OMotion

A video-security system.

Università degli Studi di Napoli "Federico II"
Facoltà di Ingegneria - C.d.l. Ingegneria Informatica

Corso di Sistemi Multimediali (6CFU) - Prof. V. Moscato

Di Mauro Claudio S. - N46001636
Auriemma Salvatore - N46001675

LaTeX

Indice

Prefazione

Già da un po' di tempo, ormai, i sistemi di videosorveglianza stanno diventando parte integrante dei sistemi di controllo domestici e aziendali. Sono uno strumento indispensabile per tenere sotto controllo, in ogni momento, un'area specifica, sia essa un appartamento, un giardino, un ufficio o un qualsiasi altro ambiente.

L'importanza di tali sistemi, ci ha portato alla scelta della progettazione di tale applicazione, la quale rappresenta una base (quasi indispensabile) per la video-sicurezza grazie proprio alle caratteristiche di questo software.

Questo documento è stato redatto per esplicare le caratteristiche e le funzionalità del progetto **OMotion**, sviluppato per il corso di Sistemi Multimediali (6 CFU) tenuto dal Prof. V. Moscato presso la *Facoltà di Ingegneria Informatica dell'Università degli Studi di Napoli "Federico II"*.

Questo testo non vuole essere una vera e propria documentazione di quelle che sono le classi e i metodi che compongono l'applicazione, in quanto, nello specifico, è tutto appositamente descritto nelle pagine web (di cui in seguito è fornito il link) che rappresentano l'ufficiale descrizione della struttura di OMotion.

Il seguente documento non si prefigge lo scopo di essere una guida utile al lettore né per l'apprendimento del framework OpenCV, né per la programmazione Python ed è per tanto rivolto ad un pubblico avente almeno delle conoscenze basilari di questi tools di sviluppo.

1 Introduzione a Python e OpenCV

1.1 Python

Python è un linguaggio di programmazione messo a punto da Guido van Rossum agli inizi degli anni '90. È un linguaggio molto malleabile, è utile, infatti, per lo sviluppo software, lo scripting, la computazione numerica ed il testing. Python è un linguaggio multi-paradigma e i suoi obiettivi fondamentali sono la dinamicità, la flessibilità e la semplicità. È un linguaggio che supporta diversi paradigmi, ma per questo progetto ci interessa in particolare il paradigma Object-Oriented.

Le principali caratteristiche di Python sono le variabili non tipizzate e l'uso dell'indentazione per definire le specifiche. Altre caratteristiche da non sottovalutare sono: overloading degli operatori, funzioni tramite delegation, molti tipi, funzioni di base e numerosissime librerie ed anche una sintassi molto avanzata.

Il controllo sui tipi è effettuato a runtime. In altre parole, una variabile è un contenitore a cui è assegnata un'etichetta (nome) la quale può essere anche assegnata a differenti tipi di contenitori durante il suo tempo di vita. Python usa un garbage collector per avere un rilascio automatico della memoria.

Python ha alcune somiglianze con Perl, ma la sua sintassi è molto più essenziale e il codice molto più leggibile. Come Perl, Python è classificato come un linguaggio di scripting, ma la vastità di librerie per esso sviluppate lo hanno trasformato in un vero e proprio linguaggio di programmazione, usato per lo sviluppo di applicazioni software.

1.2 OpenCV e Computer Vision

OpenCV è un framework molto potente usato per la computer vision. È una libreria open-source scritta in C e lavora sulle immagini e sullo streaming video (anche real-time), girando tranquillamente su tutti i sistemi operativi.

La **computer vision** è la trasformazione di informazioni da una sorgente video verso una nuova rappresentazione.

Il framework OpenCV nacque da un'iniziativa Intel, mentre la Società lavorava per migliorare le loro CPU per consentirne l'uso nelle applicazioni intensive e nella proiezione 3D. Alcuni studenti stavano mettendo a punto del codice per la computer vision e proprio a partire da questo codice, Intel diede il via allo sviluppo di OpenCV.

Questa libreria è strutturata in sei parti:

- **CXCORE:** Contiene la definizione di tutte le strutture dati e dei metodi per l'handling delle immagini e dei video;
- **CV:** contiene tutti i metodi per l'analisi ed il processamento delle immagini, la calibrazione, il tracking e la pattern recognition;
- **ML (Machine Learinig):** contiene molti metodi per la Machine Learning, come il *clustering* e la *classificazione*;

- **HighGUI:** contiene la definizione per le interfacce utente (GUI);
- **CVCAM:** contiene le interfacce per la webcam;
- **CVAUX:** contiene alcuni algoritmi sperimentali usati per scopi differenti (per esempio: segmentazione, background subtractor, etc.).

2 Introduzione al progetto

2.1 Cenni e informazioni

Questo progetto tratta un sistema di videosorveglianza in cui sono rilevati i movimenti degli oggetti ripresi da una webcam, con lo scopo di evidenziare le anomalie che potrebbero presentarsi in una determinata area.

È stato sviluppato utilizzando come linguaggio di programmazione *Python3*, integrato con il framework *OpenCV*.

Il progetto segue le regole della modulazione e pertanto è diviso in più file in modo da rendere il codice il più leggibile possibile; in particolare esiste un file per ogni classe e per ogni classe sono stati eseguiti dei test case.

Ci siamo avvicinati a questo progetto utilizzando una metodologia *top-down*, cioè abbiamo inizialmente stilato quelle che dovevano essere le features principali del programma da sviluppare, in seguito abbiamo cominciato a definirne la struttura ed in fine ne abbiamo avviato lo sviluppo.

La scelta di questo tipo di progetto è stata fatta per coprire una parte del programma del corso di Sistemi Multimediali, nello specifico, ciò che riguarda la **Computer Vision**, in particolare trattando: *image processing*, *movement detection*, *frame control*, *color conversion*.

Un sistema di videosorveglianza è un caso tangibile di **information retrieval**. Con tale termine si intende la capacità di riconoscere e trovare informazioni da un contenuto. Una ricerca può essere fatta per trovare testo, immagini, video, ma anche cose più astratte come delle azioni sospette.

Il caso di OMotion, sistema di videosorveglianza con rilevazione dei movimenti, è un esempio lampante di tutto ciò: una videocamera ha un determinato campo di analisi su cui opera; nel momento in cui è rilevato un movimento, il software lo evidenzia e si adopera per registrarlo in un file di log.

Per ulteriori informazioni su questo progetto:

- https://github.com/csdm/OMotion
- https://www.claudiodimauro.it/OMotion/Documentation

3 OMotion

OMotion è un prototipo per un sistema di videosorveglianza, sviluppato per rilevare dei movimenti che potrebbero verificarsi in un'area ripresa da una webcam che fornisce immagini in tempo reale.

La qualità del risultato fornito è strettamente legata alla qualità dei dispositivi di acquisizione video usati per rilevare i movimenti. Si è infatti testato che in un dispositivo di scarsa qualità o comunque di vecchia costruzione (quindi più propensi alla rilevazione di rumore nei frame video), la detection dei movimenti è leggermente influenzata da questi disturbi; viceversa, usando dispositivi di media/buona qualità e di ultima generazione, si ottengono dei rilevamenti nitidi e precisi per qualunque movimento presente nell'area osservata.

Gli screenshot utilizzati in questo documento, per la descrizione del progetto, sono stati effettuati mentre si testava il programma, usando la webcam di un computer portatile non proprio di ultima generazione (Asus K53SV - anno 2012). Nonostante ciò abbiamo ottenuto risultati adeguati per effettuare una dimostrazione del buon funzionamento dell'applicazione.

3.1 Direttive di avvio

Essendo un prototipo non abbiamo ancora sviluppato l'applicazione eseguibile. In virtù di ciò è un programma che momentaneamente gira solo su sistemi Unix (MacOS - Linux) dai quali è possibile avviarlo utilizzando dei comandi per shell/terminale seguendo la procedura descritta:

1. Aprire un terminale e spostarsi nella directory all'interno della quale sono presenti tutti i file che compongono il programma, in particolare dove è presente il file **main.py**:

   ```
   cd ~/.../.../OMotion/src
   ```

2. Da questa posizione lanciare il comando per avviare l'interprete Python:

   ```
   python3 main.py
   ```

3. Fatto ciò verranno mostrate a video le quattro schermate che costituiscono OMotion e che in seguito verranno illustrate in questo documento.

> **NOTA:**
> per il corretto funzionamento della procedura sopra descritta, si necessita di avere installato nel sistema Python3 ed il framework OpenCV. In particolare questo progetto è stato sviluppato utilizzando Python 3.5 e OpenCV 3.1.0.

Questa applicazione prevede un **file di log** all'interno del quale vengono salvati tutti i movimenti rilevati, accompagnati da data, ora, coordinate cartesiane dei punti in movimento e area.

Per accedere a questo file, e quindi visionare tutti i movimenti rilevati, è possibile aprire con un qualsiasi editor di testo il file **motion.log** presente nel path

```
~/.../.../OMotion/log
```

Se si vuole aprire il file **motion.log** da terminale si possono utilizzare due procedure: la prima prevede soltanto l'apertura del file in lettura statica e mostra ciò che è già stato scritto sul file; la seconda mostra il contenuto del file in lettura dinamica, cioè se visualizzo il file mentre è in esecuzione OMotion è possibile leggere in tempo reale ciò che viene scritto sul file. Vediamole entrambe.

- **Lettura statica**:
  1. Aprire un terminale e spostarsi nel path dove è presente il file di log:
     ```
     ~/.../.../OMotion/log
     ```
  2. Aprire il file di log tramite il comando:
     ```
     cat motion.log
     ```

- **Lettura dinamica**:
  1. Aprire un terminale e spostarsi nel path dove è presente il file di log:
     ```
     ~/.../.../OMotion/log
     ```
  2. Aprire il file di log tramite il comando:
     ```
     tail -f motion.log
     ```

3.2 Caratteristiche principali e funzionamento

Come già precedentemente accennato, all'avvio di **OMotion** si apriranno quattro finestre, ognuna delle quali va a fornire una rappresentazione diversa di quanto ripreso dalla webcam.

In questa sezione andremo a vedere nello specifico il funzionamento di ogni finestra.

Si noti che per ottenere delle finestre, OCV fornisce delle funzioni adatte allo scopo:

```
show(var, name='nome_finestra')
```

dove **var** è una variabile all'interno della quale (almeno nel caso di OMotion) è inserito un array contenente l'insieme dei frames che compongono il video (o più in generale ciò che si vuole mostrare nella finestra), proprio per questo nel codice di OMotion questa variabile è identificata con **frame**.

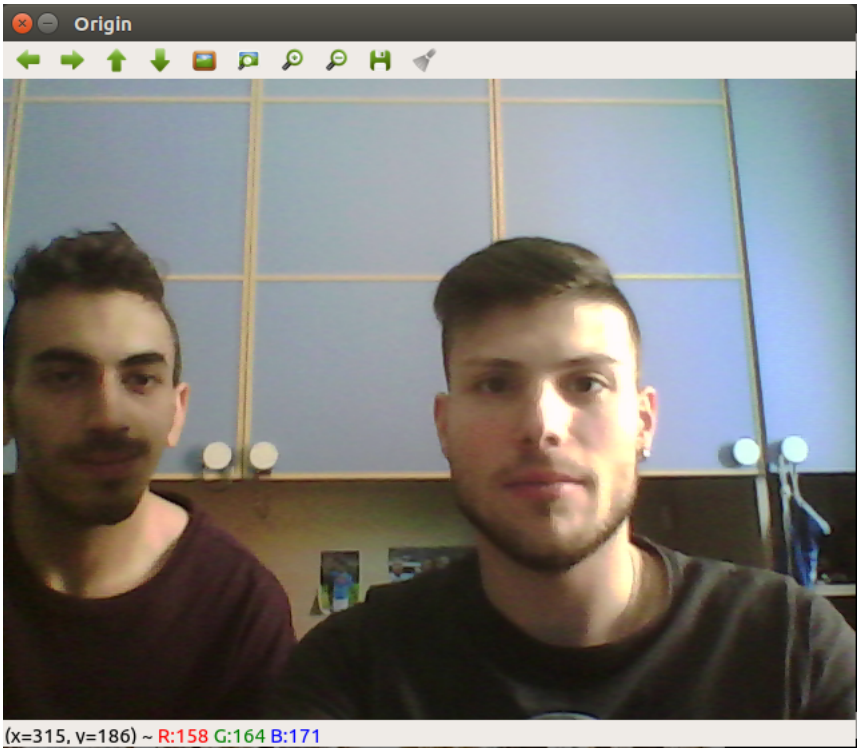
3.2.1 Origin

All'interno di questa finestra viene mostrato la sequenza di frames video così come catturati dalla webcam, senza alcuna modifica sulla composizione dell'immagine. Questo passaggio non richiede una approfondita conoscenza del framework OpenCV in quanto è un'operazione eseguibile richiamando delle semplici e basilari funzioni.

Per effettuare delle operazioni sulle immagini e sui video, OCV ci offre dei metodi già pronti all'uso facilmente utilizzabili studiando la sua documentazione. Nello specifico, abbiamo utilizzato una particolare funzione per ottenere la cattura delle immagini dalla webcam e trasferirle in una finestra:

```
cv2.videoCapture(0)
```

Con tale comando, si è detto all'interprete Python di ricercare il dispositivo di video-input connesso alla macchina, attivarlo e cominciare uno streaming video live e salvare temporaneamente in una variabile di buffer quanto ripreso.



Nonostante questa finestra sembri semplicemente la riproduzione reale di ciò che la telecamera riprende, è importante sottolineare che è proprio sui frames video mostrati da esssa che viene effettuata la ricerca dei movimenti, attraverso la funzione

```
motionsearch(self, frame)
```

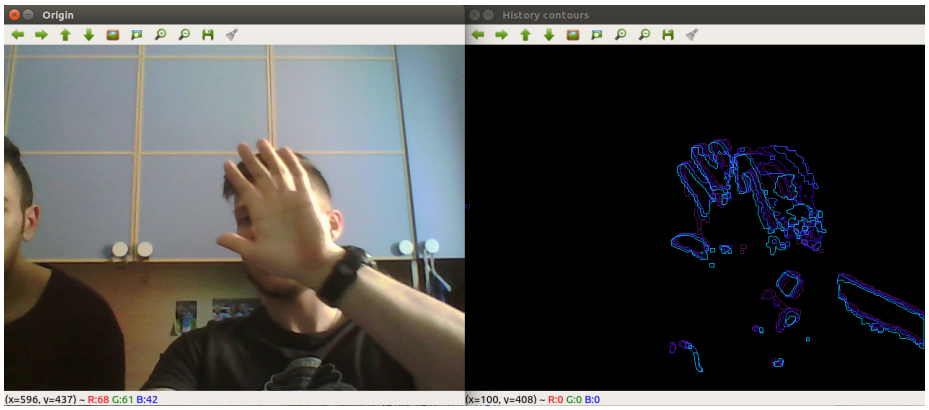
la cui implementazione può essere osservata direttamente dal codice sorgente (in particolare nel file con path *OMotion/src/lib/videocontroller/CameraController.py*).

3.2.2 History contours

Questa è la finestra in cui si inizia realmente a vedere il funzionamento di OMotion, infatti in tale finestra le immagini restituite rappresentano lo storico dei movimenti rilevati. Più precisamente in questa finestra si ha un background nero ottenuto tramite la funzione (sempre fornita da OCV)

```
np.zeros(size, dtype)
```

che rimuove i colori dallo sfondo e restituisce un background nero. Su questo background vengono disegnati i contorni di un oggetto che si muove e, seguendo i movimenti dello stesso, vengono disegnati e mantenuti a video (per un tempo limitato) i contorni dei movimenti rilevati frame per frame. Si è scelto di utilizzare un background nero al solo scopo di evidenziare maggiormente i contorni disegnati.



Per ottenere lo storico dei contorni si è implementato un metodo, appartenente alla classe CameraController, grazie al quale vengono disegnati il contorno corrente (cioè quello che evidenzia il momento esatto in cui il movimento è rilevato) e, quasi in contemporanea (dipende dalla velocità dei movimenti), i contorni dei movimenti precedentemente rilevati.

La funzione in questione è

```
drawContoursHistory(self, frame)
```

che prende in ingresso l'array di frames e richiama iterativamente, per ogni frame, il metodo OCV

```
cv2.drawContours(img, contours, -1, (0,255,0), 3)
```

che serve proprio a tracciare i contorni. Nel caso particolare di OMotion, questa funzione è stata così adattata:

```
cv2.drawContours(frame, cnt, -1, color, 1)
```

dove:

- **frame:** è un array contenente tutti i frame video ottenuti dalla webcam (è l'array principale per il funzionamento di OMotion);
- **cnt:** è una variabile in cui vengono salvati i contorni ottenuti tramite il metodo OCV

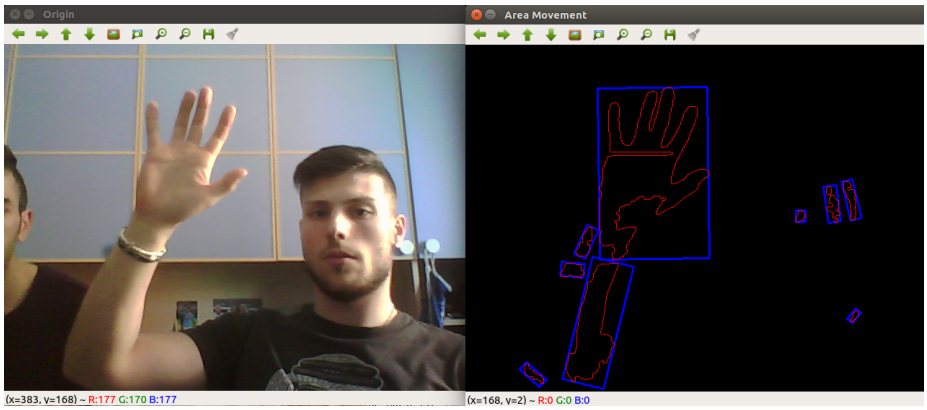
```
cv2.findContours(image, mode, method[,
    contours[, hierarchy[, offset]]])
```

- **-1:** è l'indice dei contorni, passando -1 si intende che si vogliono disegnare tutti i contorni;
- **color:** è un oggetto rappresentante l'array contenente un range di valori che variano tra 0 e 255 fatto in modo da ottenere colori differenti per ogni contorno disegnato su frame diversi. L'array da cui è istanziato **color** così definito:

```
colors = [
        ( 88,  24,  89),
        (144,  12,  63),
        (199,   9,  57),
        (255,  87,  51),
        (255, 195,   0)
    ].
```

3.2.3 Area Movement

Area Movement è il cuore del progetto. È infatti la finestra all'interno della quale vengono evidenziati con maggiore precisione i movimenti rilevati da OMotion. La complessità delle operazioni in essa mostrate, consiste nel fatto che, non solo vengono disegnati i contorni degli oggetti in movimento (nel momento esatto in cui il movimento avviene), ma in più vengono circoscritte le aree in cui tali movimenti avvengono, grazie a dei rettangoli di area ed angolazione variabile.



Come si può notare dall'immagine, OMotion disegna un contorno di colore rosso in modo da evidenziare il movimento rilevato. In aggiunta a ciò i rettangoli con bordo blu ne delimitano l'area (da ciò viene il nome della finestra).

NOTA:
è superfluo trattare nuovamente la composizione dei metodi per il disegno dei contorni, in quanto tale descrizione è già stata effettuata nel precedente paragrafo (cfr.§ 3.2.2).
Proprio in analogia con la finestra History contours, il contorno di colore rosso indica il movimento istantaneo.

Di rilevante importanza è vedere come tali aree vengono identificate. Per la creazione dei rettangoli sono state implementate diverse funzioni a partire da alcune basilari di OpenCV. Il framework offre una funzione

```
cv2.line(img, pt1, pt2, color, size)
```

che prende in ingresso le coordinate dei movimenti e disegna delle linee che congiungono tali punti.

Ciò che è stato fatto durante lo sviluppo di OMotion, è stato creare dei metodi per l'unione di tutte queste linee sotto forma di rettangoli, stando attenti a farli adattare all'angolazione del movimento (tramite apposite funzioni per la rotazione dei rettangoli). Inoltre si è anche scelto di assegnare un'area minima per la rilevazione dei movimenti, quindi, a meno del rumore prodotto dalla qualità della webcam, non ci saranno rettangoli di dimensioni molto piccole, in quanto potrebbero rappresentare dei movimenti non rilevanti al fine di un controllo di sicurezza (es. una mosca che vola o qualcosa spostato dal vento).

A partire dal metodo OCV

```
cv2.line(img, pt1, pt2, color, size)
```

è stata implementata una classe per il disegno di generici poligoni (Drawer.py):

```
import cv2

class Drawer:

        @staticmethod
        def drawPolygon(frame, polygon, color, size):
                edges = polygon.edges()

                for e in edges:
                        cv2.line(frame, e.p0.toCv(),
                        e.p1.toCv(), color, size)


        @staticmethod
        def drawPolygons(frame, polygons, color, size):
                for polygon in polygons:
                        Drawer.drawPloygon(frame, polygon,
                        color, size)
```

Tale classe prende un'ulteriore classe (**Rectangle**) definita proprio per la creazione di rettangoli, nel file Rectangle.py. Tale classe fornisce al drawer tutti i parametri necessari per il disegno di un rettangolo (centro, diagonale, area, perimetro, punti e lati).

La classe Rectangle è estesa da **RotatedRectangle**, la quale fornisce la possibilità di ri-disegnare un rettangolo applicandogli la giusta angolazione. Ovviamente è un'estensione della classe Rectangle perché tutti i metodi per tracciare un rettangolo sono già stati implementati.

3.2.4 Backg

È una finestra non rilevante ai fini di un controllo di sicurezza in quanto mostra a video l'effetto provocato dalla funzione OCV

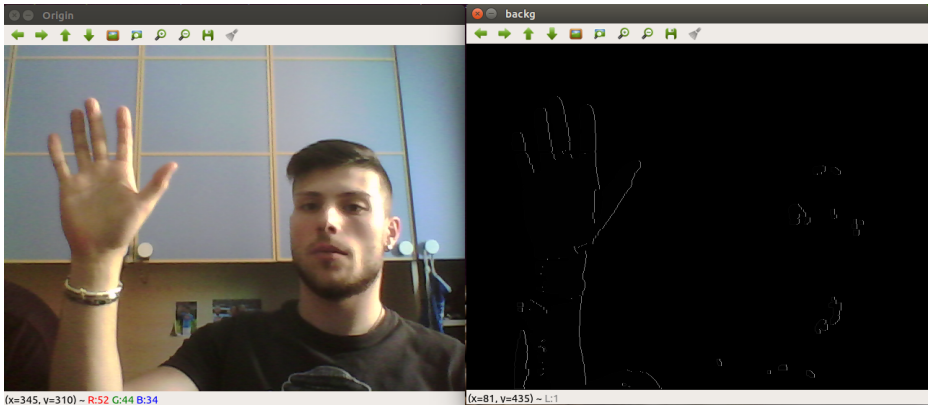
```
cv2.threshold(self.backg, 250, 255,
                cv2.THRESH_BYNARY).
```

Questa funzione effettua la sottrazione del colore del colore dallo sfondo, in particolare sulla variabile definita **backg**. Con il parametro **TRESH_BINARY** si è voluto specificare l'utilizzo di una sequenza binaria di colori (nero e un grigio -in scala-) definiti dai parametri centrali **250** e **255**.

Un ulteriore metodo OCV usato è

```
cv2.morphologyEX(self.backg, cv2.MORPH_OPEN,
                kernel)
```

che effettua un'operazione di pulizia del rumore. Il flag **MORPH_OPEN**, in particolare pulisce l'area all'esterno dei contorni, a differenza di **MORPH_CLOSE** che va a ripulire i bordi interni di un contorno evidenziato[1].



Questa finestra mostra le implementazioni effettuate per la sottrazione del colore dallo sfondo e la conversione da BRG[2] in scala di grigi.

Ovviamente, anche in questo caso sono state utilizzate le funzioni per il tracciamento dei contorni.

3.2.5 Log dei movimenti rilevati

Il log dei movimenti è un file che rappresenta un'ulteriore strategia di controllo che dimostra come il sistema tenga realmente traccia dei movimenti rilevati e dell'esatto momento in cui essi avvengono, con relative coordinate ed area.

Il file **motion.log** è consultabile al path

```
OMotion/log/motion.log
```

ed è visionabile utilizzando un qualsiasi editor di testo. Seguendo le direttive di avvio descritte al par. 3.1 è possibile leggere il contenuto da terminale.

[1] Per maggiori informazioni sull'uso di **morphologyEX()** si veda la documentazione di OpenCV.

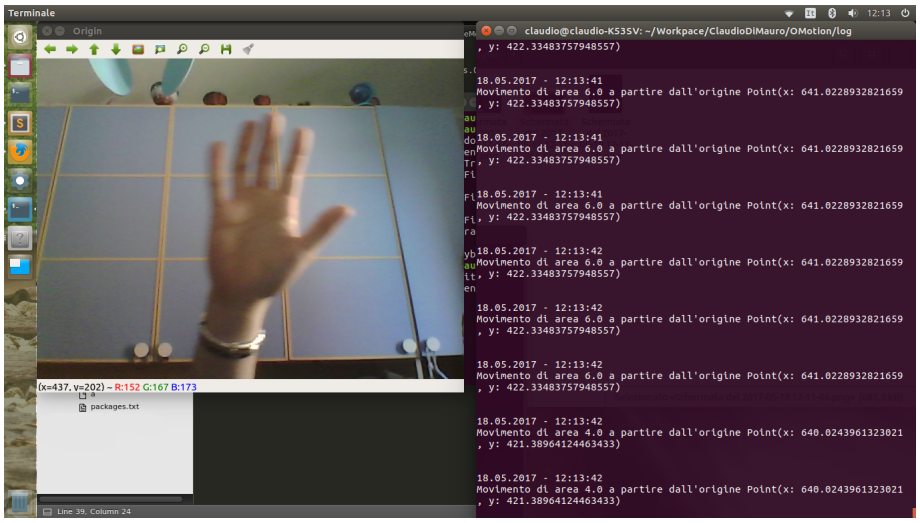
[2] OpenCV utilizza tante differenti rappresentazioni per il colore. In particolare i colori RGB vengono rappresentati in maniera opposta tramite la codifica BRG.

HiFileMotionLogger.py

```
import os
import time
import datetime

class HiFileMotionLogger (MotionLogger):

        FILE_PATH = os.getcwd()+'/../log/motion.log'
        log       = None
        logTime   = None


        def __init__ (self):

        def __del__ (self):
                if self.log:
                        os.close(self.log)

        def getTime(self):
            return
           datetime.datetime.now().strftime('%d.%m.%Y - %H:%M:%S')

        def logRotatedRect(self, rects):
                out = self.getTime() + '\n'

                for rect in rects:
                        o = str(rect.center())
                        a = str(rect.area())

                        out += 'Movimento di area ' + a +
                                ' a partire dall\'origine '
                               + o +'\n'

                with open(self.log) as file:
                        file.write(out)
```

Si possono osservare in questa classe (**HiFileMotionLogger.py**) che ci sono vari metodi. In particolare ci sono le funzioni Python per l'apertura, la scrittura e la chiusura di/su un file, la funzione per ottenere la data e l'ora di sistema, settate secondo lo standard italiano.

Per quanto riguarda la funzione di scrittura su file, vi è un richiamo alle funzioni di cui abbiamo già in precedenza parlato, cioè per la definizione delle coordinate dei rettangoli.



Il file di log, come si evince dalla immagine di cui sopra, mostra a video la data, l'ora, l'area e le coordinate del punto di origine del movimento.

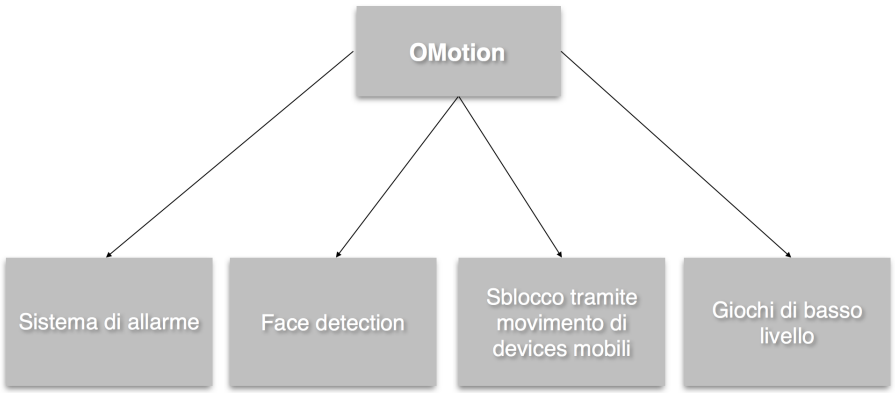
4 Possibili applicazioni e miglioramenti

Come già largamente descritto, OMotion è un sistema in via di sviluppo che si presta bene a varie tipologie di applicazioni, infatti opera bene sia come sistema di sicurezza che come analizzatore di movimenti. Insieme, queste due caratteristiche, possono costituire un valido sistema di videosorveglianza.

Il fatto che il sorgente di tale software sia disponibile a tutti gli utenti, lo rende davvero molto versatile. Integrandolo con ulteriori algoritmi può essere ampiamente migliorato ed utilizzato in vari settori.

Una possibile integrazione da implementare potrebbe essere la face-detection, cioè la possibilità di riconoscere volti a partire da un movimento e da un insieme di lineamenti di riferimento. Anche in questo caso OpenCV ci fornisce molte classi e metodi utili alla realizzazione di tale riconoscitore.

Un'ulteriore applicazione del software potrebbe essere quella di far interagire il codice con un sistema di allarme che scatterebbe in caso di movimenti rilevati e ritenuti sospetti. In questo caso conta molto la qualità del dispositivo di input video usato che non dovrà provocare rumore al fine di non falsare i risultati.



5 Licenza

OMotion è un software sviluppato per un progetto per la Facoltà di Ingegneria Informatica dell'Università degli Studi di Napoli "Federico II" ed è rilasciato sotto licenza **GNU General Public License v2**.

Ciò significa che il codice sorgente è rilasciato in maniera libera e può essere utilizzato da qualsiasi utente, sia per scopi commerciali che non. L'utilizzo di tale licenza prevede che qualora qualcuno modificasse il sorgente per apportarne dei miglioramenti, è tenuto a rilasciarlo in maniera libera e gratuita, in modo da renderlo fruibile per tutti gli utenti.

GNU GPL v2
Terms and condition for copying, distributions and modification

1. This License Agreement applies to any software library which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Library General Public License (also called "this License"). Each licensee is addressed as "you".

   A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

   The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

   "Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

2. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

3. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 2 above, provided that you also meet all of these conditions:

   - **a)** The modified work must itself be a software library.

   - **b)** You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

   - **c)** You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

   - **d)** If a facility in the modified Library refers to a function or a table of data to be supplied by an application

program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

4. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General

Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

5. You may copy and distribute the Library (or a portion or derivative of it, under Section 3) in object code or executable form under the terms of Sections 2 and 3 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 2 and 3 above on a medium customarily used for software interchange.

   If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

6. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

   However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 7 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 7.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 7. Any executables containing that work also fall under Section 7, whether or not they are linked directly with the Library itself.

7. As an exception to the Sections above, you may also compile or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

   You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

   - **a)**Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 2 and 3 above); and, if the work

is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

- **b)** Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

- **c)** If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

- **d)** Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

8. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such

a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

- **a)** Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

- **b)** Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

9. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

11. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

12. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), 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 distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

    If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

    It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

    This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

13. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of

this License.

14. The Free Software Foundation may publish revised and/or new versions of the Library 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 Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

15. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

16. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "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 LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU

ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

17. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY 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 LIBRARY (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 LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey 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 library's name and an idea of what it does.

Copyright (C) 2017

```
Di Mauro Claudio S. - Auriemma Salvatore

This library is free software; you can
redistribute it and/or modify it under
the terms of the GNU Library General Public
License as published by the Free
version 2 of the License, or
(at your option) any later version.

This library 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
Library General Public License for
more details.

You should have received a copy of the GNU
Library General Public License along with
this library; if not, write to the
Free Software Foundation, Inc.,
51 Franklin St, Fifth Floor,
Boston, MA 02110-1301, USA.
```

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

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright
interest in the library 'Frob' (a library for
tweaking knobs) written by James Random Hacker.

signature of Ty Coon, 1 April 1990
Ty Coon, President of Vice.
```