

P
oo

Attestato di partecipazione
al corso:

Facial Tracking con OpenCV Processing

Linee guida per il riconoscimento facciale.

Rilasciato a

_____ il 10 Luglio 2015

Durata: 4 ore

Sede: SPQwoRk Roma, via di Portinaccio 23/B

Corso tenuto da:
dott. Luca R. Pappa.

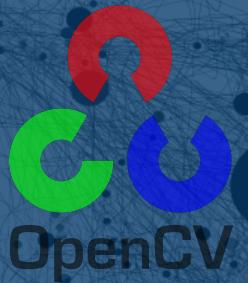


Facial Tracking con OpenCV Processing

Linee guida per il riconoscimento facciale.

Luca Rosario Pappa





1. Introduzione
 - a. Cos'è Processing
 - b. Perché lo usiamo
 - c. Le Basi
2. Primi passi
 - a. Configurazioni di base
 - b. Libreria OpenCV
 - c. Classificatori
 - d. Play with demos
3. Creiamo il nostro primo sketch
 - a. Face Detection
 - b. Eyes Detection
 - c. Mouth Detection
 - d. Background substraction
 - e. Optical Flow
4. Fai da te
 - a. Ora tocca a te
 - b. Creazione di applicazione



Introduzione. Cos'è Processing.

Processing è un linguaggio di programmazione che consente di sviluppare diverse applicazioni come giochi, animazioni e contenuti interattivi. Eredita completamente la sintassi, i comandi e il paradigma di programmazione orientata agli oggetti dal linguaggio Java ma in più mette a disposizione numerose funzioni ad alto livello per gestire facilmente gli aspetti grafici e multimediali.

È distribuito sotto la licenza libera *GNU General Public License* ed è supportato dai sistemi operativi Linux, Mac OS X e Microsoft Windows (e da poco anche iOS e Android). *vedi: Wiki.*

Introduzione. Perché usarlo.

- Versatile → Audio, Video, Arduino, Kinect ...
- Intuitivo → Funzionalità di supporto.
- Multi Piattaforma → Mac, Windows, Linux, Android.
- Vasta community → openProcessing.org,
learningProcessing.com, processing.org.

Introduzione. Le Basi.

Funzioni principali:

- **setup**: funzione che viene eseguita una sola volta. Viene utilizzata solitamente per inizializzare gli oggetti e viene chiamata all'inizio del programma.
- **draw**: funzione che viene eseguita ciclicamente per la durata del programma. Grazie alle funzioni noLoop() e loop(), si può disattivare o riattivare il ciclo.

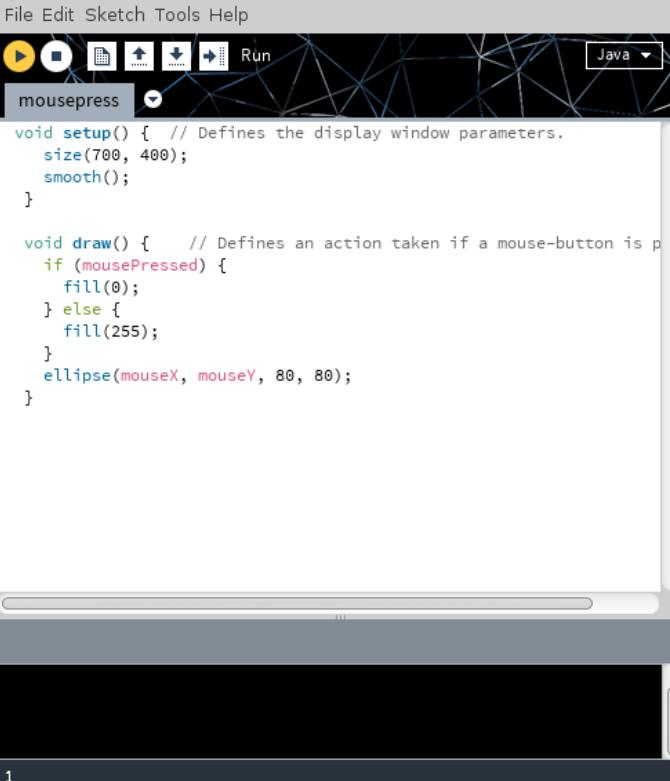
Ricorda: Gli oggetti dichiarati nella funzione setup non sono visibili nella draw, quindi, dichiara le variabili globalmente (all'esterno delle funzioni).

Primitive 2D:

- quad(x1, x2, x3, x4, y1, y2, y3, y4)
- rect(x1, y1, width, height)
- triangle(x1, x2, x3, y1, y2, y3)
- arc(a,b,c,d,strart,stop)
- ellipse(x1, x2, width, height)
- line(x1, x2, y1, y2)
- point(x1, y1)

Primi passi. Installazione.

- Download del software dal sito processing.org/download
- Non ha un installer, si esegue semplicemente l'IDE
- Si possono importare librerie esterne tramite il menu “Sketches→ Import Library→ Add Library”
- Ogni libreria ha degli esempi specifici
- Tutorial e references dal sito di processing.org



The screenshot shows the Processing IDE interface. At the top, there's a menu bar with "File", "Edit", "Sketch", "Tools", and "Help". Below the menu is a toolbar with icons for play, stop, save, and run. A dropdown menu shows "Java" selected. The main area contains a sketch window with a dark background. Inside the sketch window, there's a small white ellipse centered at the mouse position. At the bottom of the sketch window, the text "mousepress" is visible. The code area contains the following Java code:

```
File Edit Sketch Tools Help  
Java ▾  
  
void setup() { // Defines the display window parameters.  
    size(700, 400);  
    smooth();  
}  
  
void draw() { // Defines an action taken if a mouse-button is p  
    if (mousePressed) {  
        fill(0);  
    } else {  
        fill(255);  
    }  
    ellipse(mouseX, mouseY, 80, 80);  
}
```

The code uses Java syntax for Processing sketches. It defines a setup function that sets the window size to 700x400 pixels and enables smoothing. The draw function checks if the mouse is pressed; if it is, it fills the ellipse with black; otherwise, it fills it with white. The ellipse is drawn at the current mouse position with a diameter of 80 pixels.

Primi passi. classe Capture.

Funzioni della classe Capture:

available() Ritorna "true" quando un nuovo frame è disponibile alla lettura

start() Inizia a catturare frame dal device selezionato

stop() Stops capturing frames from an attached device

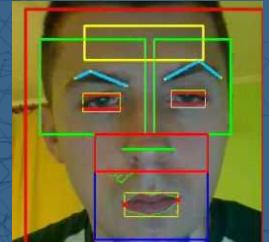
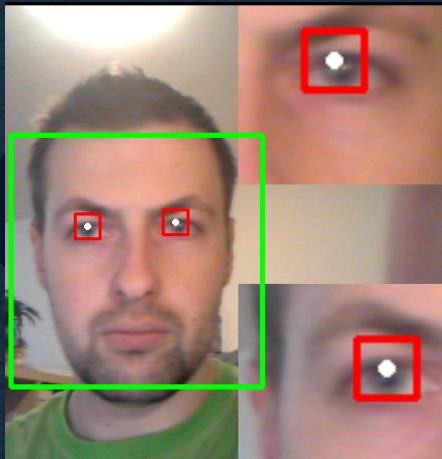
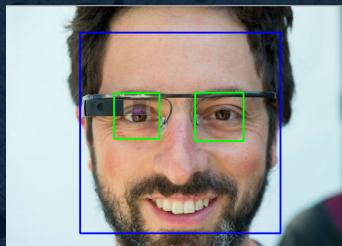
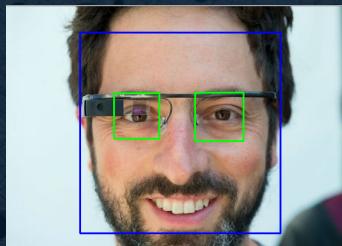
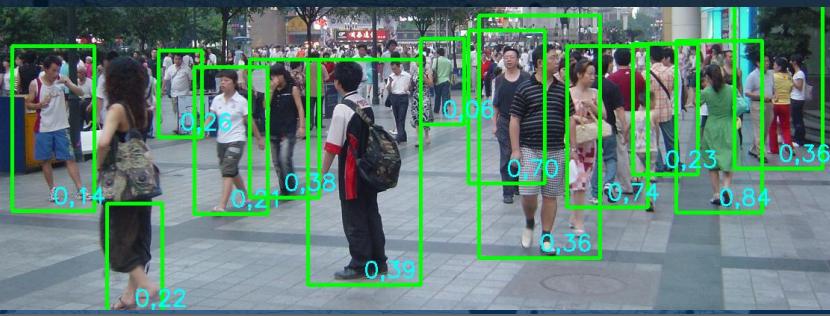
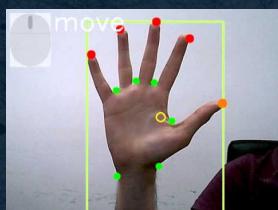
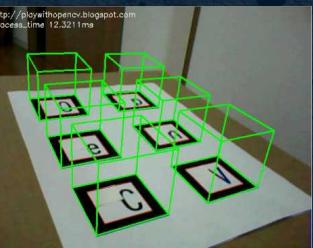
read() Reads the current video frame

list() Gets a list of all available capture devices such as a camera



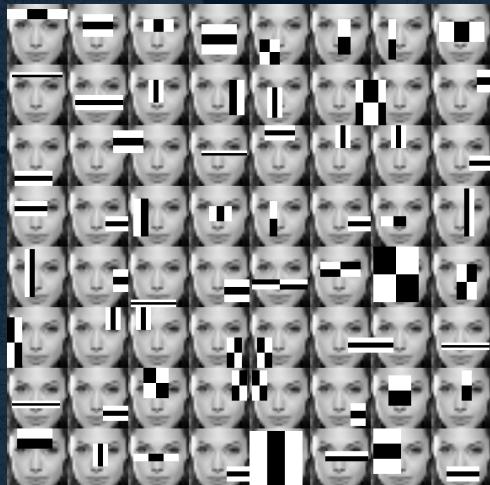
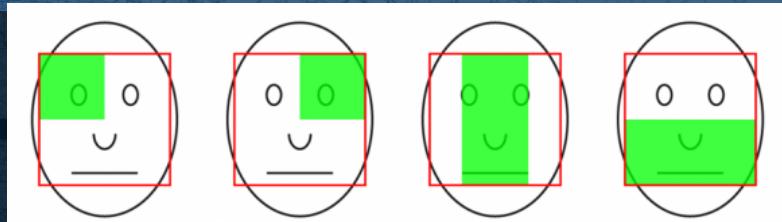
Primi passi. Libreria OpenCV.

- Cos'è OpenCV
- Come si interfaccia con Processing
- Come utilizzarlo
- I Classificatori
- Il riconoscimento facciale
- Il Tracking



Primi passi. Classificatori.

- Come fare a riconoscere un viso in un fotogramma?
- Quant fotogrammi al secondo passano sulla camera?
- Come fare a trovare un modo efficiente per riconoscere un viso così velocemente da farlo processare ad un pc ed una webcam?
- Haar-like Features, LPB
- Paul Viola and Michael J. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features



Primi passi. Classificatori.

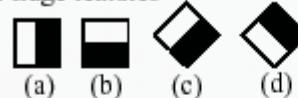
Le proprietà delle haar feature sono riassumibili in:

- La regione degli occhi è più scura delle guance.
- Il ponte che forma il naso è più chiaro degli occhi.

Grazie a queste features riusciamo a farci un'idea di:

- Posizioni - Grandezze: occhi e ponte creato dal naso
- Valori: scuro / chiaro

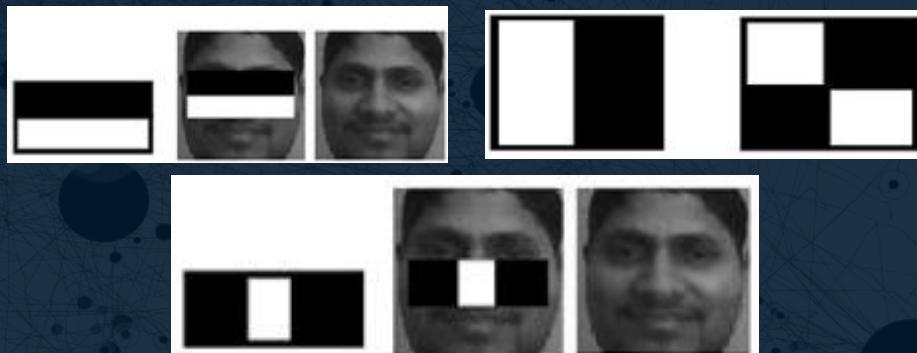
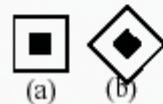
1. Edge features



2. Line features



3. Center-surround features



Primi passi. Algoritmo di Viola & Jones

Nelle prossime lezione potremmo capire come l' algoritmo di viola & jones funziona nel dettaglio

Le caratteristiche che rendono questo algoritmo efficiente ed efficace sono:

- Robusto – grazie all'alto tasso di riconoscimento (detto true-positive rate) e un basso tasso di falsi negativi.
- Real time – Per applicazioni reali vanno processato almeno due frame.
- Face detection – Ricordiamo che l'obiettivo dell'algoritmo è quello di dare risultati sulla presenza o meno di visi nell'immagine.

The algorithm has mainly 4 stages:

1. Haar Features Selection
2. Creating Integral Image
3. Adaboost Training algorithm
4. Cascaded Classifiers

$$h(\mathbf{x}) = \text{sign} \left(\sum_{j=1}^M \alpha_j h_j(\mathbf{x}) \right)$$

$$h_j(\mathbf{x}) = \begin{cases} -s_j & \text{if } f_j < \theta_j \\ s_j & \text{otherwise} \end{cases}$$

Training di un classificatore ad-hoc

Nelle prossime lezione potremmo capire come creare un classificatore custom per qualsiasi tipo di immagine si voglia apprendere.

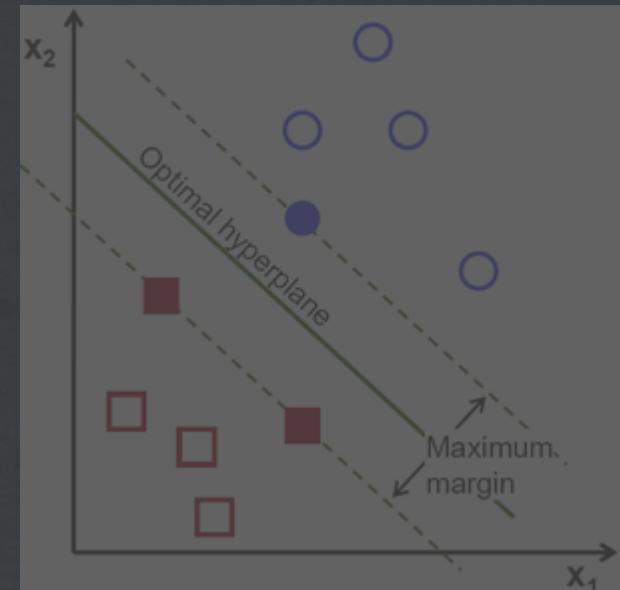
Il classificatore cascade comprende due fasi principali: training e detection.

Nella fase di training si utilizzano due tipologie di immagini:

- immagini positive (immagini che contengono l'oggetto da riconoscere)
- immagini negative (immagini arbitrarie)

Dopo una breve fase di pre-processing le immagini vengono date in pasto ad un algoritmo di apprendimento automatico che utilizza SVM (Support Vector Machine) per dividere le due classi di oggetti e dare il via alla vera fase di training.

Le variabili date in pasto all'algoritmo pregiudicano il risultato.



Training di un classificatore

Nelle prossime lezione potremmo capire come personalizzare un classificatore in base alle nostre esigenze.

```
lina@KORET:~/cygdrive/u/opencv/haarboost$ ./traincascade.exe cat train
PARAMETERS:
cascadeDirName: imgs/cascade
vecFileName: imgs/vector.vec
bgFileName: imgs/negat.dat
numPos: 200
numNeg: 40
numStages: 10
precalcValBufSize[Mb] : 256
precalcIdxBufSize[Mb] : 256
stageType: BOOST
featureType: LBP
sampleWidth: 20
sampleHeight: 35
boostType: GAB
minHitRate: 0.995
falseAlarmRate: 0.001
maxDepth: 10
maxWeakCount: 100
===== STAGE 0 =====
<BEGIN>
POS count : consumed 200 : 200
NEG count : acceptanceRatio 40 : 1
Precalculation time: 0.129
+-----+
| N | HR | FA |
+-----+
| 1| 1| 0|
+-----+
END>
===== TRAINING 1-stage =====
<BEGIN>
POS count : consumed 200 : 200
NEG count : acceptanceRatio 40 : 0.526316
Precalculation time: 0.125
+-----+
| N | HR | FA |
+-----+
| 1| 1| 0|
+-----+
END>
===== TRAINING 2-stage =====
<BEGIN>
POS count : consumed 200 : 200
NEG count : acceptanceRatio 40 : 0.363636
Precalculation time: 0.12
+-----+
| N | HR | FA |
+-----+
| 1| 1| 0|
+-----+
END>
===== TRAINING 3-stage =====
<BEGIN>
POS count : consumed 200 : 200
NEG count : acceptanceRatio 40 : 0.217391
Required leaf false alarm rate achieved - branch training terminated
```

Primi passi. Play with demos.

1. importazione librerie opencv, video Capture, AWT per gestire i poligoni.
2. inizializzazione oggetti camera e openCV.
3. viene eseguita 1 volta.
4. grandezza finestra.
5. parametri oggetto video: sketch corrente, risoluzione della finestra.
6. parametri oggetto openCV (come video).
7. Caricamento classificatore cascade.
8. facciamo partire il video.

```
import gab.opencv.*;  
import processing.video.*;  
import java.awt.*;  
  
Capture video;  
OpenCV opencv;  
  
void setup()  
{  
    size(640, 480);  
  
    video = new Capture(this, 640/2,  
480/2);  
  
    opencv =  
    new OpenCV(this, 640/2, 480/2);  
  
    opencv.loadCascade(OpenCV.  
CASCADE_FRONTALFACE);  
  
    video.start();  
}
```

Primi passi. Play with demos.

9. viene eseguita n volte.
10. zoom dello schermo*2.
11. opencv prende in input ogni frame della camera.
12. opzione di riempimento poligoni, colore delle linee e spessore.
13. ogni viso un rettangolo.
14. stampo numero visi.
15. per ogni viso crea un rettangolo e stampa le sue coordinate a schermo, in alto a sinistra.
16. Manteniamo attiva la camera (IMPORTANTE).

```
void draw()
{
    scale(2);

    opencv.loadImage(video);
    image(video, 0, 0 );

    noFill();
    stroke(0, 255, 0);
    strokeWeight(3);

    Rectangle[] faces = opencv.detect();

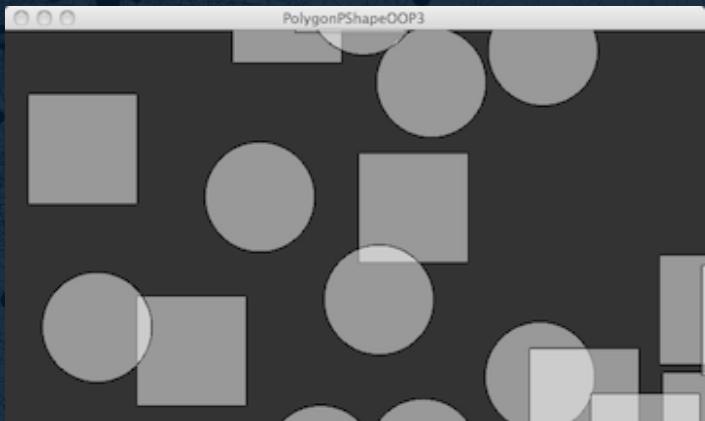
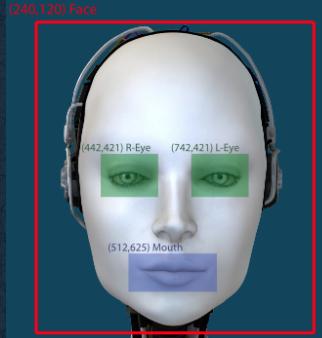
    println(faces.length);

    for (int i = 0; i < faces.length; i++)
    {
        text(faces[i].x + "," + faces[i].y,
            faces[i].x ,faces[i].y);
        rect(faces[i].x, faces[i].y,
            faces[i].width, faces[i].height);
    }
}

void captureEvent(Capture c) { c.read(); }
```

Creiamo il primo sketch.

- Ricordate le basi? rect, ellipse, quad...
- Le funzioni principali? setup, draw.
- Come Importare una libreria?
- Come utilizzare una libreria tramite i suoi esempi?
- Unire una o più librerie per creare il nostro progetto.



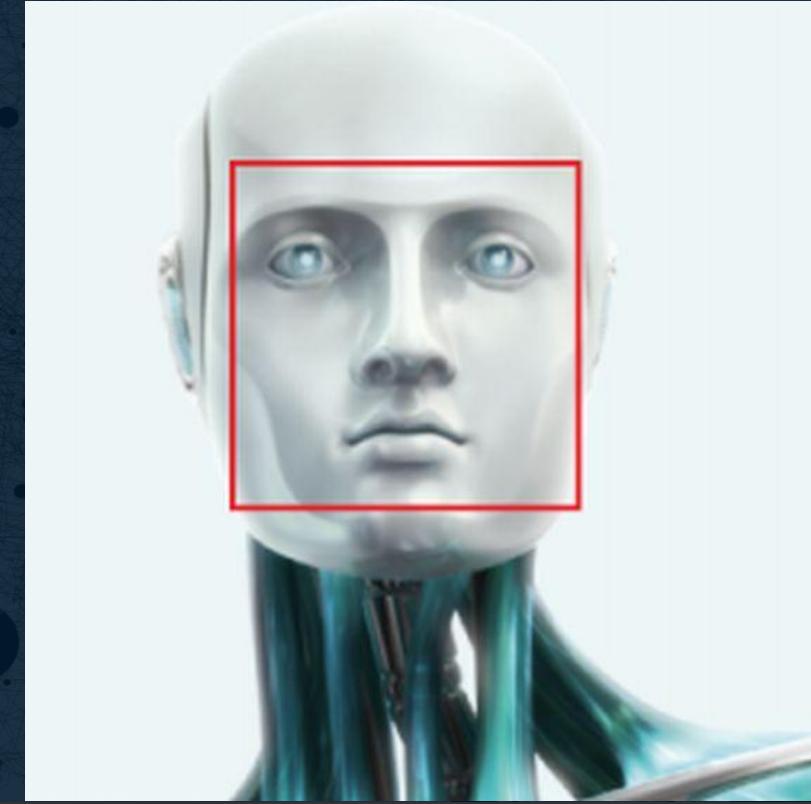
Creiamo il primo sketch. Face Detection.

- Machine Learning su migliaia di foto campione.
- Modelli matematici: Support vector machine, reti neurali, classificatori bayesiani, algoritmi di boost, k - nearest neighbour

Dopo una fase di addestramento, il classificatore può essere applicato ad una regione di interesse in un'immagine in ingresso ed emette un "1" se nella regione viene riconosciuto l'oggetto (cioè, faccia / macchina), e "0" in caso contrario.

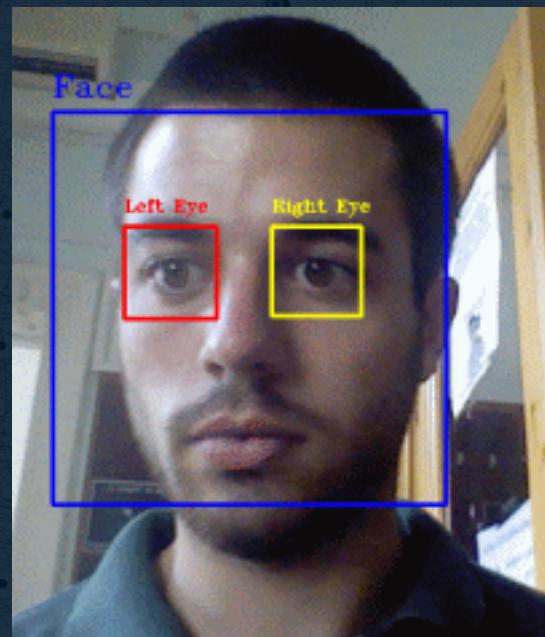
Per cercare l'oggetto la finestra di ricerca si può muovere attraverso l'immagine e controllare ogni posizione utilizzando il classificatore.

Il classificatore è progettato in modo da poter essere facilmente "ridimensionato" in modo da essere in grado di trovare gli oggetti di interesse in diverse dimensioni. Quindi, per trovare un oggetto di una dimensione sconosciuta nell'immagine la procedura di scansione viene eseguita più volte a diverse scale.



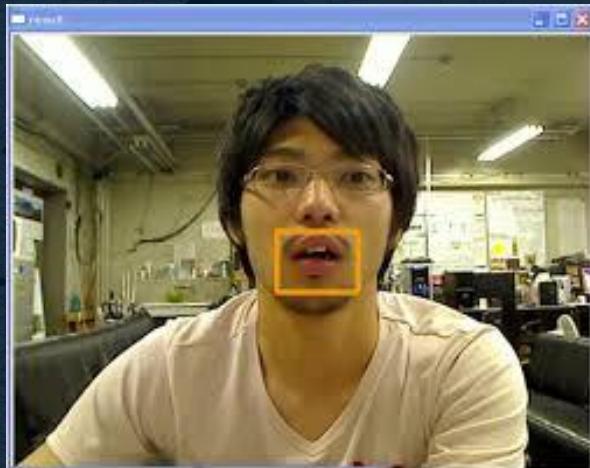
Creiamo il primo sketch. Eyes Detection.

- Per gli occhi ci sono alcune differenze riguardanti soprattutto la posizione rispetto al viso.
- Modello matematico: Supprt Vector Machine, etc.
- Percentuale di falsi positivi più alta
- Abbiamo bisogno di mettere il fuoco sulla zona superiore del viso.



Creiamo il primo sketch. Mouth Detection.

- Per il riconoscimento della bocca, come per gli occhi dobbiamo cercare la ROI (Region Of Interest).
- Nel caso della bocca la ROI si troverà sicuramente sotto la metà del viso.
- Anche in questo caso useremo un file train xml.



Bonus Stage. Background subtraction.

Le basi per un algoritmo di motion detection possono ricercarsi in questa tecnica. Il BackgroundSubtractorMOG è il Gaussian Mixture-based Background/Foreground Segmentation Algorithm che si occupa di sottrarre da un frame tutti gli elementi del frame precedente così da avere come risultato gli oggetti in movimento ad esempio.

Difetti:

- cambi di illuminazione
- vento
- riflessi
- difficile da applicare all'esterno

$$P[F(t)] = P[I(t)] - P[B] \quad |P[F(t)] - P[F(t + 1)]| > \text{Threshold}$$



Bonus Stage. Background subtraction.

L'algoritmo è l'implementazione di P. KadewTraKuPong and R. Bowden.

Le funzioni che vengono utilizzate sono:

1. `opencv.startBackgroundSubtraction(history, nMixtures, backgroundRatio);`
 - a. history 5 frame precedenti.
 - b. nMixtures 3 o 5 rappresenta la proporzione di tempo in cui il colore di sfondo viene conteggiato ai fini della subtraction.
 - c. backgroundRatio 0.5 soglia che definisce se un elemento debba essere incluso nel modello 0.9 di default.
2. `opencv.updateBackground();`
3. `for (Contour contour : opencv.findContours()) contour.draw();`

Con la prima funzione predisponiamo opencv alla sottrazione dello sfondo per la prima immagine.

Con la seconda (da scrivere nella draw) aggiorniamo lo sfondo al nuovo fotogramma.

Nel terzo punto scorriamo i contorni di ciò che si muove e li stampiamo a schermo.

Optical Flow - OpenCV

L'Optical Flow è un concetto che considera il moto di un oggetto all'interno di una rappresentazione visuale digitale.

Tipicamente il moto è rappresentato come un vettore che si origina da (o termina su) un pixel ad esempio in una sequenza di frame. Lo scopo dell'Optical Flow è quello di assegnare ad ogni pixel appartenente al frame corrente un motion vector che punta verso la posizione dello stesso pixel in un frame di riferimento successivo.

vedi: [Wiki](#).



Optical Flow.

Le assunzioni alla base di questa teoria sono:

- l'intensità di un pixel non cambia tra frame consecutivi.
- I pixel vicini hanno movimento (motion) simile.

il metodo è anche chiamato Lucas Kanade e assume che il flusso in un vicinato locale è essenzialmente costante.



Fai da te.

- Basi (quadrati, rettangoli, linea, punto, archi, ellisse)
- Controllo periferiche (tastiera, mouse, arduino ...)
- Controllo file (apri, scrivi, salva)
- Audio (registra, riproduci, crea suoni)
- OpenCV library (eye, mouth, face detection ...)
- e tanto altro: <https://processing.org/reference/>



Fai da te. Ora tocca a te.

- Giochi
- Utilities
- Software
- Social App
- Healt Care
- Sorveglianza

