



# PROGETTO MUSICA HUMANA

Progetto sviluppato in collaborazione fra Politecnico di Bari e Conservatorio di Matera

---

*Relazione: Giuseppe De Santis*



# Indice generale

Introduzione.....	1
1 Strumenti Utilizzati.....	2
1.1 OpenCV.....	2
1.2 Flandmark.....	2
1.3 Qt.....	2
2 Parametri del volto.....	3
2.1 Misure Verticali.....	3
2.2 Misure Orizzontali.....	4
2.3 Angoli.....	4
2.3.1 Funzione per il calcolo degli angoli.....	5
3 Calcolo delle note caratteristiche del volto.....	6
3.1 Forma del volto e modo associato.....	6
3.2 Calcolo delle note.....	9
4 File MIDI.....	11
4.1 Struttura del File MIDI.....	11
4.1.1 Blocco di Intestazione.....	11
4.1.2 Blocco di Traccia.....	12
4.1.3 Tempo Delta.....	13
4.1.4 Meta Dati.....	13
4.1.5 Eventi MIDI.....	13
4.1.6 Esempio di traccia MIDI.....	14
5 Implementazione.....	17
5.1 Interfaccia grafica e funzionamento dell'applicazione.....	17
5.2 Metodo utilizzato per il calcolo della nota.....	20
5.3 Impostazioni dell'applicazione.....	24
5.4 Deploy in ambiente Mac OS.....	24

## **Introduzione**

---

L'obiettivo di questo progetto è la realizzazione di un'applicazione che permetta di caricare un'immagine o acquisirla tramite webcam e sia in grado di calcolare distanze e angoli individuati dai punti caratteristici presenti sul volto della persona inquadrata (occhi, naso, bocca...).

Il sistema deve produrre in output un file midi caratterizzato da una successione di note che verranno calcolate in base ai rapporti fra le distanze calcolate.

E' stata realizzata una versione desktop, compilata per essere eseguita su Mac Os, e una versione per cellulari con sistema operativo Android.

## **1 Strumenti Utilizzati**

---

### **| 1.1 OpenCV**

E' stata utilizzata la libreria OpenCV sia per la versione desktop sia per quella mobile.

La libreria è gratuita sia per fini accademici che commerciali, è scritta in C/C++, possiede interfacce per i linguaggi: C, C++, Java, Python e supporta sistemi operativi Windows, Linux, Mac OS, Android e iOS.

### **| 1.2 Flandmark**

Flandmark è una libreria open source scritta in linguaggio C, e rilasciata sotto licenza GNU/GPL versione 3, che implementa un sistema per l'individuazione di punti caratteristici facciali. Questa libreria è stata utilizzata esclusivamente per la versione desktop, insieme al classificatore di Haar per il volto presente in OpenCV.

Tramite l'utilizzo di questa libreria si riescono ad individuare accuratamente gli estremi degli occhi, il naso e gli estremi della bocca.

Il 28 Aprile 2015 è stata presentata la più recente libreria Clandmark, che sostituisce la precedente, per sviluppi futuri può essere utile considerare l'aggiornamento del codice utilizzando tale libreria.

### **| 1.3 Qt**

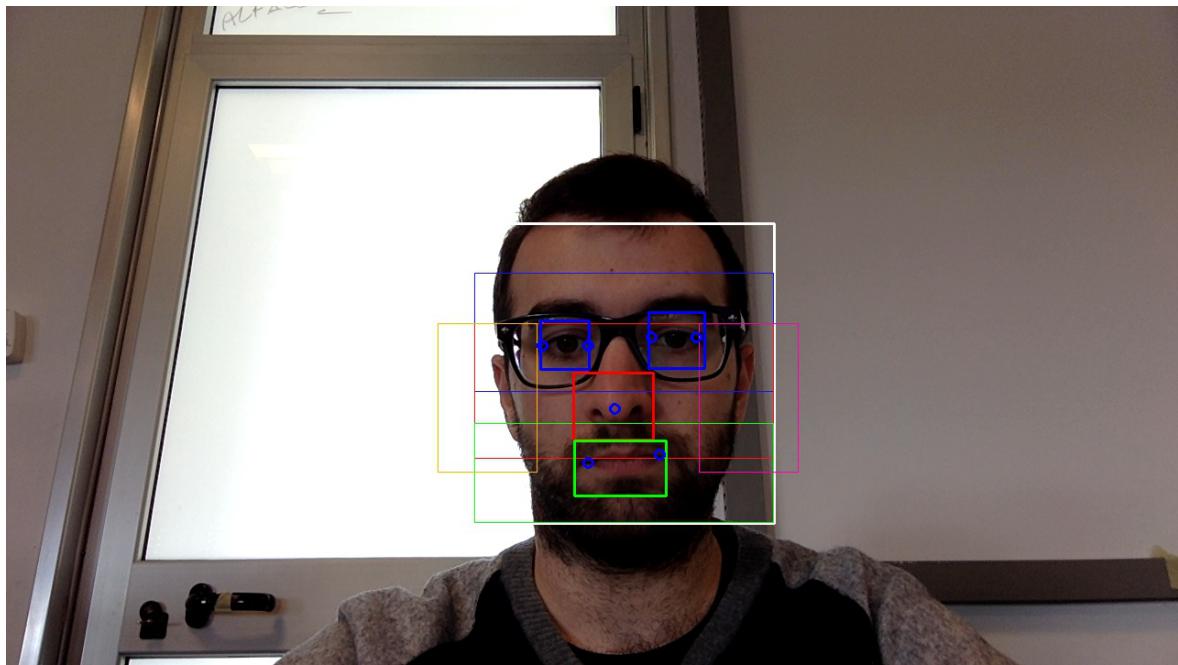
Qt è una libreria multipiattaforma per lo sviluppo di programmi con interfaccia grafica tramite l'uso di widget, è stata scelta in quanto sviluppata in C++, linguaggio utilizzato per la realizzazione del progetto, e che per questo si integra perfettamente con l'applicazione sviluppata.

## **2 Parametri del volto**

---

### **| 2.1 Misure Verticali**

Il primo parametro calcolato è l'altezza del volto, misurata dall'attaccatura dei capelli, fino al mento (nell'immagine: altezza del rettangolo bianco).



Tramite questo parametro viene calcolata una distanza di riferimento per le misure successive (che daranno indicazioni sulle note caratteristiche) dividendo tale misura per 12 (numero di semitonini in un'ottava).

Nell'immagine sono visualizzati dei rettangoli colorati, questi indicano le zone di interesse all'interno delle quali sono ricercati i punti caratteristici.

I parametri calcolati effettuando misure verticali sono:

- Ampiezza della fronte (dalla linea bianca superiore all'altezza media dei 4 punti rilevati sugli occhi);

- Ampiezza del naso (dall'altezza media degli occhi al punto inferiore del naso);
- Distanza bocca-naso.

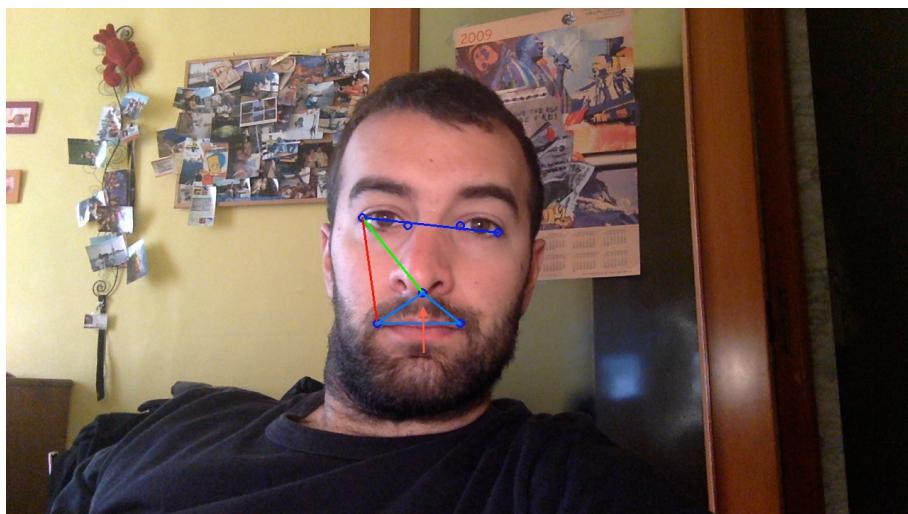
## | 2.2 Misure Orizzontali

I parametri calcolati effettuando misure orizzontali sono:

- Distanza fra gli occhi (dai punti più esterni rilevati);
- larghezza della bocca.

## | 2.3 Angoli

Sono stati calcolati gli angoli del triangolo che si forma collegando gli estremi della bocca e il naso (triangolo azzurro nell'immagine seguente); in particolare siamo interessati all'angolo che si forma fra i segmenti ottenuti unendo l'estremo destro della bocca con il naso e il naso con l'estremo sinistro della bocca, nell'immagine è indicato dalla freccia rossa; vengono prima calcolati i due angoli sulla linea della bocca e successivamente viene calcolato l'angolo desiderato effettuando la sottrazione fra 180 e gli altri due angoli del triangolo, questo perché gli angoli alla base della bocca sono sempre minori di 90°, cosa non vera per l'angolo che si forma sul naso.



### 2.3.1 Funzione per il calcolo degli angoli

```

float MainWindow::calcolaAngolo(Point a, Point b, Point c, Point d)
{
    float angolo;
    float m1, m2;
    //se le coordinate x sono uguali -> m assume valore infinito, assegno un
    valore alto
    if (b.x==a.x){
        m1=1000;
    } else {
        m1=((float)b.y-(float)a.y)/((float)b.x-(float)a.x);
    }

    if (d.x==c.x){
        m2=1000;
    } else {
        m2=((float)d.y-(float)c.y)/((float)d.x-(float)c.x);
    }

    float argomentoTangente =(m1-m2)/(1+ (m1*m2) );
    angolo = atan( argomentoTangente )

    if (angolo<0){
        angolo = angolo * (-1);
    }

    return angolo;
}

```

La funzione ha come parametri i punti a e b che individuano il primo segmento e i punti c e d che individuano il secondo segmento; vengono calcolati i coefficienti delle due rette ( $m_1$  e  $m_2$ ) e viene calcolato l'angolo che si forma fra le due rette con la formula:

$$\text{angolo}=\arctan\left(\left|\frac{m_1-m_2}{1+m_1m_2}\right|\right)$$

### 3 Calcolo delle note caratteristiche del volto

#### 3.1 Forma del volto e modo associato

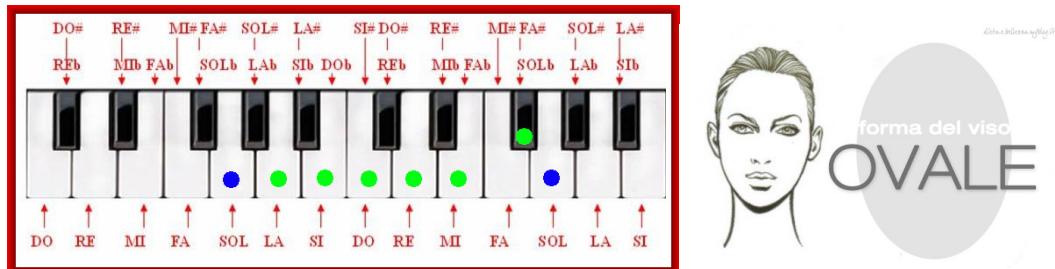
La prima variabile presa in considerazione è la forma del volto, l'utente deve selezionare la forma che ritiene più appropriata fra 7 scelte, ad ogni scelta è associato un modo, costruito sulla nota SOL; il modo determina gli intervalli che saranno utilizzati per il calcolo di ciascuna nota.

Di seguito sono riportati i modi associati ad ogni volto, le note appartenenti al modo e gli intervalli che separano ogni nota dalla nota successiva.

- Volto Ovale → Modo Ionico:

note: Sol - La - Si - Do - Re - Mi - Fa # - Sol;

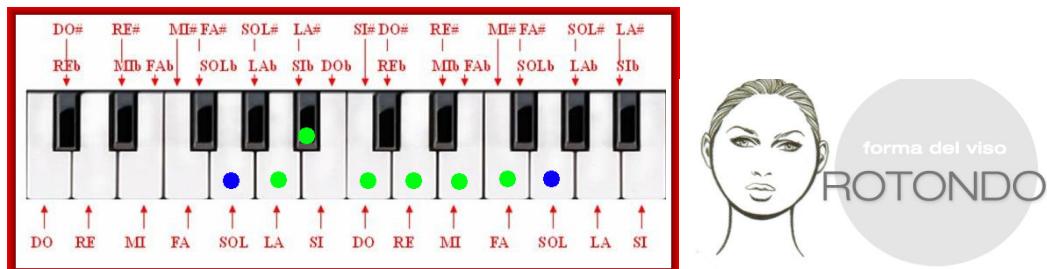
intervalli: T - T - S - T - T - T - S;



- Volto Rotondo → Modo Dorico:

note: Sol - La - Si b - Do - Re - Mi - Fa - Sol;

intervalli: T - S - T - T - T - S - T;



## Calcolo delle note caratteristiche del volto

- Volto Quadrato → Modo Frigio:

note: Sol – La *b* – Si *b* – Do – Re – Mi *b* – Fa - Sol;

intervalli: S – T – T – T – S – T – T;

**form del viso QUADRATO**

- Volto a Diamante → Modo Lidio:

note: Sol – La – Si – Do # – Re – Mi – Fa # - Sol;

intervalli: T – T – T – S – T – T – S;

**form del viso A DIAMANTE**

- Volto a Cuore → Modo Misolidio:

note: Sol – La – Si – Do – Re – Mi – Fa - Sol;

intervalli: T – T – S – T – T – S – T;

**form del viso A CUORE**

## Calcolo delle note caratteristiche del volto

- Volto Rettangolare → Modo Eolio:

note: Sol - La - Si b - Do - Re - Mi b - Fa - Sol;

intervalli: T - S - T - T - S - T - T;

Detailed description: This diagram shows a segment of a piano keyboard with specific notes highlighted in green. Above the keyboard, red arrows point upwards from each of the highlighted notes. Red labels above the keys identify them: DO#, RE#, MI# FA#, SOL#, LA#, SI# DO#, RE#, MI# FA#, SOL#, LA#. To the right of the keyboard, there is a line drawing of a woman's face with short hair, and the text 'forma del viso RETTANGOLARE'.

- Volto Triangolare → Modo Locrio:

note: Sol - La b - Si b - Do - Re b - Mi b - Fa - Sol;

intervalli: S - T - T - S - T - T - T;

Detailed description: This diagram shows a segment of a piano keyboard with specific notes highlighted in green. Above the keyboard, red arrows point upwards from each of the highlighted notes. Red labels above the keys identify them: DO#, RE#, MI# FA#, SOL#, LA#, SI# DO#, RE#, MI# FA#, SOL#, LA#. To the right of the keyboard, there is a line drawing of a woman's face with a triangular shape, and the text 'forma del viso TRIANGOLARE'.

### | 3.2 Calcolo delle note

La nota di partenza per il calcolo della prima nota è SOL-0, il primo parametro che viene considerato, per il calcolo della nota più bassa (o fondamentale), è l'ampiezza dell'angolo che si forma unendo gli estremi della bocca con il naso, la nota sarà selezionata in base alla tabella seguente:

ANGOLO	NOTA
$0 < x \leq 60$	SOL
$60 < x \leq 67$	LA
$67 < x \leq 73,5$	SI
$73,5 < x \leq 79$	DO
$79 < x \leq 83,4$	RE
$83,4 < x \leq 86,7$	MI
$86,7 < x \leq 88,9$	FA
$88,9 < x \leq 91,1$	SOL
$91,1 < x \leq 93,3$	LA
$93,3 < x \leq 96,6$	SI
$96,6 < x \leq 101$	DO
$101 < x \leq 106,5$	RE
$106,5 < x \leq 113,1$	MI
$113,1 < x \leq 120$	FA
$120 < x < 180$	SOL

Per il calcolo delle note successive vengono considerate, nell'ordine, le seguenti misure: ampiezza della fronte, ampiezza del naso, distanza verticale fra bocca e naso, distanza fra i punti esterni degli occhi, ampiezza della bocca.

Ogni misura viene divisa per un riferimento (calcolato come altezza del volto/12), questa operazione serve a rendere il sistema robusto alla variazione dello zoom dell'immagine acquisita, altrimenti tutte le misure risulterebbero diverse sullo stesso soggetto che scatta più foto a distanze differenti rispetto alla fotocamera.

### *Calcolo delle note caratteristiche del volto*

La misura effettuata, una volta divisa per il riferimento e arrotondata all'intero più vicino, rappresenta l'intervallo fra la fondamentale e la nota che andrà scritta nel file midi, ogni nota sarà calcolata a partire dall'ottava successiva rispetto a quella utilizzata per il calcolo della nota precedente, in questo modo tutte le note sono distribuite equamente sul registro del pianoforte.

## 4 File MIDI

---

Il formato MIDI (Musical Instrument Digital Interface) è un archivio contenente dati, informazioni espresse in uno o più byte. Tali informazioni sono comprensibili dai lettori MIDI (sequencer), che le riconoscono e le interpretano come istruzioni da eseguire.

### | 4.1 Struttura del File MIDI

Il file MIDI è composto da complesse strutture, tecnicamente chiamate: Track Chunk (Blocco-traccia), formate da un certo numero di byte (informazioni-istruzioni). Ogni Blocco è un insieme di informazioni che conferisce a questa struttura un senso compiuto ai fini del riconoscimento e dell'esecuzione del file MIDI da parte del sequencer.

Aprendo il file midi con un editor esadecimale possiamo osservare la sua struttura.

#### | 4.1.1 Blocco di Intestazione

La prima struttura, posta solo all'inizio del file MIDI e composta sempre da 14 byte, è chiamata: Midi Track header chunk (MThd), ossia Blocco d'intestazione. Alcuni byte di questo Blocco-traccia d'intestazione del file MIDI sono invariabili, altri invece variabili a seconda ovviamente delle informazioni che trasmettono. In un file esiste un solo blocco Midi Track header (MThd). Il Blocco d'Intestazione (Header Chunk) specifica alcune informazioni di base e generiche relative ai dati presenti nel file. Aperto il file MIDI con un lettore del codice esadecimale di file, il Midi Track header chunk sarà così visualizzato:

4D 54 68 64 00 00 00 06 00 0n 00 nn 0n nn

laddove n rappresenta i byte di valore variabile.

4D 54 68 64 = (in codice ASCII: MThd) definisce il chunk come Midi Track header. Esso è il blocco d'intestazione, formato sempre da 4 byte invariabili, che definisce questa traccia come blocco header.

00 00 00 06 = definisce la lunghezza in byte della restante parte del blocco header. Indica che immediatamente dopo vi sono 6 byte restanti.

00 0n = questi due byte indicano il tipo di file MIDI – smf: Standard Midi File.

Per il formato 0 n = 0; per il formato 1 n = 1; per il formato 2 n = 2:

Formato 0: le tracce di un brano sono fuse in una singola che contiene, però, tutte le informazioni degli eventi di tutte le tracce originarie del brano.

Formato 1: le tracce sono memorizzate singolarmente condividendo gli stessi valori di tempo e metrica. La velocità del brano viene inserita nella prima traccia, la quale fa da riferimento a tutte le altre. Il formato 1 permette una gestione multitraccia di un brano nei sequencer e riproduttori di file MIDI ed è il formato più usato.

Formato 2: le tracce sono gestite indipendentemente l'una dall'altra, anche per il tempo e per la metrica.

00 nn = questi due byte indicano quante tracce (del tipo MTrk) sono presenti nel file MIDI.

0n nn = questi ultimi due byte indicano il “numero di impulsi (o risoluzione) per nota da  $\frac{1}{4}$ ”: PPQN (Pulse Per Quarter Note). Ogni impulso è chiamato “Tick” (istante).

#### | **4.1.2 Blocco di Traccia**

Dopo l'intera descrizione del Midi Track header chunk (Blocco primario d'intestazione) seguono le Tracce (almeno una) contenenti gli eventi del MIDI.

Ciascuna traccia è composta da un'intestazione, che la definisce come tale, formata sempre da 4 byte invariabili:

4D 54 72 6B = (in codice ASCII: MTrk) definisce il chunk come Midi Track, ossia lo identifica come blocco di traccia.

Qualora nel file sia presente più di una traccia MTrk, la prima traccia MTrk, chiamata anche “Traccia del Tempo”, contiene solitamente i Dati Meta (separati fra loro e da altri eventuali eventi MIDI dal Tempo Delta – T $\Delta$ ) relativi all'indicazione della suddivisione della misura, del tempo metronomico e della tonalità della scala.

Le tracce successive (per esempio una per ogni strumento musicale diverso) contengono tutti gli altri eventi midi, e possono contenere anche eventi Dati Meta in caso di cambio dei valori dei Dati Meta iniziali contenuti nella prima traccia MTrk, ossia nella Traccia del Tempo.

Dopo i 4 byte identificativi della traccia seguono sempre altri 4 byte:

nn nn nn nn = indicano quanti byte successivi vi sono sino alla fine della traccia.

Dopo i descritti 4 byte è sempre presente il Tempo Delta (Delta Time), espresso con uno o più byte. Seguono, quindi, gli eventi contenuti nella traccia che rappresentano quegli elementi che danno sostanza al file MIDI in quanto codifica informatica di un brano o di una partitura musicale. Tutti gli Eventi sono sempre separati fra loro da un valore di Tempo Delta variabile a seconda delle necessità e dello svolgersi del brano.

#### | **4.1.3    *Tempo Delta***

All'interno delle tracce MTrk fra ogni Evento viene sempre inserito il dato temporale Tempo Delta (Delta-time), che esprime in byte il tempo che scorre tra due singoli eventi (dopo quanto tempo un evento Midi avverrà rispetto all'evento che lo precede). Rappresenta, quindi, la durata in PPQN tra un evento e quello che lo segue.. Se, ad esempio,  $T\Delta = 00$ , allora gli eventi MIDI separati da tale  $T\Delta$  avvengono simultaneamente.

#### | **4.1.4    *Meta Dati***

I dati meta sono informazioni non fondamentali dei file MIDI e sono caratterizzati dal valore del primo byte che è sempre uguale FF, possono contenere informazioni sul tempo metronomico del brano, sulla tonalità, sul nome della traccia ecc...

#### | **4.1.5    *Eventi MIDI***

Gli Eventi MIDI sono quegli elementi, contenuti nelle tracce MTrk, che possiamo considerare le informazioni più importanti del file MIDI. Poiché il MIDI fu concepito inizialmente soprattutto per le tastiere musicali, i messaggi MIDI (Byte) si riferiscono ad un'azione musicale applicata alla tastiera. Se vi è più di una traccia MTrk, gli eventi MIDI saranno contenuti nelle tracce successive alla prima (Traccia del Tempo), restando presenti in detta traccia i Dati Meta.

#### | **4.1.6 Esempio di traccia MIDI**

Di seguito è riportata una traccia MIDI ottenuta come output dell'applicazione, con alcuni commenti sui messaggi più importanti. Il file è composto da un header e 5 tracce: una traccia per il tempo, una traccia contenente l'accordo, due contenenti un arpeggio e una contenente la melodia, con le note caratteristiche riportate all'ottava centrale.

##### **Header**

4D 54 68 64 Midi Track Header Chunk

00 00 00 06 Lunghezza in byte della restante parte dell'header

00 01 Multitraccia sincrona (Tipo 1)

00 05 Numero di tracce

03 C0 PPQN

##### **Traccia del tempo**

4D 54 72 6B Midi Track Chunk

00 00 00 24 Lunghezza in byte della restante parte della traccia

00 FF 51 03 07 A1 20 Tempo metronomico

00 FF 59 02 00 00 Chiave di tonalità

00 FF 03 07 6F 75 74 70 75 74 00 Nome della traccia

00 FF 58 04 04 02 18 08 Metro della misura

00 FF 2F 00 Chiusura della traccia

##### **Traccia dell'accordo**

4D 54 72 6B 00 00 00 A3 00

B0 00 Bank Select

00 00 20 00 00 FF 03 09 74 72 61 63 63 69 61 31 00 00

B0 07 5A Control Change n.7 (Volume) sul canale 0

00 00 00 00 20 00 00 07 5A

00 5B 7F Control Change n.91 (Riverbero)

00 00 00 00 20 00 00 07 5A 00 00 00 00 20 00 00 07 5A 00 00 00 00 20 00 00 07 5A 00 00 00 00 20 00  
00 07 5A

00 Delay Time

***Accordo Iniziale***

90 1A 5A Note On (90 = note on; 1A = nota da suonare; 5A = velocity)

00 2D 5A 00 36 5A 00 3E 5A 00 53 5A 00 5A 5A Altre note da suonare (00 = delay time, nn = nota da suonare; 5A = velocity)

81 B4 00 Delay time

80 1A 00 Note Off (80 = note off; 1A = nota da rilasciare; 00 = velocity)

00 2D 00 00 36 00 00 3E 00 00 53 00 00 5A 00 84 D8 00

***Accordo sincrono alla melodia***

90 1A 5A 00 2D 5A 00 36 5A 00 3E 5A 00 53 5A 00 5A 5A 81 A5 00 80 1A 00 00 2D 00 00 36 00 00 3E  
00 00 53 00 00 5A 00 8E 7F

B0 7F 00 All Note Off

00 Delay Time

FF 2F 00 Fine Traccia

***Traccia del primo arpeggio – Ascendente con minime***

4D 54 72 6B 00 00 00 5A 00 B1 00 00 00 20 00 00 FF 03 09 74 72 61 63 63 69 61 32 00 00

B1 07 5A Control change n.7 sul canale 1

81 F0 00 Delay Time iniziale

91 1A 5A Note on sul canale 1

8F 00 Delay Time corrispondente a una minima

81 1A 00 Note off

00 Delay Time dall'evento di Note Off. Vale 0 perché la nota successiva dell'arpeggio viene

*File MIDI*

suonata appena viene rilasciata quella precedente.

91 2D 5A 8F 00 81 2D 00 00 91 36 5A 8F 00 81 36 00 00 91 3E 5A 8F 00 81 3E 00 00 91 53 5A 8F 00  
81 53 00 00 91 5A 5A 8F 00 81 5A 00 84 F5 7F B0 7F 00 00 FF 2F 00

**Traccia del secondo arpeggio – Ascendente e discendente con semiminime**

4D 54 72 6B 00 00 00 90 00 B2 00 00 00 20 00 00 FF 03 09 74 72 61 63 63 69 61 33 00 00 B2 07 5A 83  
86 00 92 1A 5A 87 40 82 1A 00 00 92 2D 5A 87 40 82 2D 00 00 92 36 5A 87 40 82 36 00 00 92 3E 5A 87  
40 82 3E 00 00 92 53 5A 87 40 82 53 00 00 92 5A 5A 87 40 82 5A 00 00 92 5A 5A 87 40 82 5A 00 00 92  
53 5A 87 40 82 53 00 00 92 3E 5A 87 40 82 3E 00 00 92 36 5A 87 40 82 36 00 00 92 2D 5A 87 40 82 2D  
00 00 92 1A 5A 87 40 82 1A 00 83 DF 7F B0 7F 00 00 FF 2F 00

**Traccia della melodia**

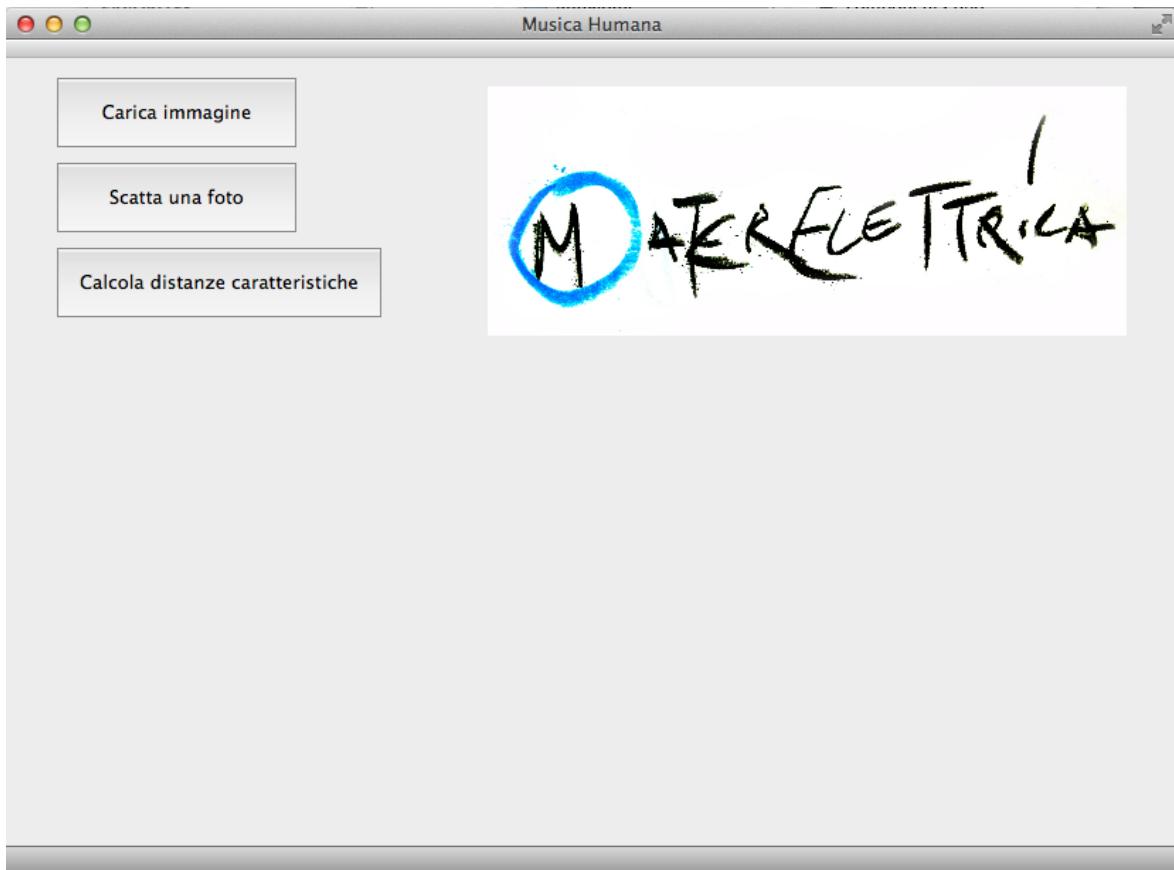
4D 54 72 6B 00 00 00 EA 00 B3 00 00 00 20 00 00 FF 03 09 74 72 61 63 63 69 61 34 00 00 B3 07 5A  
84 9C 00 93 3E 5A 8F 00 83 3E 00 00 93 45 5A 8F 00 83 45 00 00 93 42 5A 8F 00 83 42 00 00 93 3E 5A 8F  
00 83 3E 00 00 93 47 5A 8F 00 83 47 00 00 93 42 5A 8F 00 83 42 00 00 93 47 5A 8F 00 83 47 00 00 93 3E  
5A 8F 00 83 3E 00 00 93 42 5A 8F 00 83 42 00 00 93 45 5A 8F 00 83 45 00 00 93 3E 5A 8F 00 83 3E 00 CB  
00 93 3E 5A 8F 00 83 3E 00 00 93 45 5A 8F 00 83 45 00 00 93 42 5A 8F 00 83 42 00 00 93 3E 5A 8F 00 83  
3E 00 00 93 47 5A 8F 00 83 47 00 00 93 42 5A 8F 00 83 42 00 00 93 47 5A 8F 00 83 47 00 00 93 3E 5A 8F  
00 83 3E 00 00 93 42 5A 8F 00 83 42 00 00 93 45 5A 8F 00 83 45 00 00 93 3E 5A 8F 00 83 3E 00 8E 7F B0  
7F 00 00 FF 2F 00

## 5 Implementazione

---

### 5.1 Interfaccia grafica e funzionamento dell'applicazione

All'apertura dell'applicazione viene mostrata un'immagine di default e 3 pulsanti: Carica Immagine, Scatta una foto e Calcola distanze caratteristiche.



Cliccando su “Carica Immagine” viene chiamata la funzione per il caricamento di un'immagine, l'immagine viene salvata all'interno dell'applicazione tramite la funzione opencv imwrite(String path, Mat image) e visualizzata nella schermata dell'applicazione tramite la funzione visualizzaImmagine (Mat, CVImageWidget) definita in MainWindow.cpp, con questo

metodo se l'immagine è più grande del widget di visualizzazione, viene adattata con il metodo cv::resize(Mat src, Mat dst, Size).

Cliccando su “Scatta una foto” viene aperta la camera di default del sistema istanziando un oggetto di tipo cv::VideoCapture e viene mostrata a video l'immagine catturata dalla camera, alla pressione di un tasto viene chiusa la webcam, viene memorizzata l'immagine e viene visualizzata sempre con il metodo visualizzaImmagine (Mat, CVImageWidget).

Dopo aver caricato un'immagine è possibile calcolare le distanze fra i punti caratteristici del volto e salvare un file MIDI come output. All'interno del metodo per il calcolo delle distanze viene istanziato un nuovo oggetto di tipo cv::Mat atto a contenere l'immagine acquisita convertita in scala di grigi, in quanto sia il classificatore di openCV sia quello della libreria Flandmark utilizzano immagini in scala di grigio.

Sull'immagine in scala di grigi viene lanciata la funzione:

```
face_cascade.detectMultiScale( frame_gray, faces, 1.1, 2,  
CV_HAAR_FIND_BIGGEST_OBJECT, dimensioneMinima );
```

face\_cascade è il classificatore caricato nell'inizializzazione dell'applicazione tramite l'xml fornito con openCV (haarcascade\_frontalface\_alt.xml).

Fra i parametri della funzione ci sono:

- frame\_gray: Oggetto di tipo Mat contenente l'immagine in scala di grigi;
- faces: vettore di elementi di tipo cv::Rect, che conterrà la lista dei rettangoli all'interno dei quali sono contenuti i volti individuati.

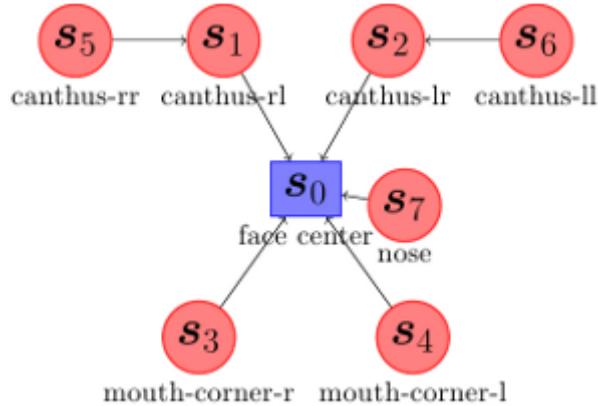
Nel caso in cui nell'immagine sia stato individuato almeno un volto, viene selezionato quello più grande e si utilizza la libreria Flandmark, la quale utilizza le immagini all'interno di oggetti di tipo IplImage, quindi è necessario rileggere l'immagine acquisita con la funzione cv::cvLoadImage(String path) e ritrasformarla in scala di grigi con cv::cvCvtColor().

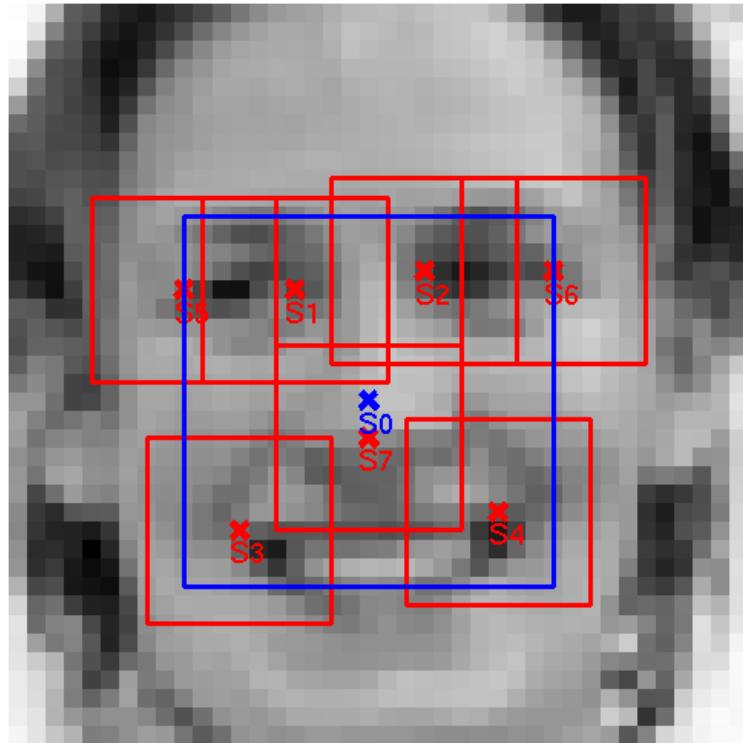
Inizialmente sono stati utilizzati sia il classificatore di haar fornito con OpenCV, sia quello della libreria Flandmark, successivamente è stato utilizzato solo quello di Flandmark in quanto più

accurato, sono comunque disponibili le funzioni di opencv all'interno del codice, ma sono commentate perché non più utilizzate ai fini dell'output.

La funzione per il detect di Flandmark inserisce in un vettore di double (double \*landmarks) le coordinate dei seguenti punti:

- Centro del volto (v[0], v[1])
- punto interno dell'occhio destro (v[2], v[3])
- punto interno dell'occhio sinistro (v[4], v[5])
- punto destro della bocca (v[6], v[7])
- punto sinistro della bocca-naso (v[8], v[9])
- punto esterno dell'occhio destro (v[10], v[11])
- punto esterno dell'occhio sinistro (v[12], v[13])
- naso (v[14], v[15])





Combinando questi punti si effettuano le varie misurazioni e si procede alla scrittura del file midi.

## | 5.2 Metodo utilizzato per il calcolo della nota

Come è già stato scritto, la selezione del volto indica il modo che sarà utilizzato, quindi le alterazioni che dovrà avere ciascuna nota.

Quando viene chiamata la funzione per il calcolo delle distanze fondamentali viene chiesto all'utente di selezionare la forma del volto del soggetto nell'immagine, a seconda di questa scelta viene inizializzato il vettore vettoreIntervalli[7], ogni elemento del vettore rappresenta la distanza (in semitonni) dal SOL nel modo selezionato per ogni nota della scala () .

Una volta individuata la fondamentale (nota più bassa, definita dall'ampiezza dell'angolo fra bocca e naso) si procede a inizializzare vettoreIntervalliDaTonica[7], ogni elemento di questo

vettore indica la distanza in semitonni di ogni intervallo dalla fondamentale, considerando la scala modale costruita sul SOL in precedenza.

Esempio:

Viene selezionato il volto a forma di diamante che implica che le note dovranno appartenere alla scala di SOL lidio, cioè Sol - La – Si – Do # – Re – Mi – Fa # – Sol, viene quindi inizializzato vettoreIntervalli nel modo seguente:

```
vettoreIntervalli[0] = 0; //vale sempre 0
vettoreIntervalli[1] = 2; //distanza in semitonni fra La e Sol
vettoreIntervalli[2] = 4; //distanza in semitonni fra Si e Sol
vettoreIntervalli[3] = 6; //distanza in semitonni fra Do# e Sol
vettoreIntervalli[4] = 7; //distanza in semitonni fra Re e Sol
vettoreIntervalli[5] = 9; //distanza in semitonni fra Mi e Sol
vettoreIntervalli[6] = 11; //distanza in semitonni fra Fa# e Sol
```

Successivamente viene individuato l'angolo fra naso e bocca, supponiamo che l'angolo sia di  $102^\circ$  e rientra quindi nell'intervallo  $101 < x \leq 106,5$  che implica che la prima nota è un RE e la variabile int **tonica** assume il valore di 4 (Considerando sol=0, la=1, si=2, do=3, re=4, mi=5, fa=6).

Si procede ad inizializzare vettoreIntervalliDaTonica[7] nel modo seguente:

```
vettoreIntervalliDaTonica[0]=0;
for (int i=1; i<7; i++){
    vettoreIntervalliDaTonica[i]=vettoreIntervalliDaTonica[i-
1]+vettoreIntervalli[(tonica+i)%7]-vettoreIntervalli[(tonica+i-1)%7];
    if (vettoreIntervalliDaTonica[i]<0){
        vettoreIntervalliDaTonica[i]=vettoreIntervalliDaTonica[i-1] + 12-
vettoreIntervalli[(tonica+i-1)%7];
    }
}
```

Quindi:

- i=1

$vettoreIntervalliDaTonica[1] = vettoreIntervalliDaTonica[0] + vettoreIntervalli[5] - vettoreIntervalli[4] = 0 + 9 - 7 = 2$

- i=2

$vettoreIntervalliDaTonica[2] = vettoreIntervalliDaTonica[1] + vettoreIntervalli[6] - vettoreIntervalli[5] = 2 + 11 - 9 = 4$

- i=3

vettoreIntervalliDaTonica[3] = vettoreIntervalliDaTonica[2] + vettoreIntervalli [0] -  
vettoreIntervalli [6] =  $4 + 0 - 11 = -7$

vettoreIntervalliDaTonica[3] = vettoreIntervalliDaTonica[2] + 12 - vettoreIntervalli[6] =  
 $4 + 12 - 11 = 5$

- i=4

vettoreIntervalliDaTonica[4] = vettoreIntervalliDaTonica[3] + vettoreIntervalli [1] -  
vettoreIntervalli [0] =  $5 + 2 - 0 = 7$

- i=5

vettoreIntervalliDaTonica[5] = vettoreIntervalliDaTonica[4] + vettoreIntervalli [2] -  
vettoreIntervalli [1] =  $7 + 4 - 2 = 9$

- i=6

vettoreIntervalliDaTonica[6] = vettoreIntervalliDaTonica[5] + vettoreIntervalli [3] -  
vettoreIntervalli [2] =  $7 + 6 - 4 = 11$

Quindi vettoreIntervalliDaTonica è formato dai valori: 0, 2, 4, 5, 7, 9, 11.

Questo vettore è essenziale per il calcolo di tutte le note successive, che vengono calcolate con la funzione calcolaNumeroNotaMidi(int, int).

```
int MainWindow::calcolaNumeroNotaMidi(int intervalloDiPartenza, int ottavaDiPartenza)
{
    int ottava = ottavaDiPartenza;

    // se il parametro vale 1 si suona la tonica, quindi la nota 0.
    int intervallo = intervalloDiPartenza - 1;

    // se l'intervallo è maggiore di una settima, salgo di ottava e sottraggo 8
    all'intervallo
    while (intervallo>7){
        ottava++;
        intervallo = intervallo - 8;
    }

    //la nota si ottiene come la tonica + l'intervallo a partire dalla tonica (in
```

```

semitoni) + l'ottava di riferimento per 12 (semitoni)
    int nota = vettoreNote[0] + vettoreIntervalliDaTonica[intervallo] +
ottava*12;

    return nota;
}

```

Questo metodo riceve in input l'intervallo (definito dalla distanza fra i punti caratteristici del volto) e l'ottava sulla quale vogliamo ottenere la nota, il valore restituito indica la codifica in MIDI della nota desiderata.

Numeri corrispondenti a ciascuna nota per i file MIDI:

<b>Ottava</b>	<b>Numero della Nota</b>											
	<b>DO</b>	<b>DO#</b>	<b>RE</b>	<b>RE#</b>	<b>MI</b>	<b>FA</b>	<b>FA#</b>	<b>SOL</b>	<b>SOL#</b>	<b>LA</b>	<b>LA#</b>	<b>SI</b>
<b>-1</b>	0	1	2	3	4	5	6	7	8	9	10	11
<b>0</b>	12	13	14	15	16	17	18	19	20	21	22	23
<b>1</b>	24	25	26	27	28	29	30	31	32	33	34	35
<b>2</b>	36	37	38	39	40	41	42	43	44	45	46	47
<b>3</b>	48	49	50	51	52	53	54	55	56	57	58	59
<b>4</b>	60	61	62	63	64	65	66	67	68	69	70	71
<b>5</b>	72	73	74	75	76	77	78	79	80	81	82	83
<b>6</b>	84	85	86	87	88	89	90	91	92	93	94	95
<b>7</b>	96	97	98	99	100	101	102	103	104	105	106	107
<b>8</b>	108	109	110	111	112	113	114	115	116	117	118	119
<b>9</b>	120	121	122	123	124	125	126	127				

I numeri corrispondenti alle note selezionate vengono memorizzati in un vettore di interi: vettoreNote[6], che viene passato alla funzione salvaMIDI(int,int), questa funzione riceve in ingresso anche il numero delle note che vogliamo salvare, questo parametro potrebbe essere ricavato tramite vettoreNote.size(), tuttavia è stato inserito nel caso in cui si volessero salvare sul file midi solo alcune note ed è stato utilizzato in fase di test e sviluppo. Il metodo salvaMidi() è responsabile della creazione del file midi.

### | 5.3 Impostazioni dell'applicazione

E' stato creato un pannello per memorizzare le preferenze dell'utente, in fase di sviluppo è stato utilizzato per modificare alcuni parametri per il calcolo delle note, in particolare l'ottava di partenza per ogni nota, che ora è reso statico e non modificabile; rimane la possibilità di impostare un percorso di default per il salvataggio del file midi.

Per memorizzare le impostazioni viene utilizzata la classe QSettings fornita da Qt, la quale salva le preferenze in un file che si trova in:

- \$HOME/Library/Preferences/ e /Library/Preferences/ in ambiente Mac OS
- \$HOME/.config/ e /etc/xdg/ in ambiente Linux
- HKEY\_CURRENT\_USER\Software\MySoft\ in ambiente Windows

### | 5.4 Deploy in ambiente Mac OS

Per il deploy dell'applicazione in ambiente Mac OS seguire la seguente procedura:

1. Effettuare il build del progetto (modalità release)
2. Aprire il terminale e posizionarsi nella cartella in cui è stato compilato il progetto
3. Utilizzare il tool per il deploy fornito con qt con il comando:

```
path_for_qt/version/clang_64/bin/macdeployqt MusicaHumana.app
```

Saranno generati dei file all'interno della cartella MusicaHumana.app

4. Eseguire sempre su terminale:

```
otool -L MusicaHumana.app/Contents/MacOS/MusicaHumana
```

```
Computer:build-MusicaHumana-Desktop_Qt_5_3_0_clang_64bit-Release giuseppe$ otool -L MusicaHumana.app/Contents/MacOS/MusicaHumana
MusicaHumana.app/Contents/MacOS/MusicaHumana:
    lib/libopencv_core.2.4.dylib (compatibility version 2.4.0, current version 2.4.10)
    lib/libopencv_highgui.2.4.dylib (compatibility version 2.4.0, current version 2.4.10)
    lib/libopencv_imgproc.2.4.dylib (compatibility version 2.4.0, current version 2.4.10)
    lib/libopencv_objdetect.2.4.dylib (compatibility version 2.4.0, current version 2.4.10)
    lib/libopencv_features2d.2.4.dylib (compatibility version 2.4.0, current version 2.4.10)
/Users/giuseppe/flandmark-master/bin/libflandmark/libflandmark_shared.dylib (compatibility version 0.0.0, current version 0.0.0)
@executable_path/../Frameworks/QtWidgets.framework/Versions/5/QtWidgets (compatibility version 5.3.0, current version 5.3.0)
@executable_path/../Frameworks/QtGui.framework/Versions/5/QtGui (compatibility version 5.3.0, current version 5.3.0)
@executable_path/../Frameworks/QtCore.framework/Versions/5/QtCore (compatibility version 5.3.0, current version 5.3.0)
/System/Library/Frameworks/OpenGL.framework/Versions/A/OpenGL (compatibility version 1.0.0, current version 1.0.0)
/System/Library/Frameworks/AGL.framework/Versions/A/AGL (compatibility version 1.0.0, current version 1.0.0)
/usr/lib/libc++.1.dylib (compatibility version 1.0.0, current version 120.0.0)
/usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current version 1197.1.1)
```

Saranno mostrate le dipendenze dell'applicazione, occorre modificare il percorso dei file delle librerie openCV e flandmark, in quanto quelle impostate sono locali e l'applicazione non funzionerebbe su macchine differenti da quella su cui si sta facendo il deploy.

5. Per modificare i percorsi eseguire i seguenti comandi da terminale:

```
install_name_tool -change lib/libopencv_core.2.4.dylib
@executable_path/../Frameworks/libopencv_core.2.4.dylib
MusicaHumana.app/Contents/MacOS/MusicaHumana;

install_name_tool -change lib/libopencv_highgui.2.4.dylib
@executable_path/../Frameworks/libopencv_highgui.2.4.dylib
MusicaHumana.app/Contents/MacOS/MusicaHumana;

install_name_tool -change lib/libopencv_imgproc.2.4.dylib
@executable_path/../Frameworks/libopencv_imgproc.2.4.dylib
MusicaHumana.app/Contents/MacOS/MusicaHumana;

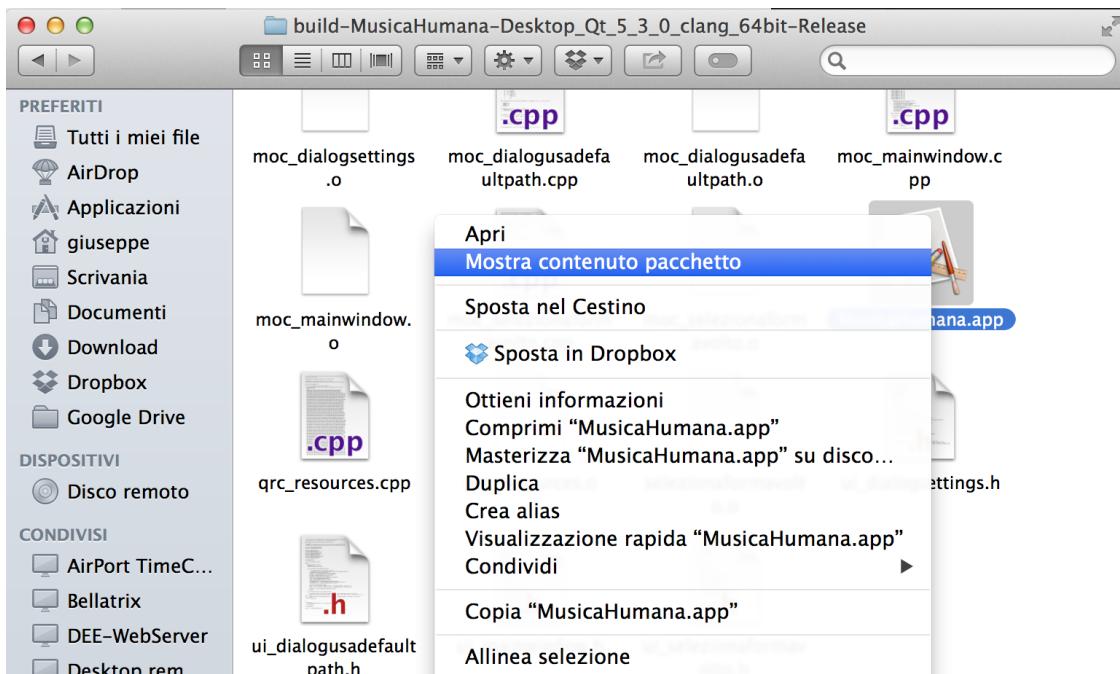
install_name_tool -change lib/libopencv_objdetect.2.4.dylib
@executable_path/../Frameworks/libopencv_objdetect.2.4.dylib
MusicaHumana.app/Contents/MacOS/MusicaHumana;

install_name_tool -change lib/libopencv_features2d.2.4.dylib
@executable_path/../Frameworks/libopencv_features2d.2.4.dylib
MusicaHumana.app/Contents/MacOS/MusicaHumana;

install_name_tool -change /Users/giuseppe/flandmark-
master/bin/libflandmark/libflandmark_shared.dylib
@executable_path/../Frameworks/libflandmark_shared.dylib
MusicaHumana.app/Contents/MacOS/MusicaHumana
```

6. Occorre modificare anche le dipendenze dei file della libreria, può essere fatto da terminale con la stessa procedura effettuata ai punti 4 e 5, oppure si può sostituire la cartella Frameworks con la cartella Frameworks presente nella cartella dei sorgenti, ottenuta da quella originale applicando install\_name\_tools -change ove necessario.

Per visualizzare la cartella Frameworks, aprire la cartella Contents visualizzabile cliccando con il tasto destro sull'applicazione e selezionando “Mostra contenuto pacchetto”.



7. Inserire all'interno dell'applicazione (con “Mostra contenuto pacchetto”), in Contents/Mac OS, i file flandmark\_model.dat, materelettrica.jpg e la cartella cascades, contenente gli xml dei classificatori utilizzati da OpenCV
8. Come ultimo passo, si può sostituire l'icona della applicazione selezionando con click destro MusicaHumana.app nella cartella dove è stato compilato il progetto e selezionando “Ottieni informazioni” (oppure cmd+i), si seleziona l'icona in alto a sinistra e con cmd+v si incolla l'immagine copiata precedentemente.