo piano da quelli sullo sfondo con la seguente procedura.
Sia $P(x, y)$ la nuova immagine acquisita, allora:

$$|M(x, y) - P(x, y)| \leq \lambda\sigma(x, y) \tag{5}$$

dunque:

$$|M(x, y) - P(x, y)|^2 \leq (\lambda\sigma(x, y))^2$$

$$|M(x, y) - P(x, y)|^2 \leq \lambda^2\sigma(x, y)^2$$

dove $\sigma(x, y)^2$ è:

$$\sigma(x, y)^2 = \frac{Sq(x, y)}{N} - M(x, y)^2 \tag{6}$$

E λ è una costante che può essere settata a 3, in accordo con quanto ricavato da [8].

6.3 Processazione e valutazione

6.3.1 L'algoritmo scelto

7 Alcuni Test

8 Osservazioni e sviluppi futuri

9 Coclusioni

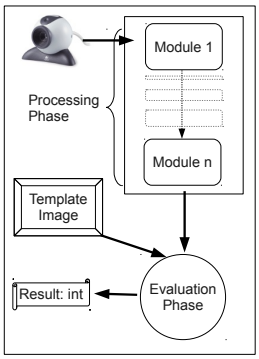

Figura 5: Schema del procedimento di processazione e valutazione dell'immagine in input

A quick look to GOD

Riferimenti bibliografici

[1] Cmake: official wiki. http://www.cmake.org/Wiki/CMake.

[2] *Guichan wiki*. http://code.google.com/p/guichan/.

[3] Leafninja website. http://www.leafninja.com/.

[4] Opencv: Official wiki. http://opencv.willowgarage.com/wiki/.

[5] Sdl simple directmedia layer forums. http://forums.libsdl.org/viewtopic.php?t=6442&sid=7e7ea3a081a8906de37de3c2f2548b76.

[6] Stackoverflow: blog. http://stackoverflow.com/questions/393954/how-to-convert-an-opencv-iplimage-to-an-sdl-surface .

[7] Tinyxml: xml manager library. http://sourceforge.net/projects/tinyxml/.

[8] The basics of background substraction, Mar 2009. http://blog.damiles.com/?p=127.

[9] *OpenCV Reference Manual v2.1*, Mar, 18 2010.

[10] Flavio Bernardotti. *Computer Vision Encyclopaedic Forum*. 15100 Alessandria.

[11] Ramon Mas Cristina Manresa, Javier Varona and Francisco J. Perales. Hand Tracking and Gesture Recognition for Human-Computer Interaction. In *Electronic Letters on Computer Vision and Image Analysis*, volume 5, pages 96–104. 2005.

[12] Erdem Yörük, Ender Konuko ğlu, Bülent Sankur, Jérôme Darbon. Shape-based hand recognition. *IEEE TRANSACTIONS ON IMAGE PROCESSING*, 15(7), June 2006.

[13] Marco Roccetti Paola Salomoni Stefano Ferretti. *Slide e appunti di lezione del corso di Sistemi ed Applicazioni Multimediali*, Feb-May 2009-10. http://www.cs.unibo.it/~roccetti/multimedia-09-10.html.

[14] Adrian Kaehler Gary Bradski. *Learning OpenCV*. O'Really, 2008.

[15] Intel Corporation. *Open Source Computer Vision Library*. http://developer.intel.com.

[16] Reinhard Klein Markus Schlattmann. Simultaneous 4 gestures 6 dof real-time two-hand tracking without any markers. In *ACM Symposium on Virtual Reality Software and Technology*.

[17] Adrien Bernhardt Marie-Paule Cani Paul G. Kry1, Adeline Pihuit. Handnavigator: Hands-on interaction for desktop virtual reality. In *ACM Symposium on Virtual Reality Software and Technology*, pages 53–60, 2008. http://hal.inria.fr/inria-00336348/fr/.

[18] Aaron Saenz. Mit's ridiculously colorful glove is latest hand tracking interface. In *human computer interfaces*. May, 28 2010. http://singularityhub.com/2010/05/28/mits-ridiculously-colorful-glove-is-the-latest-hand-tracking-interfacevideo/.

[19] Robert Y. Wang. Real-time hand-tracking as a user input device. Monterey, CA, October 19-22 2008.

[20] A. Argyros X. Zabulis, H. Baltzakis. Vision-based hand gesture recognition for human-computer interaction. In *The Universal Access Handbook*, 2009.