Progetto d'informatica
Esame di stato
anno 2012/13

Relazione

Computer Vision

Autore: **Domenico Luciani**
Classe: 5B Informatica Abacus

- # Introduzione

Il progetto si basa sull'utilizzo delle librerie *OpenCV,* delle librerie utilizzate nel campo della robotica per la visione artificiale.
Al giorno d'oggi vengono usate praticamente ovunque, dalle macchine fotografiche digitali alle macchine industriali, dai robot più semplici a quelli più complessi.

Apprendendo l'esistenza di queste librerie open source (sotto licenza BSD, originariamente sviluppate dall'intel) mi sono voluto cimentare in piccoli progetti che testavano più o meno approfonditamente tale strumento.
È reperibile online un'ottima documentazione dal quale è possibile studiare e approfondire le tantissime funzioni messe a disposizione.

Per finire questo progetto ho impiegato vari mesi per studiare, approfondire, scrivere codice, debuggare ed implementare il tutto arricchendo la mia conoscenza in proposito.

Relazionerò il progetto da me creato spiegando passo per passo tutto quello che ho utilizzato, perché l'ho fatto e come l'ho fatto.
Inserirò alla fine della relazione una copia dei sorgenti con licenza GPL 3 con i relativi commenti.
In modo tale da seguire passo passo la spiegazione senza confondersi.

Ho sviluppato tutti i sorgenti presentati usando il compilatore *g++* e l'editor di testo *vim*.
Per poter compilare ed usare i programmi bisogna aver installato *ffmpeg* ed *opencv* sul proprio sistema.
Al momento utilizzo la versione 0.11 di ffmpeg e la versione 2.4.1 delle OpenCV.

Per poter compilare i sorgenti bisogna eseguire questa stringa:
g++ `pkg-config –cflags –libs opencv` -pthread nome_sorgente.c -o nome_binario

e per eseguirlo:

```
./nome_binario 1
```

o

```
./nome_binario 0
```

Rispettivamente 0 per la webcam interna e 1 per la webcam esterna.

N.B. In compilazione potrebbero esserci dei warning, il tutto è normalissimo dato che io compilo con g++ (le opencv sono sviluppate in c++) ma uso il C (è possibile anche usarle in java, in python e in altri linguaggi).

La distro GNU/Linux su cui ho testato e creato il tutto è una *Slackware 13.37 -current*.

I software da me realizzati sfruttano tecniche di manipolazione delle immagini, color tracking, head tracking, movement detecting, applicazioni di filtri e tanto altro.

Consiglio di leggere questa relazione con i sorgenti a portata d'occhio in modo tale da vedere ciò a cui mi riferisco.

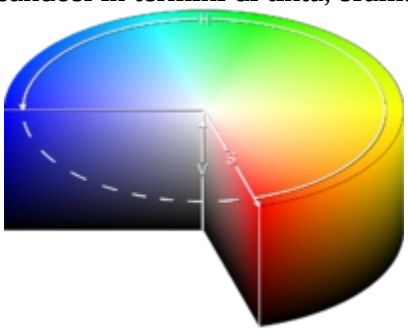
- # Funzioni

Per sviluppare il mio progetto ho creato una piccolissima libreria che mi consente di semplificarmi il lavoro evitando di riscrivere più volte le stesse cose.
La libreria si trova nella cartella “lib” dove possiamo trovare i file “f*unzioni.h*” e “f*unzioni.c*” rispettivamente l'header e la libreria con l'implementazione delle funzioni.

Inizialmente troviamo una struttura *HSV* che ci servirà nei prossimi sorgenti per semplificarci il lavoro durante l'utilizzo di questo modello additivo di composizione dei colori con 3 interi che rappresentano rispettivamente il valore della tonalità (H = Hue), il valore della saturazione (S = Saturation) e il valore della luminosità (V = Value).

- ## HSV

Il modello HSV è orientato per emulare il più possibile la visione umana essendo basato sulla percezione che si ha di un colore basandosi in termini di tinta, sfumatura e tono.



Modello HSV
Il sistema di coordinate è cilindrico ed è definito come un cono distorto.
H = Angolo intorno all'asse verticale.
S = La saturazione va da 0, sull'asse del cono, a uno sulla superficie del cono.
V = La luminosità è l'altezza del cono.

Soppianteremo quindi l'RGB per usare l'HSV così da aumentare la possibilità ed ottimizzare il tracking del colore che ci serve.

Dopo troviamo una semplice struttura *Rettangolo* che ci mette a disposizione degli interi per definire i vertici della figura geometrica.
Questa struttura ci consentirà di definire il rettangolo che potremmo disegnare sull'immagine o anche solo rilevare se abbiamo o meno toccato un punto specifico da noi interessato.

Proseguendo troviamo la funzione l*eggiConf*, che passati i puntatori alla struttura HSV per i primi due parametri più il file di configurazione da cui trarremmo i dati ci permetterà di prelevare i diversi valori che ci consentiranno di trovare il nostro colore all'interno dell'immagine per poi stampare su console i valori presi.
Questa funzione verrà usata all'interno di ogni programma che dovrà trackare un determinato colore scelto da noi.

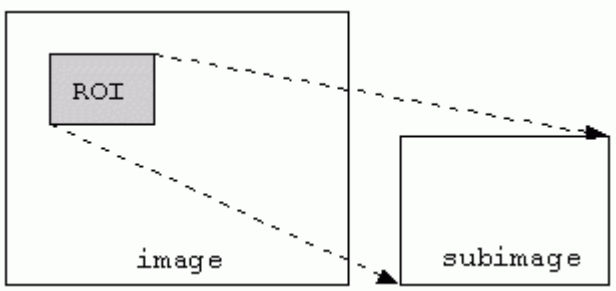
Continuando troviamo la funzione *scriviConf* che accetta gli stessi parametri della funzione l*eggiConf* con la differenza che questa funzione permette il salvataggio dei dati sul file di configurazione.
Questa funzione ci permetterà di applicare a tutti i sorgenti che usano il file di configurazione di apportare le modifiche del colore una volta sola senza bisogno di editare o ricompilare i sorgenti.

La funzione *diminuisci* che troviamo dopo permette di ridurre la grandezza di un'immagine di una determinata percentuale scelta da noi.
La funzione ritorna un puntatore alla struttura *IplImage* definita nelle openCV che ci consente la gestione delle immagini e accetta come parametri un puntatore a IplImage (quindi un'immagine) e un intero che c'indicherà la percentuale di riduzione.
All'interno della funzione creiamo un'immagine più piccola grazie alla funzione *cvCreateImage* indicando la grandezza dell'immagine da creare (in questo caso la larghezza moltiplicata la percentuale diviso 100,), i il numero di bit per pixel (8) e il numero di canali dell'immagine (va da 1 a 4).
Passiamo il tutto alla funzione *CvResize* che si occuperà di ridurre l'immagine iniziale ed adattarla all'immagine successivamente creata per poi ritornare il puntatore a quell'immagine,
Per abbreviare le cose mi sono creato una funzione *riduciNoise* che riceve come parametri un'immagine sorgente e un'immagine destinataria effettuando la cosiddetta “apertura” e “chiusura” dell'immagine dilatando l'immagine, erodendola e poi applicare lo smooth.
La dilatazione genera un incremento delle dimensioni dell'immagine riempiendo i buchi e le aree mancanti unendole tra di loro.
L'erosione genera un decremento delle dimensioni dell'immagine rimuovendo piccole anomalie.
Lo smooth è un particolare filtro per pulire l'immagine da disturbi vari evidenziando i pattern significativi attenuando il rumore.
Questa funzione verrà utilizzata per migliorare il controllo sull'immagine così da avere meno disturbi e quindi meno falsi positivi.

Dato che in giro si trova poco ho anche creato la funzione inserisci che prende come parametri un'immagine piccola, un'immagine grande e 2 interi.
Tale funzione permette d'inserire un'immagine piccola in un'immagine più grande in delle coordinate definite dai 2 interi passatogli.
Una volta dentro la funzione creiamo una variabile con le coordinate ricevute e la dimensione dell'immagine piccola.
Dopo di che settiamo un *ROI* nell'immagine grande di dimensioni contenute in dim nella posizione contenuta sempre in dim.
Dopo copiamo l'immagine piccola nell'immagine grande per poi resettare il ROI.

- **ROI**

Il ROI è la cosiddetta *Region Of Interest.*
Cioè un'area in un'immagine che c'interessa.



Una volta definito il ROI molte funzioni di OpenCV andranno a lavorare SOLO in quella locazione tralasciando tutto il resto. (come vediamo in cvCopy che copierà la piccola immagine SOLO nel ROI grande quanto se stessa lasciando stare tutto il resto)
Una volta resettato il ROI tutto torna normale.

Il ROI è utilissimo soprattutto per concentrarci su porzioni dell'immagine senza processare su tutta l'immagine, il che non solo ci permetterà di alleggerire i calcoli ma anche d'incentrarci maggiormente su dettagli che altrimenti verrebbe scomodo analizzare.

La libreria conclude qui.
Le funzioni potevano essere implementate ancor di più ma dato il semplice progetto da me creato non ne ho visto il bisogno.

Ora andremo ad analizzare i file contenuti nella cartella *config*; la cartella contenente il file di configurazione dove troveremo il programma di calibrazione per l'identificazione del colore che ci permetterà d'identificare il range che più c'interessa.

- # Calibrazione

Partiamo includendo le librerie necessarie al corretto funzionamento delle OpenCV cioè *cv.h* e *highgui.h* e poi la libreria *funzioni.h* vista in precedenza.

Definiamo dove trovare il file di configurazione, il nome della gui dove mostreremo l'immagine del frame preso dalla webcam e il nome della gui dove mostreremo il risultato in binario dei valori presi.

Controllo se ho inserito i parametri giusti in modo tale da sapere quale webcam usare.

Inizializzo la webcam con la funzione *cvCaptureFromCAM* specificando come parametro 0 per la webcam interna e 1 per la webcam esterna.
Setto le proprietà della webcam utilizzando le dimensioni 640x480
Prendo il primo frame e lo salvo in imm.
In questo momento imm conterrà un frame estrapolato dalla nostra webcam.
Creo un'immagine hsv dove salveremo l'immagine convertita in hsv di dimensioni pari a quelle contenute in imm.
Creo un'immagine binary dove salveremo l'immagine binaria che troveremo.

Prendiamo lo step dell'immagine binary; il campo *widthStep* indica il numero di bytes tra l'inizio di ogni riga di pixel.
Dopo abbiamo target, cioè i pixel veri e propri dell'immagine.
Abbiamo però bisogno di castare come unsigned char la variabile perché di suo è definita come char* e la maggior parte delle volte tratteremo immagini unsigned char.

Definiamo una variabile tasto che useremo per prendere il tasto premuto dall'utente.

Allochiamo 2 puntatori alla struttura HSV e un puntatore alla struttura Rettangolo viste precedentemente.

Prendiamo i valori dal file di configurazione e li mettiamo rispettivamente in low e high.

Dopo creiamo 2 gui.

Ora entriamo nella parte viva del programma, cioè nel while che avrà il compito di ripetersi finché imm non ritorna false.

Creiamo delle trackbar con cui potremmo interagire e vedere il cambiamento in tempo reale.
Le trackbar saranno 6, ognuna di essi gestirà il valore minimo di H,S,V e il valore massimo di H,S,V che vorremmo analizzare.
I valori vanno da 0 a 255 e le trackbar appariranno nella gui NORMAL.

Ruotiamo il frame in modo tale da averla specchiata a noi.
Convertiamo l'immagine da BGR ad HSV ed inseriamo l'immagine convertita in hsv.
La funzione *cvInRangeS* è una funzione di filtraggio.
Attraverso i valori di massimo e minimo che gli vengono passati lei come risultato darà un'immagine binaria con pixel bianchi per identificare la zona trovata e pixel neri per tutto il resto.

Riduco i disturbi nell'immagine binaria (puntini bianchi che possono sfasare il processo di tracking)

Qui inizia la ricerca vera e propria, abbiamo la nostra immagine binaria con i suoi pixel bianchi, dobbiamo solo trovarli.
Per far questo facciamo una ricerca su ogni pixel che compone l'immagine.
Se troviamo un pixel bianco (255) salviamo i punti dei vertici del nostro rettangolo.

Con la funzione *cvRectangle* disegniamo il nostro rettangolo in imm (il nostro frame) di colore rosso.
Con la funzione *cvCircle* disegniamo il baricentro del nostro rettangolo di blu.

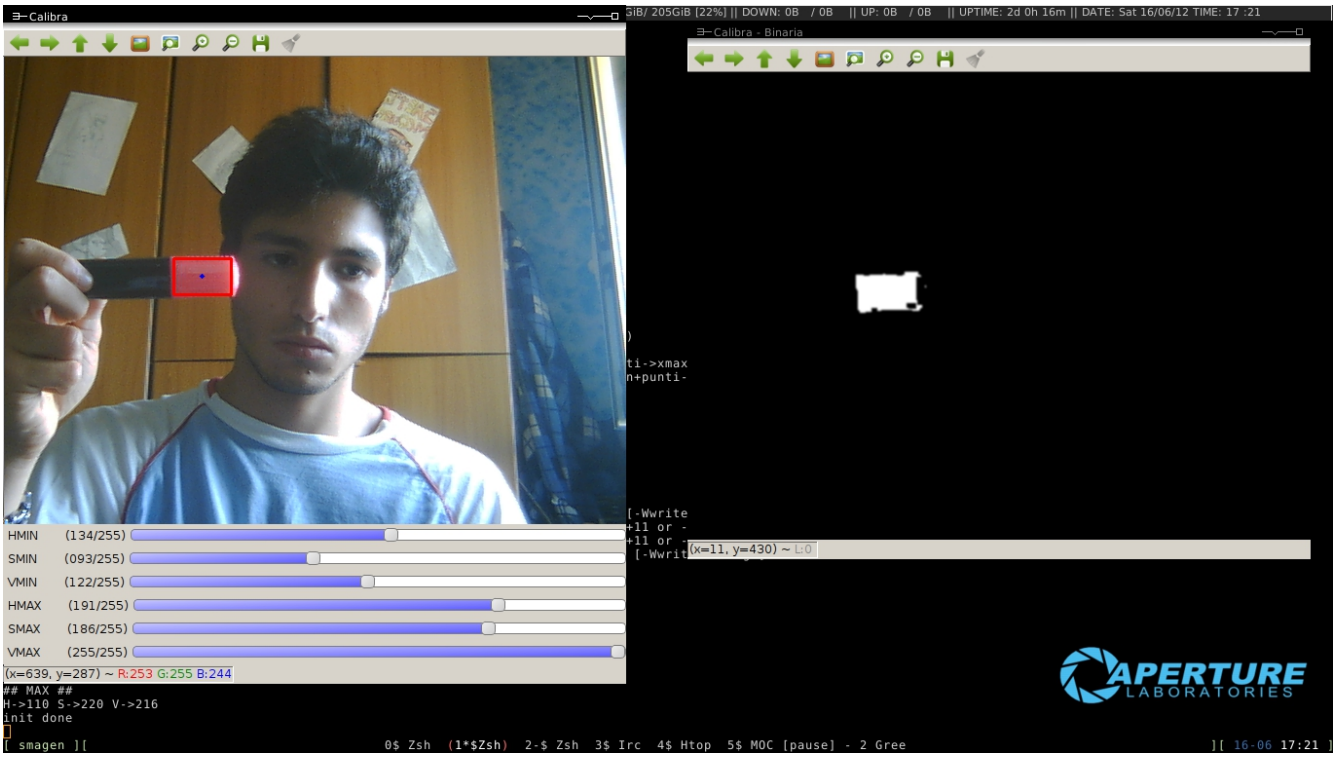
Dopo mostriamo le gui con il nostro frame (su cui abbiamo disegnato) e l'immagine binaria trovata

La funzione *cvWaitKey* aspetta 15 millisecondi e ci permette di controllare se abbiamo premuto un tasto, infatti con lo switch controlliamo:
Se premiamo il tasto 'q' il nostro programma termina.
Se premiamo il tasto 's' scriviamo i dati presi sul file di configurazione in modo che tutti i software che leggano da quel file usino esclusivamente i valori prelevati.

Alla fine prendiamo un altro frame e torniamo all'inizio del ciclo.

In caso di uscita deallochiamo il tutto.

Ecco uno screenshot dell'applicazione in funzione:



Come si può vedere abbiamo le trackbar e abbiamo il nostro rettangolino rosso solo ed esclusivamente intorno all'area interessata, cioè quella di colore fucsia.
Nella gui con l'immagine binaria possiamo vedere come venga preso solo il colore da noi impostato precedentemente segnando i pixel di bianco e mettendo di nero tutto il resto.
Il rettangolino seguirà quel colore per tutto lo schermo, segnando anche il baricentro.

- # Bubbles game

All'interno della cartella *Bubbles* troviamo il *bubbles game*, un piccolo giochino che ho creato che ci permette di far scoppiare le “bolle” su schermo semplicemente “toccandole”.

Per sviluppare questo giochino ho dovuto ricorrere all'uso dei *thread* poiché dovevo risolvere il problema di scrivere sul frame in modo asincrono rispetto alla velocità del frame preso.
Cioè, le “bolle” dovevano scendere più lentamente rispetto a quanti frame per secondo venivano presi quindi ho dovuto dividere il programma in più thread per gestire le bolle, i sistema del punteggio e tanto altro.
Il tutto gestito da *mutex* che gestiranno i vari thread.

Partiamo dallo includere le librerie utili allo sviluppo del software.
Definiamo il nome della gui, dove si trova il file di configurazione, il numero massimo di bolle da creare, il file dei punteggi, definiamo i valori per passare da un livello all'altro.
Dichiariamo delle variabili globali necessarie per il punteggio e per dire se abbiamo toccato una bolla “nemica”.

Creiamo una struttura Bolla che conterrà tutte le informazioni di ogni bolla come se fosse un oggetto con i vari attributi, questa struttura servirà ai thread per scambiarsi fra di loro varie risorse.

Ogni bolla avrà un proprio mutex e delle coordinate che verranno modificate ogni volta.
Viene anche memorizzato che tipo di bolla è, normale o nemica.

A questo punto vediamo la funzione *nuoviValori* che servirà ad inizializzare la bolla passata come parametro facendo ritornare la bolla all'inizio in una posizione pseudo randomica.

Ora troviamo la funzione *up*.
Questa funzione verrà eseguita in un thread a parte e gestirà le posizioni delle bolle.

Dentro ad un ciclo while infinito aspetta dei millisecondi a caso, blocchiamo il mutex.
Se non abbiamo preso un nemico controlliamo che la bolla non abbia raggiunto la fine della gui, se la bolla nemica raggiunge la fine incrementiamo il punteggio di +10 altrimenti lo decrementiamo di -10.
Se la bolla non ha raggiunto la fine della gui incrementiamo la sua coordinata y e le diamo nuovi valori.
Sblocchiamo il mutex.

Ora vediamo la funzione video, questa funzione è come se fosse la nostra funzione principale.
Dichiariamo le bolle e le variabili necessarie.
Facciamo esattamente le stesse cose che abbiamo fatto con il programma di calibrazione per quanto riguarda il color tracking.
Le funzioni *cvInitFont* inizializzano lo stile da usare per scrivere su un'immagine.

Alloco le Bolle e creo una prima bolla per poi creare il thread corrispondente passando la prima bolla.

Leggo qual è il punteggio massimo fatto fin'ora.

Entro nel ciclo while che si occupa di prendere i frame e gestire la stampa a video.
Effettuo le solite operazioni sul frame.
Creo un piccolo ciclo per far comparire a video la scritta “START”

Cerco in tutta l'immagine e se trovo il colore che cercavo faccio un ciclo verificando che io tocchi una delle bolle che fin'ora ho creato.
Se la tocco e vedo che è un nemico allora perdo, altrimenti blocco il mutex di quella determinata bolla, aumento di 10 il punteggio, scrivo “+10” sopra la bolla e do nuovi valori alla bolla sbloccando poi alla fine di tutto il mutex.

Se durante il ciclo ho toccato un nemico faccio un ciclo di pochi secondi dove faccio spuntare a video la scritta “GAME OVER”, il punteggio, il livello e il record.
Se supero il record scrivo il nuovo record sul file del punteggio.

Se non ho perso faccio disegnare su schermo ogni bolla che fin'ora ho creato, il livello, il record e il punteggio.

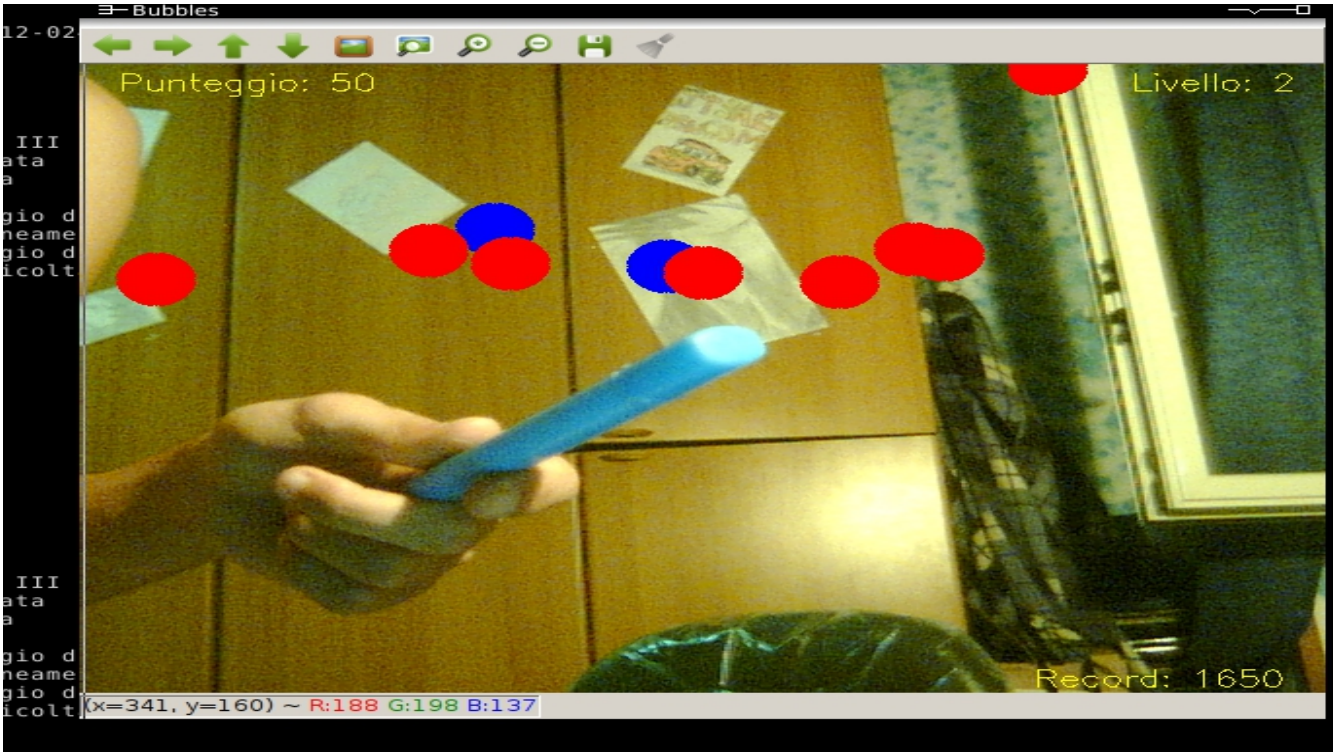
Ora vediamo la gestione del livello.
Se arrivo a 100, 500, 1000, 1500 aumento il livello aumentando le bolle e anche le bolle nemiche.
Dopo mostro il tutto.

Se premo 'q' esco e libero tutto.

Ora abbiamo il nostro main vero e proprio che si occuperà di creare il primo thread e aspetterà che

termini il suo ciclo.

Un giochino piuttosto semplice nella sua praticità ma è la prima cosa che mi è venuta in mente e farla in C non è stato uno scherzetto dal punto di vista ideativo.
Mostra tranquillamente come è possibile usare il color tracking per interagire con delle bolle disegnate su schermo.
Questo procedimento è possibile applicarlo ad immagini e ad altre figure.
La cosa che più conta è che la webcam riesca a trackare per bene il colore altrimenti avrà dei problemi a rilevare se tocchiamo o meno le bolle.
Ecco uno screenshot:



- **Draw**

Ora passiamo alla cartella *Draw*.

In pratica si tratta di un piccolo programmino che permette di “disegnare” sulla gui con un qualsiasi oggetto da noi scelto.
Per realizzarlo ho pensato di usare il frame, poi disegnare su un'altra immagine e alla fine fare il merge delle 2 immagini facendo spuntare l'immagine risultante a video.
In questo modo avremo il “disegno” sul frame in modo tale da non cancellare il disegno fatto ogni volta che prendiamo un nuovo frame.
Premendo 'c' il buffer viene ricreato cancellando il disegno fatto.
Inizialmente il buffer non viene usato, per far vedere il “puntatore”, inzierà a scrivere solo quando

Includiamo le solite librerie necessarie, definiamo i nomi delle gui e facciamo le solite cose.

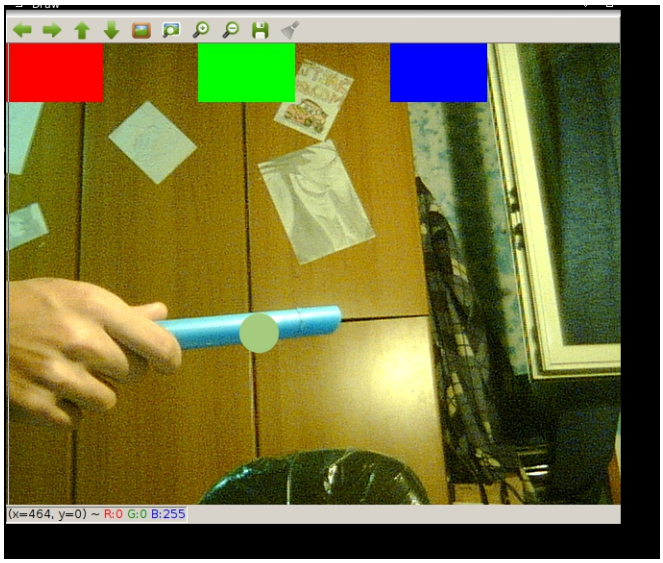
Creiamo 3 rettangoli di colori diversi: uno rosso, uno verde e uno blu tutti pieni.
Prendiamo come centro il baricentro dell'oggetto.
Se entro dentro un qualsiasi rettangolo imposto il colore del baricentro con il colore del rettangolo toccato.
Prendo il punto dove sono e se ho toccato un rettangolo scrivo un cerchio di colore pari al rettangolo toccato e faccio il merge.

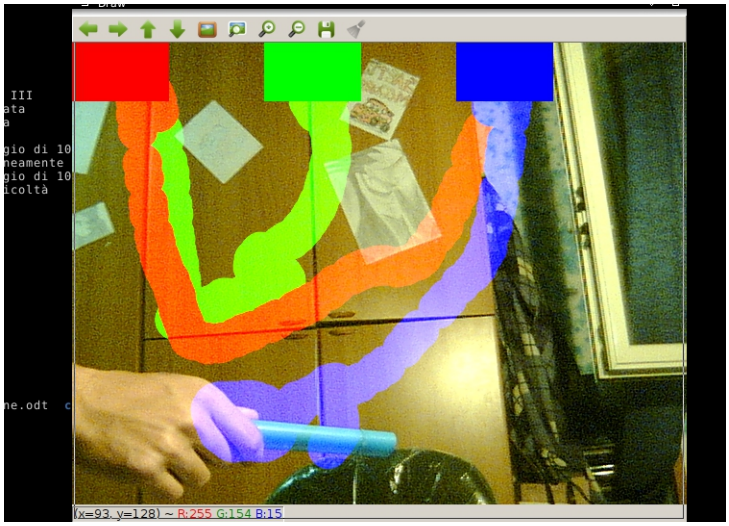
Se non ho toccato il rettangolo disegno un cerchio nel frame che verrà cancellato subito dopo, usato principalmente come “puntatore”.

Aspetto 15 millisecondi e se premo il tasto 'q' esco.
Se premo il tasto 'c' distruggo il buffer per ricrearlo vuoto e faccio tornare il puntatore.

Se esco distruggo tutto liberando memoria.

Il programma termina qui, la maggior parte delle funzioni viste le abbiamo incontrate precedentemente.
Quindi ho preferito evitare di essere troppo ridondante e di arrivare alla parte viva del programma tralasciando parti già viste.





- **Head**

Questo software utilizza gli H*aarCascade files* per identificare un particolare volto.
Di solito questo tipo di software viene usato nelle fotocamere digitali per riconoscere il volto delle persone.
Si basa su dei tipi di file che contengono dei particolari dati per riconoscere un volto, ricavati grazie a uno speciale procedimento che consiste nell'analizzare centinaia di foto di volti (positivi) e tanti background (negativi) per poi ricavare il file che più o meno bene permetterà il tracciamento di un qualsiasi volto.

Inizialmente abbiamo le nostre librerie.

Ho definito vari file cascade che è possibile utilizzare.
Nel mio software utilizzeremo soltanto quelli del volto e degli occhi.

Possiamo scegliere se rilevare solo il volto o anche gli occhi, di default è disabilitata questa funzione.

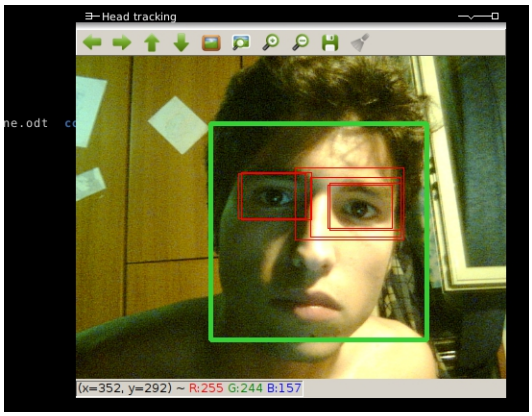
La funzione *rilevaDettagli* serve -se abilitato il RILEVA_EXTRA- di prendere oltre alla faccia anche gli occhi.

Creiamo dello spazio con la funzione *cvCreateMemStorage* e carichiamo il file cascade.
La funzione più interessante è la *cvHaarDetectObjects* che si occuperà d'individuare ciò che cerchiamo.
Se trovo qualcosa disegno un rettangolo attorno a ciò che ho trovato.
Mostro il risultato a schermo.

La funzione *rilevaVolto* è la funzione che come dice dal nome “rilverà i volti” all'interno dell'immagine.
Le prime righe sono uguali a quelle di prima.
Controlliamo se troviamo volti e dopo averli trovati controlliamo se dobbiamo cercare i dettagli.
Se dobbiamo cercarli ho ideato un metodo per semplificare la ricerca.
Setto il ROI per ogni volto trovato, in questo modo andremo a passare SOLO ed ESCLUSIVAMENTE l'immagine del volto trovato e non più l'intera immagine.
Alla fine resetto il ROI e disegno il rettangolo del volto che ho trovato.

Ora entriamo nel main dove effettuiamo le solite operazioni d'inizializzazione webcam, dichiarazione delle variabili,etc.

L'utilizzo dei file cascade risultano molto pesanti e quindi per ottimizzare le cose e rendere “fluida” la cosa ho ridotto l'immagine su cui lavorare del 50% per poi passarla come parametro alla funzione rilevaVolto, poi libero spazio e prendo un alto frame per poi aspettare 15 millisecondi per poter individuare se viene premuto il tasto 'q' dopo di che dealloco tutto.



Tale procedimento è molto efficace per quanto riguarda le immagini statiche come foto o altro.
Mentre è meno efficace quando dobbiamo usarlo durante uno stream video data la lentezza dell'algoritmo.

- **Movement**

Visto che i miei compagni hanno dovuto comprare un sensore per capire se c'era movimento nella stanza mi è balenata in mente l'idea di realizzarmi un sensore di movimento tutto mio.

Ho scelto 2 modalità per il sensore, dinamica e statica.
Questo piccolo software funziona facendo un confronto fra un frame e un altro.
Se trova delle differenze in alto fa spuntare “REC” seguito da una pallina rossa e inizia a salvare tutti i frame che trova “diversi”.
Quando si chiude il programma lui prima di terminare salva tutti i frame trovati e ne fa un video mettendo come nome la data corrente e l'ora.
In questo modo avremo a disposizione dei video con SOLO i cambiamenti.
Quindi i video non saranno pesanti parecchi GB di ore ma solo quei frame dove è passato qualcuno o si

è mosso qualcosa nell'ambiente.

La parte statica funziona in questo modo:
Abbiamo una parte da monitorizzare.
Posizioniamo la webcam verso la parte da controllare (assicurandoci di uscire dalla sua vista) e premiamo 'v'; da quel momento qualsiasi cosa passerà nella zona monitorizzata o se la web cam vedrà dei cambiamenti rispetto al frame preso da noi.

La parte dinamica funziona in una modalità simile alla precedente con la differenza che noi non dobbiamo interagire con il programma.
Il programma una volta avviato prenderà i vari frame ed ogni frame lo confronta con il precedente.
Se trova una discordanza fra i 2 salva il frame.
Alla fine crea un video dei frame presi esattamente come la modalità precedente.
In ogni frame per sicurezza salva orario e data.

Inizialmente includiamo le solite librerie.
Inizializziamo la webcam e creiamo le immagini che ci servono.
Creiamo la gui e tramite la funzione l*ocaltime* ci ricaviamo l'ora e la data corrente per poi salvare il tutto nella variabile nome.

Ci creiamo il writer passando come parametri il nome del file “o-r-a_d-a-t-a.avi”, la codifica (usiamo la divx per una compressione maggiore), i frame per secondo, la grandezza del video e se è un video a colori o meno.

Inizializziamo le scritte
e prendiamo i dati dell'immagine.

Facciamo scegliere la modalità, giriamo l'immagine aggiungiamo al frame l'orario e la data e dopiamo il frame in una variabile copia che ci servirà per il video in uscita, non dovrà essere modificata.

Riduciamo i disturbi nel frame ed entriamo nel ciclo.
Se abbiamo scelto la modalità dinamica controlliamo che quello corrente sia il primo frame.
Se è il primo frame creiamo una immagine temporanea dove copieremo questo primo frame.
Se non è il primo frame controlliamo le differenze fra il frame corrente e il precedente con la funzione *cvAbsDiff* passando tramite parametri il frame corrente, quello precedente e un'immagine contenente il risultato.
Dopo convertiamo l'immagine in bianco e nero ed effettuiamo un t*hreshold* dell'immagine
per poi ridurre i noise dell'immagine binaria ricavata.
Dopo copiamo il frame nella variabile temporanea aggiornandola.

Nel caso avessimo scelto la modalità statica controlliamo se abbiamo la nostra immagine temporanea contenente il frame da monitorare.
Se l'abbiamo scelta facciamo esattamente la stessa cosa di poco fa senza però copiare il frame, ovviamente.

Controlliamo tutta l'immagine binaria in cerca di ciò che cerchiamo, se troviamo un pixel bianco significa che abbiamo trovato differenze.

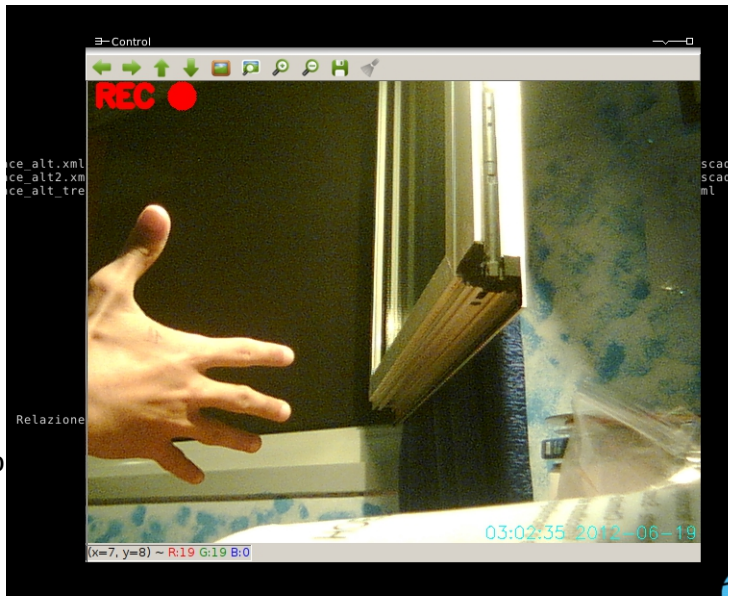
Se troviamo differenze scriviamo sul frame “REC O” così da avvertire che stiamo “registrando” e con *cvWriteFrame* memorizziamo il frame appena preso.

Mostriamo il tutto e aspettiamo 15 millisecondi.

Se premiamo il tasto 'v' e abbiamo scelto la modalità statica creiamo l'immagine temporanea e copiamo il frame dentro di essa per poi prendere un altro frame e tornare all'inizio del ciclo.

Una volta usciti dal ciclo Se abbiamo preso dei frame il nostro video verrà creato e salvato.

Una volta usciti lo possiamo vedere con un video player qualsiasi.



Il mio progetto termina qui.

Pensa di aver spiegato il più dettagliatamente possibile tutto ciò che ho usato e tutto ciò che ho fatto. Per eventuale documentazione ci si può rivolgere al sito ufficiale delle *OpenCV*
http://opencv.willowgarage.com/wiki/

Domenico Luciani
Classe 5°B
Informatica Abacus
Anno 2012/13

Sorgenti

funzioni.h

```
/*
# This file is part of Computer Vision Exam Project
#
# Copyright(c) 2012 Domenico Luciani
# domenico.luciani@email.it
#
#
# This file may be licensed under the terms of of the
# GNU General Public License Version 2 (the ``GPL'').
#
# Software distributed under the License is distributed
# on an ``AS IS'' basis, WITHOUT WARRANTY OF ANY KIND, either
# express or implied. See the GPL for the specific language
# governing rights and limitations.
#
# You should have received a copy of the GPL along with this
# program. If not, go to http://www.gnu.org/licenses/gpl.html
# or write to the Free Software Foundation, Inc.,
# 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.
*/

#include "funzioni.c"

/*
 * -Funzione per leggere il file di configurazione-
 * I primi 2 parametri indicano dove salvare i valori presi.
 * I valori presi vengono salvato su 2 strutture apposite.
 * Il terzo parametro è il nome del file da cui leggere.
 */
void leggiConfig(HSV*,HSV*,char*);
```

```
/*
 * -Funzione per scrivere sul file di configurazione-
 * Stessi parametri di prima.
 */
void scriviConfig(HSV*,HSV*,char*);

/*
 * -Funzione per diminuire la dimensione di un'immagine-
 *  Il primo parametro indica l'immagine.
 *  Il secondo parametro è il valore in percentuale di riduzione.
 *  Viene ritornata l'immagine modificata.
 */
IplImage* diminuisci(IplImage*,int);

/*
 * -Funzione per ridurre i disturbi in un'immagine-
 *  Il primo parametro indica l'immagine sorgente.
 *  Il secondo parametro indica l'immagine di destinazione.
 */
void riduciNoise(IplImage*,IplImage*);

/*
 * -Funzione per inserire un'immagine in un'altra immagine-
 *  Il primo parametro indica l'immagine da inserire.
 *  Il secondo parametro indica dove inserire l'immagine.
 *  Il terzo e il quarto parametro indicano le coordinate dove posizionare
l'immagine.
 */
void inserisci(IplImage*,IplImage*,int,int);
```

funzioni.c

```
/*
# This file is part of Computer Vision Exam Project
#
# Copyright(c) 2012 Domenico Luciani
# domenico.luciani@email.it
#
#
# This file may be licensed under the terms of of the
# GNU General Public License Version 2 (the ``GPL'').
#
# Software distributed under the License is distributed
# on an ``AS IS'' basis, WITHOUT WARRANTY OF ANY KIND, either
# express or implied. See the GPL for the specific language
```

```
# governing rights and limitations.
#
# You should have received a copy of the GPL along with this
# program. If not, go to http://www.gnu.org/licenses/gpl.html
# or write to the Free Software Foundation, Inc.,
# 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.
*/



typedef struct
{
    int H,S,V;
}HSV;

typedef struct
{
    int xmin,xmax;
    int ymin,ymax;
}Rettangolo;

void leggiConfig(HSV *m,HSV *n,char *file)
{
    FILE *config;
    config = fopen(file,"r");
    if(config != NULL)
        fscanf(config,"%d,%d,%d,%d,%d,%d",&m->H,&m->S,&m->V,&n->H,&n->S,&n->V);
    else
    {
        m->H = m->S = m->V = 0;
        n->H = n->S = n->V = 255;
    }

    printf("--- VALORI PRESI ---\n## MIN ##\nH->%d S->%d V->%d\n## MAX ##\nH->%d
S->%d V->%d\n",m->H,m->S,m->V,n->H,n->S,n->V);
    fclose(config);
}

void scriviConfig(HSV *m,HSV *n,char *file)
{
    FILE *config;
    config = fopen(file,"w");
    if(config != NULL)
    {
        fprintf(config,"%d,%d,%d,%d,%d,%d",m->H,m->S,m->V,n->H,n->S,n->V);
        puts("Valori scritti correttamente");
    }
    fclose(config);
}

IplImage* diminuisci(IplImage* file,int perc)
{
    IplImage *ris = cvCreateImage(cvSize((int)((file->width*perc)/100),(int)
((file->height*perc)/100)),8,file->nChannels);

    cvResize(file,ris);

    return ris;
```

```
}

void riduciNoise(IplImage *src,IplImage *dst)
{
    IplImage *buff = cvCreateImage(cvGetSize(src),8,dst->nChannels);
    cvDilate(src,buff,NULL,1);
    cvErode(buff,buff,NULL,2);
    cvSmooth(buff,dst,CV_GAUSSIAN,5);
    cvReleaseImage(&buff);
}
void inserisci(IplImage *small,IplImage *big,int pos1,int pos2)
{
    CvRect dim = cvRect(pos1,pos2,(int)small->width,(int)small->height);
    cvSetImageROI(big,dim);
    cvCopy(small,big);
    cvResetImageROI(big);
}
```

calibra.c

```
/*

# This file is part of Computer Vision Exam Project
#
# Copyright(c) 2012 Domenico Luciani
# domenico.luciani@email.it
#
#
# This file may be licensed under the terms of of the
# GNU General Public License Version 2 (the ``GPL'').
#
# Software distributed under the License is distributed
# on an ``AS IS'' basis, WITHOUT WARRANTY OF ANY KIND, either
# express or implied. See the GPL for the specific language
# governing rights and limitations.
#
# You should have received a copy of the GPL along with this
# program. If not, go to http://www.gnu.org/licenses/gpl.html
# or write to the Free Software Foundation, Inc.,
# 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.
*/

//Librerie necessarie
#include <cv.h>
#include <highgui.h>
#include "../lib/funzioni.h"
//File di configurazione
#define FILE_CONFIG "config.txt"
//Nomi delle GUI
#define NORMAL "Calibra"
#define BINARY "Calibra - Binaria"

int main(int argc,char *argv[])
{
```

```
    int web;
    //Controllo se i parametri sono corretti
    if(argc != 2)
        printf("usage: %s <mode>\n0 - integrate webcam\n1 - external
webcam",argv[0]);
    else
    {
        web = atoi(argv[1]);


        if(web >= 0 && web <= 1)
        {
            //Inizializzo la webcam
            CvCapture *capt = cvCaptureFromCAM(web);
            //Setto le proprietà della webcam a 640x480
            cvSetCaptureProperty(capt,CV_CAP_PROP_FRAME_WIDTH,640);
            cvSetCaptureProperty(capt,CV_CAP_PROP_FRAME_HEIGHT,480);
            //Prendo il primo frame dalla webcam e lo salvo
            IplImage *imm = cvQueryFrame(capt);
            //Creo immagini
            IplImage *hsv = cvCreateImage(cvGetSize(imm),8,3);
            IplImage *binary = cvCreateImage(cvGetSize(imm),8,1);

            //Variabili varie
            int i,j,step = binary->widthStep/sizeof(uchar);
            uchar *target = (uchar*)binary->imageData;
            char tasto;
            //Alloco spazio per i valori HSV
            HSV *low = (HSV*)malloc(sizeof(HSV));
            HSV *high = (HSV*)malloc(sizeof(HSV));
            //Alloco spazio per il rettangolo
            Rettangolo *punti = (Rettangolo*)malloc(sizeof(Rettangolo));
            //Leggo i dati dal file di configurazione
            leggiConfig(low,high,FILE_CONFIG);
            //Creo le GUI
            cvNamedWindow(NORMAL,CV_WINDOW_AUTOSIZE);
            cvNamedWindow(BINARY,CV_WINDOW_AUTOSIZE);
            //Ciclo while che si occuperà di prendere i frame
            while(imm)
            {
                //Creo le trackbar
                cvCreateTrackbar("HMIN",NORMAL,&low->H,255,NULL);
                cvCreateTrackbar("SMIN",NORMAL,&low->S,255,NULL);
                cvCreateTrackbar("VMIN",NORMAL,&low->V,255,NULL);

                cvCreateTrackbar("HMAX",NORMAL,&high->H,255,NULL);
                cvCreateTrackbar("SMAX",NORMAL,&high->S,255,NULL);
                cvCreateTrackbar("VMAX",NORMAL,&high->V,255,NULL);
               //Ruoto l'immagine
                cvFlip(imm,imm,1);
                //Converto in hsv
                cvCvtColor(imm,hsv,CV_BGR2HSV);
                //cerco il colore
                cvInRangeS(hsv,cvScalar(low->H,low->S,low->V),cvScalar(high-
>H,high->S,high->V),binary);
                //Riduco i disturbi
                riduciNoise(binary,binary);
                //Resetto il rettangolo
```

```
                *punti = {10000,0,10000,0};

                //Controllo pixel per pixel
                for(i=0; i < binary->height; i++)
                {
                    for(j=0; j < binary->width; j++)
                    {



                        if(target[i*step+j] == 255)
                        {
                            //I punti del nostro rettangolo
                            if(j < punti->xmin)
                                punti->xmin = j;
                            if(j > punti->xmax)
                                punti->xmax = j;
                            if(i < punti->ymin)
                                punti->ymin = i;
                            if(i > punti->ymax)
                                punti->ymax = i;
                        }
                    }
                }
                //Creo il rettangolo
                cvRectangle(imm,cvPoint(punti->xmin,punti->ymin),cvPoint(punti-
>xmax,punti->ymax),CV_RGB(255,0,0),2,8,0);
               //Creo il baricentro
                cvCircle(imm,cvPoint((punti->xmin+punti->xmax)/2,(punti-
>ymin+punti->ymax)/2),2,CV_RGB(0,0,255),-1,CV_AA,0);
                //Mostro le GUI
                cvShowImage(NORMAL,imm);
                cvShowImage(BINARY,binary);
                //Aspetto 15 millisecondi
                tasto = cvWaitKey(15);

                switch(tasto)
                {
                    //Se premo 'q' esco
                    case 'q':
                        return 2;
                        break;
                    case 's':
                        //Se premo 's' salvo i dati che mi servono
                        scriviConfig(low,high,FILE_CONFIG);
                        break;
                }
                //Prendo un altro frame
                imm = cvQueryFrame(capt);
            }
            //Libero tutto ciò che ho allocato in precedenza
            cvReleaseImage(&imm);
            cvReleaseImage(&binary);
            cvReleaseImage(&hsv);
            cvReleaseCapture(&capt);
        }
        else
            puts("webcam not found");
```

```
    }

    return 0;
}
```

bubbles.c

```
/*
# This file is part of Computer Vision Exam Project
#
# Copyright(c) 2012 Domenico Luciani
# domenico.luciani@email.it
#
#
# This file may be licensed under the terms of of the
# GNU General Public License Version 2 (the ``GPL'').
#
# Software distributed under the License is distributed
# on an ``AS IS'' basis, WITHOUT WARRANTY OF ANY KIND, either
# express or implied. See the GPL for the specific language
# governing rights and limitations.
#
# You should have received a copy of the GPL along with this
# program. If not, go to http://www.gnu.org/licenses/gpl.html
# or write to the Free Software Foundation, Inc.,
# 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.
*/
//Librerie necessarie al funzionamento
#include <cv.h>
#include <highgui.h>
#include <unistd.h>
#include <pthread.h>
#include <time.h>
//Libreria con funzioni mie
#include "../lib/funzioni.h"

//Nome della Gui
#define NOME "Bubbles"
//File di configurazione
#define FILE_CONFIG "../config/config.txt"
//Numero massimo di bolle da creare
#define NUM_MAX_BOLLE 50
//File del punteggio
#define CLASSIFICA "punteggio.txt"
//Punteggio per definire i vari livelli
#define SECONDO_LIVELLO 100
#define TERZO_LIVELLO 500
#define QUARTO_LIVELLO 1000
//Variabile globale per il punteggio.
int punt = 0;
//variabile per indicare se ho preso il nemico
int preso = 0;
```

```
//Struttura della bolla
typedef struct
{
    //Mutex
    pthread_mutex_t mutex;
    //Coordinate e passo
    int X,Y,passo;
    //Se e' un nemico o meno
    int chi,web;
}Bolla;

//Funzione per dare nuovi valori alla bolla passata
void nuoviValori(Bolla *dato)
{
    dato->X = rand() % (500 - 0 + 1) + 0;
    dato->Y = 1;
    dato->passo = 1;
}

//Funzione per aggiornare con un ritardo la posizione delle bolle
void *up(void *var)
{
    Bolla *val = (Bolla*)var;

    //Inizio del ciclo che modifica i valori
    while(1)
    {
            //Aspetto dei millisecondi randomici
            usleep((rand() % (30000 - 800 + 1) + 800));
            //Blocco il mutex su questo thread
            pthread_mutex_lock(&val->mutex);
            if(preso != 1)
            {
            //Se la bolla arriva alla fine dello schermo ritorna su con nuovi
valori
            if(val->Y >= 475 )
            {
                //Se non sono un nemico diminuisco il punteggio
                if(val->chi != 1)
                    punt -= 10;
                else
                    punt += 10;

                nuoviValori(val);
            }
            else
                //Faccio camminare la bolla
                val->Y += val->passo;
            //Sblocco il mutex
```

```
            }
            pthread_mutex_unlock(&val->mutex);
    }
    //Distruggo il mutex alla fine del thread
    pthread_mutex_destroy(&val->mutex);
}



//Funzione che prende il frame ed esegue tutte le altre operazioni a schermo
void *video(void *stato)
{
    //Che webcam usare
    int *web = (int*)stato;
    //Indico da quale sorgente prendere i frame
    CvCapture *frame = cvCaptureFromCAM(*web);
        cvSetCaptureProperty(frame,CV_CAP_PROP_FRAME_WIDTH,640);
        cvSetCaptureProperty(frame,CV_CAP_PROP_FRAME_HEIGHT,480);
    //Inizializzo le immagini
    IplImage *img = cvQueryFrame(frame);
    IplImage *hsv = cvCreateImage(cvGetSize(img),8,3);
    IplImage *binary = cvCreateImage(cvGetSize(img),8,1);

    //Inizializzo le bolle
    Bolla *val[NUM_MAX_BOLLE];

    //File dei punteggi
    FILE *filetto;
    //Stringhe varie
    char tasto,messaggio[20];
    //Livello
    int livello=1;
    //Vari indici
    int k,i,j;
    //Numero di bolle fin'ora create
    int num=1;
    //Iteratori
    int champion,ciclo_in=0,ciclo_out=0;

    //Step per l'immagine
    int step = binary->widthStep/sizeof(uchar);

    uchar *target = (uchar*)binary->imageData;
    //Alloco spazio per prendere i valori HSV da usare
    HSV *low = (HSV*)malloc(sizeof(HSV));
    HSV *high = (HSV*)malloc(sizeof(HSV));
    //creo NUM_MAX_BOLLE thread
    pthread_t th[NUM_MAX_BOLLE];
    //Inizializzo le scritte
    CvFont scritta,avviso;
    //Imposto lo stile del font
    cvInitFont(&scritta,CV_FONT_HERSHEY_SIMPLEX,.6,.6,0,1,6);
    cvInitFont(&avviso,CV_FONT_HERSHEY_SIMPLEX,1.0,1.0,0,5,CV_AA);
    //Creo una Gui chiamata NOME
    cvNamedWindow(NOME,1);

    srand((unsigned)time(NULL));
```

```
    for(k=0; k < NUM_MAX_BOLLE; k++)
        val[k] = (Bolla*)malloc(sizeof(Bolla));
    //Prima Bolla
    pthread_mutex_init(&val[0]->mutex,NULL);
    nuoviValori(val[0]);
    val[0]->chi = 0;
    //Creo il thread
    pthread_create(&th[0],NULL,&up,(void*)val[0]);

    //Prendo i dati dal file di config
    leggiConfig(low,high,FILE_CONFIG);

    filetto=fopen(CLASSIFICA,"r");
    if(filetto == NULL)
    {
        fclose(filetto);
        filetto = fopen(CLASSIFICA,"a");
        champion = 0;
    }
    else
        fscanf(filetto,"%d",&champion);

    fclose(filetto);

    //Prende i frame
    while(img)
    {
        //Ruoto l'immagine
        cvFlip(img,img,1);
        //Converto l'immagine da RGB a HSV
        cvCvtColor(img,hsv,CV_BGR2HSV);
        //cerco il mio colore
        cvInRangeS(hsv,cvScalar(low->H,low->S,low->V),cvScalar(high->H,high-
>S,high->V),binary);
        //Riduco i disturbi
        riduciNoise(binary,binary);
        //Ciclo per far comparire la scritta START
        if(ciclo_in >= 0 && ciclo_in <= 9)
        {
            cvPutText(img,"START",cvPoint((binary->width/2)-50,binary-
>height/2),&avviso,CV_RGB(255,50,60));
            ciclo_in++;
        }

        //Cerco in tutta l'immagine pixel per pixel
        //Altezza
        for(i=0; i < binary->height; i++)
        {
            //Larghezza
            for(j=0; j < binary->width; j++)
            {
                //Se trovo il colore bianco
                if(target[i*step+j] == 255)
                {
                    //Controllo per tutte le bolle che ho fin'ora
                    for(k=0; k < num; k++)
                    {
                        //Se tocco una bolla
```

```
                        if(j >= val[k]->X && j <= val[k]->X+10 && i >= val[k]->Y
&& i <= val[k]->Y+10)
                        {
                            //Se tocco un nemico
                            if(val[k]->chi == 1)
                                preso = 1;



                           else
                           {

                               pthread_mutex_lock(&val[k]->mutex);
                              if(preso != 1)
                              {
                                  punt += 10;
                                   //Scrivo +10 sopra la bolla
                                   cvPutText(img,"+10",cvPoint(val[k]->X,val[k]-
>Y-10),&scritta,CV_RGB(0,0,255));
                                   nuoviValori(val[k]);
                              }
                                   pthread_mutex_unlock(&val[k]->mutex);
                           }
                        }

                    }

                }

            }
        }

        //Se Tocco un nemico esco
        if(preso == 1)
        {
           if(ciclo_out >= 0 && ciclo_out <= 50)
           {

                //Visualizzo messaggi vari
                cvPutText(img,"GAME OVER",cvPoint((binary->width/2)-50,binary-
>height/2),&avviso,CV_RGB(0,0,255));
                sprintf(messaggio,"Punteggio: %d ",punt);
                cvPutText(img,messaggio,cvPoint((binary->width/2)-110,(binary-
>height/2)+80),&avviso,CV_RGB(0,0,255));

                sprintf(messaggio,"Livello: %d ",livello);
                cvPutText(img,messaggio,cvPoint(img->width-
100,20),&scritta,CV_RGB(255,255,0));
                sprintf(messaggio,"Record: %d ",champion);
                cvPutText(img,messaggio,cvPoint(img->width-150,img->height-
5),&scritta,CV_RGB(255,255,0));
                ciclo_out++;
           }
           if(punt > champion)
           {
               puts("NUOVO RECORD!");
               filetto = fopen(CLASSIFICA,"w");
               fprintf(filetto,"%d",punt);
```

```
                fclose(filetto);
                puts("Record salvato!");
                champion = punt;
            }

            if(ciclo_out == 50)
                 break;
        }

        else
        {
            //Per ogni bolla
            for(k=0; k < num; k++)
            {
                //Disegno un nemico
                if(val[k]->chi == 1)
                    cvCircle(img,cvPoint(val[k]->X,val[k]-
>Y),10,CV_RGB(0,0,255),20,8);
                else
                    //Disegno una bolla
                    cvCircle(img,cvPoint(val[k]->X,val[k]-
>Y),10,CV_RGB(255,0,0),20,8);
            }
            sprintf(messaggio,"Punteggio: %d ",punt);
            cvPutText(img,messaggio,cvPoint(20,20),&scritta,CV_RGB(255,255,0));
            sprintf(messaggio,"Livello: %d ",livello);
            cvPutText(img,messaggio,cvPoint(img->width-
100,20),&scritta,CV_RGB(255,255,0));
            sprintf(messaggio,"Record: %d ",champion);
            cvPutText(img,messaggio,cvPoint(img->width-150,img->height-
5),&scritta,CV_RGB(255,255,0));

            //Se aumenta il punteggio aumento il livello
            switch(punt)
            {
                case 100:
                    if(livello == 1)
                    {
                        //Creo altre bolle incrementando la difficoltà
                        for(k=num; k < 10; k++)
                        {
                            pthread_mutex_init(&val[k]->mutex,NULL);
                            nuoviValori(val[k]);
                            //Creo nemici
                            if(k <= 2)
                                val[k]->chi = 1;
                            else
                                val[k]->chi = 0;

                            pthread_create(&th[k],NULL,&up,(void*)val[k]);
                        }
                        num = 10;
                        livello++;
                    }
                    break;
                case 500:
                    if(livello == 2)
                    {
```

```
            for(k=num; k < 20; k++)
            {
                pthread_mutex_init(&val[k]->mutex,NULL);
                nuoviValori(val[k]);
                if(k <= 15)
                    val[k]->chi = 1;



                   else
                    val[k]->chi = 0;

                pthread_create(&th[k],NULL,&up,(void*)val[k]);
            }
            num = 20;
            livello++;
        }
        break;
    case 1000:
        if(livello == 3)
        {
            for(k=num; k < 30; k++)
            {
                pthread_mutex_init(&val[k]->mutex,NULL);
                nuoviValori(val[k]);
                if(k <= 25)
                    val[k]->chi = 1;
                else
                    val[k]->chi = 0;

                pthread_create(&th[k],NULL,&up,(void*)val[k]);
            }
            num = 30;
            livello++;
        }
        break;
    case 1500:
        if(livello == 4)
        {
            for(k=num; k < 40; k++)
            {
                pthread_mutex_init(&val[k]->mutex,NULL);
                nuoviValori(val[k]);
                if(k <= 35)
                    val[k]->chi = 1;
                else
                    val[k]->chi = 0;

                pthread_create(&th[k],NULL,&up,(void*)val[k]);
            }
            num = 40;
            livello++;
        }
        break;
    case 2000:
        if(livello == 5)
        {
            for(k=num; k < 49; k++)
```

```
                        {
                            pthread_mutex_init(&val[k]->mutex,NULL);
                            nuoviValori(val[k]);
                            if(k <= 45)
                                val[k]->chi = 1;
                            else
                                val[k]->chi = 0;

                            pthread_create(&th[k],NULL,&up,(void*)val[k]);
                        }
                        num = 49;
                        livello++;
                    }
                    break;
            }
        }

        cvShowImage(NOME,img);

        //Aspetto il tasto
        tasto = cvWaitKey(15);
        //Se premo q esco
        if(tasto == 'q')
            break;
        //Prendo l'altro frame
        img = cvQueryFrame(frame);
    }

    //Distruggo tutti i mutex
    for(k=0; k < num; k++)
        pthread_mutex_destroy(&val[k]->mutex);

    //Distruggo tutto
    cvReleaseImage(&img);
    cvReleaseCapture(&frame);
}

//Main
int main(int argc,char *argv[])
{
    //thread
    pthread_t th1;
    int web;

    if(argc != 2)
        printf("usage: %s <mode>\n0 - integrate webcam\n1 - external
webcam\n",argv[0]);
    else
    {
        web = atoi(argv[1]);
        if(web >= 0 && web <= 1)
        {
            printf( "Bubbles Game v. 1.0\n"
                    "Progettato e realizzato da Domenico Luciani\n"
                    "Classe 5B Informatica Abacus\n"
                    "dell'Istituto Tecnico Industriale Vittorio Emanuele III\n"
                    "Passare 0 come parametro per usare la webcam integrata\n"
                    "Passare 1 come parametro per usare la webcam esterna\n"
```

```
                    "Legenda:\n"
                    "1 - Ogni pallina rossa presa fa aumentare il punteggio di 10
punti\n"
                    "2 - Se viene presa una pallina blu si perde
istantaneamente\n"
                    "3 - Ogni pallina rossa persa fa diminuire il punteggio di 10
punti\n"
                    "4 - All'aumentare del livello aumenta anche la difficoltà\n"
                    "Buon divertimento\n");
            //Creo un nuovo thread
            pthread_create(&th1,NULL,&video,(void*)&web);
            //Il processo aspetta che lo streaming video smetta di esistere
            pthread_join(th1,NULL);
        }
        else
            puts("webcam not found");
    }
    return 0;
}
```

draw.c

```
/*
# This file is part of Computer Vision Exam Project
#
# Copyright(c) 2012 Domenico Luciani
# domenico.luciani@email.it
#
#
# This file may be licensed under the terms of of the
# GNU General Public License Version 2 (the ``GPL'').
#
# Software distributed under the License is distributed
# on an ``AS IS'' basis, WITHOUT WARRANTY OF ANY KIND, either
# express or implied. See the GPL for the specific language
# governing rights and limitations.
#
# You should have received a copy of the GPL along with this
# program. If not, go to http://www.gnu.org/licenses/gpl.html
# or write to the Free Software Foundation, Inc.,
# 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.
*/

//Librerie necessarie
#include <cv.h>
#include <highgui.h>
#include "../lib/funzioni.h"

//Nome della Gui
#define NOME "Draw"
//File di configurazione
#define FILE_CONFIG "../config/config.txt"

//main
int main(int argc, char *argv[])
{
    if(argc != 2)
```

```
    {
        printf("usage: %s <mode>\n0 - integrate webcam\n1 - external
webcam\n",argv[0]);
        exit(-1);
    }
    else
    {
        int web=atoi(argv[1]);

        if(web >= 0 && web <= 1)
        {
            CvCapture *frame = cvCaptureFromCAM(web);
            cvSetCaptureProperty(frame,CV_CAP_PROP_FRAME_WIDTH,640);
            cvSetCaptureProperty(frame,CV_CAP_PROP_FRAME_HEIGHT,480);
            //Immagini
            IplImage *img = cvQueryFrame(frame);
            IplImage *hsv = cvCreateImage(cvGetSize(img),8,3);
            IplImage *binary = cvCreateImage(cvGetSize(img),8,1);
            IplImage *buffer = cvCreateImage(cvGetSize(img),8,3);

            int step = binary->widthStep/sizeof(uchar);

            int i,j,capo=0,XX,YY;
            char tasto;

            uchar *target = (uchar*)binary->imageData;
           //Valori
            HSV *low = (HSV*)malloc(sizeof(HSV));
            HSV *high = (HSV*)malloc(sizeof(HSV));
            //Il rettangolo
            Rettangolo *punti = (Rettangolo*)malloc(sizeof(Rettangolo));

            CvPoint pos,last;
            //Colore
            CvScalar colore = CV_RGB(191,255,255);
            //Creo la Gui
            cvNamedWindow(NOME,1);
            //Leggo i dati
            leggiConfig(low,high,FILE_CONFIG);

            while(img)
            {
                //Ruoto l'immagine
                cvFlip(img,img,1);
                //Converto l'immagine da RGB a HSV
                cvCvtColor(img,hsv,CV_BGR2HSV);
                //cerco il mio colore
                cvInRangeS(hsv,cvScalar(low->H,low->S,low->V),cvScalar(high-
>H,high->S,high->V),binary);
                //Riduco i disturbi
                riduciNoise(binary,binary);

                punti->xmin = 10000;
                punti->xmax = 0;
                punti->ymin = 10000;
                punti->ymax = 0;

                //Rettangolo rosso
```

```
cvRectangle(img,cvPoint(0,0),cvPoint(100,60),CV_RGB(255,0,0),CV_FILLED,8,0);
                //Rettangolo verde

cvRectangle(img,cvPoint(200,0),cvPoint(300,60),CV_RGB(0,255,0),CV_FILLED,8,0);
                //Rettangolo blu

cvRectangle(img,cvPoint(400,0),cvPoint(500,60),CV_RGB(0,0,255),CV_FILLED,8,0);

                //Controllo l'immagine pixel per pixel
                for(i=0; i < binary->height; i++)
                {
                    for(j=0; j < binary->width; j++)
                    {
                        //Se trovo il colore
                        if(target[i*step+j] == 255)
                        {
                            //Coordinate del mio rettangolo
                            if(j < punti->xmin)
                                punti->xmin = j;
                            if(j > punti->xmax)
                                punti->xmax = j;
                            if(i < punti->ymin)
                                punti->ymin = i;
                            if(i > punti->ymax)
                                punti->ymax = i;
                            //Baricentro del mio rettangolo
                            XX = (punti->xmin+punti->xmax)/2;
                            YY = (punti->ymin+punti->ymax)/2;

                            //Se entro dentro il rettangolo rosso
                            if(XX >= 0 && XX <= 100 && YY >= 0 && YY <= 60)
                            {
                                colore = CV_RGB(255,0,0);
                                capo = 1;
                            }
                            //Se entro dentro il rettangolo verde
                            else if((XX >= 200 && XX <= 300) && (YY >= 0 && YY <=
60))
                            {
                                colore = CV_RGB(0,255,0);
                                capo = 1;
                            }
                            //Se entro dentro il rettangolo blu
                            else if((XX >= 400 && XX <= 500) && (YY >= 0 && YY <=
60))
                            {
                                colore = CV_RGB(0,0,255);
                                capo = 1;
                            }
                        }
                    }

                }
                //Se ho toccato un rettangolo posso scrivere
                last = cvPoint(XX,YY);
                if(capo == 1)
                {
```

```
                    //Creo il cerchio nell'immagine buffer
                    cvCircle(buffer,last,20,colore,-1,CV_AA,0);
                    //Faccio il merge fra il frame e il buffer
                    cvAdd(img,buffer,img,NULL);
                }
                else
                //Disegno il cerchio direttamente nel frame in modo da usarlo come
puntatore
                   cvCircle(img,last,20,CV_RGB(159,200,120),-1,CV_AA,0);

                //Mostro la gui
                cvShowImage(NOME,img);
                //Prendo il tasto premuto
                tasto = cvWaitKey(15);

                //Se premo q esco
                if(tasto == 'q')
                    return 0;
                //Se premo C cancello ogni cosa
                else if(tasto == 'c')
                {
                    cvReleaseImage(&buffer);
                    buffer = cvCreateImage(cvGetSize(img),8,3);
                    capo = 0;
                }
                img = cvQueryFrame(frame);
            }
            cvReleaseImage(&buffer);
            cvReleaseImage(&img);
            cvReleaseImage(&binary);
            cvReleaseImage(&hsv);
            cvReleaseCapture(&frame);
        }
        else
            puts("webcam not found\n");
    }
    return 0;
}
```

head.c

```
/*
# This file is part of Computer Vision Exam Project
#
# Copyright(c) 2012 Domenico Luciani
# domenico.luciani@email.it
#
#
# This file may be licensed under the terms of of the
# GNU General Public License Version 2 (the ``GPL'').
#
# Software distributed under the License is distributed
# on an ``AS IS'' basis, WITHOUT WARRANTY OF ANY KIND, either
# express or implied. See the GPL for the specific language
# governing rights and limitations.
#
```

```
# You should have received a copy of the GPL along with this
# program. If not, go to http://www.gnu.org/licenses/gpl.html
# or write to the Free Software Foundation, Inc.,
# 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.
*/

//Header
#include <cv.h>
#include <highgui.h>
#include "../lib/funzioni.h"

//File dove prendo le informazioni per il riconoscimento del volto
#define FILE_FACCIA_1 "haarcascade_frontalface_alt.xml"

#define FILE_OCCHIO_DESTRO "haarcascade_mcs_righteye.xml"
#define FILE_OCCHIO_SINISTRO "haarcascade_mcs_lefteye.xml"

//Nome della finestra
#define NOME_FINESTRA1 "Head tracking"
//Indica al programma se rilevare anche gli occhi, il naso e la bocca. (0 = no, 1
= si')
#define RILEVA_EXTRA 0

void rilevaDettagli(IplImage *img,char *file)
{
    //Faccio spazio e prendo le informazioni dal file passato
    CvMemStorage *spazio = cvCreateMemStorage(0);
    CvHaarClassifierCascade *cascade =
(CvHaarClassifierCascade*)cvLoad(file,spazio);
    // Analizzo gli elementi
    CvSeq *elementi =
cvHaarDetectObjects(img,cascade,spazio,1.3,1,CV_HAAR_DO_CANNY_PRUNING,cvSize(40,40
));
    //Controllo se ci sono elementi
    if(elementi->total > 0)
    {
        int i;
        //Creo i rettangoli
        for(i=0; i < elementi->total; i++)
        {
            //Prendo gli elementi
            CvRect *rettangolo = (CvRect*)cvGetSeqElem(elementi,i);
            //Creo i rettangoli di rosso nelle zone degli elementi
            cvRectangle(img,cvPoint(rettangolo->x,rettangolo-
>y),cvPoint(rettangolo->x+rettangolo->width,rettangolo->y+rettangolo-
>height),CV_RGB(255,0,0),1,8,0);
        }
    }
    //Mostro i rettangoli appena disegnati
    cvShowImage(NOME_FINESTRA1,img);
}


//Funzione per il rilevamento del volto, passiamo l'immagine, il file cascade e lo
spazio da utilizzare
void rilevaVolto(IplImage *img,CvHaarClassifierCascade *cascade, CvMemStorage
*spazio)
{
```

```
    //Analizzo gli elementi
    CvSeq *volti =
cvHaarDetectObjects(img,cascade,spazio,1.1,3,CV_HAAR_DO_CANNY_PRUNING,cvSize(40,40
));
    //Controllo se ci sono volti
    if(volti->total > 0)
    {
        int i;

            //Creo i rettangoli
            for(i=0; i < volti->total; i++)
            {
                //Prendo i volti trovati
                CvRect *rettangolo = (CvRect*)cvGetSeqElem(volti,i);
                //Se voglio cercare anche i dettagli
                if(RILEVA_EXTRA == 1)
                {
                    //Faccio il focus solo sulla faccia trovata escludendo il
resto
                    cvSetImageROI(img,*rettangolo);
                    //Rilevo l'occhio sinistro
                    rilevaDettagli(img,FILE_OCCHIO_SINISTRO);
                    //Rilevo l'occhio destro
                    rilevaDettagli(img,FILE_OCCHIO_DESTRO);
                    //Faccio ritornare il focus sull'immagine normale
                    cvResetImageROI(img);
                }
                //Disegno il rettangolo del volto
                cvRectangle(img,cvPoint(rettangolo->x,rettangolo-
>y),cvPoint(rettangolo->x+rettangolo->width,rettangolo->y+rettangolo-
>height),CV_RGB(50,205,50),3,8,0);
            }
    }
}

int main(int argc,char **argv)
{
    if(argc != 2)
    {
        printf("usage: %s <mode>\n0 - integrate webcam\n1 - external
webcam\n",argv[0]);
        exit(-1);
    }
    else
    {
        int web=atoi(argv[1]);
        if(web >= 0 && web <= 1)
        {
            //Questo è il frame
            IplImage *frame,*ridotta;
            //Immagine dalla cam
            CvCapture *immagine;
            //Creo uno spazio di memoria
            CvMemStorage *spazio=cvCreateMemStorage(0);
            //Carico il file cascade -IMPORTANTE PER IL FUNZIONAMENTO DEL
TRACKING-
            CvHaarClassifierCascade *cascade = (CvHaarClassifierCascade*)
cvLoad(FILE_FACCIA_1,spazio);
```

```
            //Prendo i dati dalla cam
            immagine = cvCaptureFromCAM(web);

            cvSetCaptureProperty(immagine,CV_CAP_PROP_FRAME_WIDTH,640);
            cvSetCaptureProperty(immagine,CV_CAP_PROP_FRAME_HEIGHT,480);

            //Salvo il frame
            frame = cvQueryFrame(immagine);

            //Creo la finestra
            cvNamedWindow(NOME_FINESTRA1,CV_WINDOW_AUTOSIZE);

            //Ciclo finchè ci sono frame
            while(frame)
            {
                cvFlip(frame,frame,1);
                frame = diminuisci(frame,50);
                //Rilevo il volto e i relativi dettagli
                rilevaVolto(frame,cascade,spazio);
                //Mostra l'immagine
                cvShowImage(NOME_FINESTRA1,frame);
                //Libero lo spazio
                cvClearMemStorage(spazio);
                //Prende un altro frame
                frame = cvQueryFrame(immagine);
                //Aspetta 10 ms per controllare se viene premuto il tasto q (per
uscire)
                char c = cvWaitKey(15);
                if(c == 'q') break;
            }
            cvReleaseHaarClassifierCascade(&cascade);
            cvReleaseCapture(&immagine);
            cvReleaseImage(&frame);
            cvDestroyWindow(NOME_FINESTRA1);
        }
        else
            puts("webcam not found");
    }
    return 0;
}
```

movement.c

```
/*
# This file is part of Computer Vision Exam Project
#
# Copyright(c) 2012 Domenico Luciani
# domenico.luciani@email.it
#
#
# This file may be licensed under the terms of of the
# GNU General Public License Version 2 (the ``GPL'').
#
# Software distributed under the License is distributed
# on an ``AS IS'' basis, WITHOUT WARRANTY OF ANY KIND, either
# express or implied. See the GPL for the specific language
```

```
# governing rights and limitations.
#
# You should have received a copy of the GPL along with this
# program. If not, go to http://www.gnu.org/licenses/gpl.html
# or write to the Free Software Foundation, Inc.,
# 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.
*/



//Librerie utili
#include <cv.h>
#include <highgui.h>
#include <time.h>
#include "../lib/funzioni.h"
//Nome della gui
#define NOME "Control"

int main(int argc,char **argv)
{
    if(argc != 2)
    {
        printf("usage: %s <mode>\n0 - integrate webcam\n1 - external
webcam\n",argv[0]);
        exit(-1);
    }
    else
    {
        int web=atoi(argv[1]);
        if(web >= 0 && web <= 1)
        {
            CvCapture *cam = cvCaptureFromCAM(web);
            cvSetCaptureProperty(cam,CV_CAP_PROP_FRAME_WIDTH,640);
            cvSetCaptureProperty(cam,CV_CAP_PROP_FRAME_HEIGHT,480);
            IplImage *img = cvQueryFrame(cam);
            IplImage *copia = cvCreateImage(cvGetSize(img),8,3);
            IplImage *prima = NULL;
            IplImage *binary = cvCreateImage(cvGetSize(img),8,1);
            IplImage *ris = cvCreateImage(cvGetSize(img),8,3);

            cvNamedWindow(NOME,1);
            //Variabili per prendere l'orario e la data correnti
            time_t tempo;
            struct tm *timeobj;
            time(&tempo);
            timeobj = localtime(&tempo);

            char nome[25];
            long int num=0;
            //Funzione per inserire i dati del tempo in nome
            strftime(nome,24,"%H-%M-%S_%F.avi",timeobj);
            //Creo il writer che si occuperà di scrivere i vari frame presi come
video compresso in formato divx
            CvVideoWriter *video =
cvCreateVideoWriter(nome,CV_FOURCC('D','I','V','X'),15,cvSize(640,480),1);
            //Inizializzo i font
            CvFont scritta,info;
            cvInitFont(&scritta,CV_FONT_HERSHEY_SIMPLEX,1.0,1.0,0,5,CV_AA);
```

```
            cvInitFont(&info,CV_FONT_HERSHEY_SIMPLEX,.6,.6,0,1,6);

            char tasto;
            int i,j,trovato=0,scelta,step = binary->widthStep/sizeof(uchar);
            uchar *target = (uchar*)binary->imageData;
            //Scelta fra dinamica e statica
            do
            {

printf("-- Scelta modalita' --\n1)Dinamica -- Se ci saranno variazioni tra un
frame e l'altro\n2)Statica -- Se ci sono variazioni fra un determinato frame e il
frame corrente\nScelta: ");
                scanf("%1d",&scelta);
            }while(scelta < 1 || scelta > 2);

            while(img)
            {
                //Ruoto l'immagine
                cvFlip(img,img,1);
                //Prendo le informazioni sul tempo
                time(&tempo);
                timeobj = localtime(&tempo);
                strftime(nome,24,"%H:%M:%S %F",timeobj);
                //Scrivo le info a schermo
                cvPutText(img,nome,cvPoint(415,475),&info,CV_RGB(0,255,255));
                //Copio il frame
                cvCopy(img,copia);

                riduciNoise(img,img);
                //Dinamica
                if(scelta == 1)
                {
                    //Se è il primo frame preso
                    if(prima == NULL)
                    {
                        prima = cvCreateImage(cvGetSize(img),8,3);
                        //Copio img in prima
                        cvCopy(img,prima);
                    }
                    else
                    {
                        //Se non è il primo frame controllo se ci sono differenze
                        cvAbsDiff(img,prima,ris);
                        //Da colore a grigia
                        cvCvtColor(ris,binary,CV_BGR2GRAY);
                        //Il threshold dell'immagine
                        cvThreshold(binary,binary,62,255,CV_THRESH_BINARY);
                        riduciNoise(binary,binary);
                        cvCopy(img,prima);
                    }

                }
                //Statica
                else
                {
                    //Se ho preso il frame da monitorare
                    if(prima != NULL)
                    {
```

```
                        cvAbsDiff(img,prima,ris);
                        cvCvtColor(ris,binary,CV_BGR2GRAY);
                        cvThreshold(binary,binary,62,255,CV_THRESH_BINARY);
                        riduciNoise(binary,binary);

                    }

                }

                //Controllo l'immagine pixel per pixel
                for(i=0; i < binary->height; i++)
                {
                    for(j=0; j < binary->width; j++)
                    {
                        if(target[i*step+j] == 255)
                            trovato = 1;
                    }
                }

                //Se trovo un cambiamento
                if(trovato)
                {
                    num++;
                    //Inserisco "REC O" nell'immagine

cvPutText(copia,"REC",cvPoint(10,25),&scritta,CV_RGB(255,0,0));
                    cvCircle(copia,cvPoint(100,15),5,CV_RGB(255,0,0),20,8);
                    //Salvo il frame trovato
                    cvWriteFrame(video,copia);
                    trovato = 0;
                }
                //Mostro l'immagine
                cvShowImage(NOME,copia);

                tasto = cvWaitKey(15);

                if(tasto == 'q')
                    break;
                //Se premo v salvo il frame da monitorare
                else if(tasto == 'v' && scelta == 2)
                {
                    prima = cvCreateImage(cvGetSize(img),8,3);
                    cvCopy(img,prima);
                }

                img = cvQueryFrame(cam);
            }
            //Se ho preso dei frame
            if(num != 0)
            {
                //Scrivo il video
                cvReleaseVideoWriter(&video);
                printf("Video %s salvato\n",nome);
            }
        }
        else
            puts("webcam not found");
    }
```

```
    return 0;
}
```

LICENSE

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will

not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.
TERMS AND CONDITIONS
0. Definitions.

?This License? refers to version 3 of the GNU General Public License.

?Copyright? also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

?The Program? refers to any copyrightable work licensed under this License. Each licensee is addressed as ?you?. ?Licensees? and ?recipients? may be individuals or organizations.

To ?modify? a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a ?modified version? of the earlier work or a work ?based on? the earlier work.

A ?covered work? means either the unmodified Program or a work based on the Program.

To ?propagate? a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To ?convey? a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays ?Appropriate Legal Notices? to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.
1. Source Code.

The ?source code? for a work means the preferred form of the work for making modifications to it. ?Object code? means any non-source form of a work.

A ?Standard Interface? means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The ?System Libraries? of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A ?Major Component?, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The ?Corresponding Source? for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.
2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.
3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under

any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

* a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
* b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to ?keep intact all notices?.
* c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
* d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an ?aggregate? if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

* a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on

a durable physical medium customarily used for software interchange.

* b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

* c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

* d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

* e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A ?User Product? is either (1) a ?consumer product?, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, ?normally used? refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

?Installation Information? for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object

code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.
7. Additional Terms.

?Additional permissions? are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

    * a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
    * b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
    * c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
    * d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
    * e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
    * f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered ?further restrictions? within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such

relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.
8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.
9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.
10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An ?entity transaction? is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted

or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.
11. Patents.

A ?contributor? is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's ?contributor version?.

A contributor's ?essential patent claims? are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, ?control? includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a ?patent license? is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To ?grant? such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. ?Knowingly relying? means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is ?discriminatory? if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in

connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License ?or any later version? applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM ?AS IS? WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU

ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS
How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the ?copyright? line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year> <name of author>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

    <program> Copyright (C) <year> <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts

of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an ?about box?.

You should also get your employer (if you work as a programmer) or school, if any, to sign a ?copyright disclaimer? for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

Domenico Luciani

Classe 5°B
Informatica Abacus
Anno 2012/13