



Università
Ca' Foscari
Venezia

Corso di Laurea
In Informatica

Tesi di Laurea

Una Web Application per la Didattica della Computer Vision

Relatore

Ch. Prof. Filippo Bergamasco

Laureando

Fabio Dainese
Matricola 857661

Anno Accademico
2017 / 2018

Abstract

In questa tesi si è sviluppato una web application per fornire supporto all'insegnamento di alcune tecniche base di computer vision. Quest'applicazione nasce dal porting di un applicativo Java esistente, che dunque richiede la necessità di avere installato la JVM (Java Virtual Machine) per essere eseguito, ragion per cui la web app ne riprende ed espande le funzionalità precedentemente fornite in un contesto dove si richiedono meno vincoli possibili per l'utente per poterla utilizzare. Per rispettare queste caratteristiche nello sviluppo di questo progetto si sono utilizzate le più moderne tecnologie web per realizzare una SPA (Single-Page Application) fortemente interattiva gestita da semplici controlli sintattici sul codice prodotto dagli studenti sfruttando le caratteristiche fornite da linguaggi dinamici come Javascript.

Indice dei contenuti

| | |
|---|-----------|
| 1. Introduzione..... | 7 |
| 1.1. Scopo del progetto..... | 7 |
| 1.2. Requisiti | 8 |
| 2. Architettura..... | 9 |
| 2.1. Componenti..... | 9 |
| 2.1.1. Webcam..... | 10 |
| 2.1.2. Editor di scripting | 12 |
| 2.2. Librerie | 13 |
| 3. Workflow..... | 15 |
| 3.1. Homepage | 15 |
| 3.2. Procedura “Chroma Key” | 19 |
| 3.3. Procedura “Buttons Detection” | 21 |
| 4. Conclusioni | 23 |
| 5. Bibliografia/Appendice | 25 |

1. Introduzione

Negli ultimi anni, grazie al fatto del continuo aumento d'interesse rivolto da esperti e non nel campo della computer vision e anche dal sempre costante incremento delle prestazioni offerte dai personal computer, ci sono stati degli sviluppi considerevoli in quest'ambito che hanno portato al rilascio di nuove librerie software che tra l'altro hanno ampliato il raggio di copertura delle stesse anche per lo sviluppo di applicazioni web real-time.

Queste librerie, infatti, sfruttano le più moderne tecnologie web che offrono metodi d'interfacciamento ed elaborazione delle immagini sempre più performanti e con una richiesta di potenza computazionale sempre più esigua; fattore cruciale per cui questo progetto nasce con l'intento di creare una SPA (Single Page Application) rivolta all'introduzione di alcune tecniche di base di computer vision.

1.1. Scopo del progetto

Come anticipato nello scorso paragrafo questo progetto nasce con l'esigenza di creare un'applicazione fortemente interattiva per fornire un supporto all'insegnamento di alcune tecniche base di computer vision, rifacendosi alle funzionalità, ed ampliandole, di un applicativo Java precedentemente sviluppato da parte di altri studenti dell'università di Ca' Foscari, sviluppandolo però questa volta sotto forma di web-app.

Si è sentita questa necessità di compiere il porting di quest'applicazione poiché si voleva rendere il più possibile indipendente il programma dalla piattaforma e dai vincoli richiesti da quest'ultima per essere eseguito.

L'approccio dell'apprendimento offerto da quest'app si rifà al concetto della teoria educativa "learning by doing", letteralmente "imparare facendo", ovvero la spiegazione di nuove conoscenze viene presentata sotto forma pratica ed interattiva.

Nel nostro caso specifico all'utente viene richiesto il completamento di una serie di task, da sviluppare sotto forma di script (in JavaScript), per aver alla fine tutta l'applicazione funzionante.

L'interazione uomo-macchina avviene attraverso l'uso di una particolare striscia colorata composta da due colori ben distinti fra di loro e disposti come in Figura 1.



Figura 1 – Striscia colorata usata per interagire con l'applicazione.

Sfruttando l'utilizzo della webcam e di particolari librerie software, descritte in seguito, si potrà catturare e analizzare, attraverso scripting, il comportamento

(posizione, inclinazione, etc.) di questa striscia colorata in modo da assolvere i vari task richiesti.

L'analisi dettagliata dei vari task e di come questi ultimi possono influire sull'aspetto finale della web-app verranno descritti nella sezione "3 – *Workflow*".

1.2. Requisiti

L'applicativo è stato sviluppato tenendo ben in mente l'user experience a tutto tondo, per cui nello studio di fattibilità compiutisi nelle fasi iniziali si è deciso di fissare dei requisiti minimi che l'applicazione dovesse rispettare in tutte le sue fasi evolutive.

Questi requisiti, infatti, hanno avuto un grosso impatto nella scelta delle tecnologie e nei metodi implementativi usati durante tutto lo sviluppo della web-app.

I requisiti principali su cui ci si è focalizzati sono riassumibili nelle seguenti categorie:

- **Performance:** Vista la natura del tipo di applicazione da sviluppare è stato essenziale prestare la massima attenzione sul lato delle prestazioni, poiché tutte le varie interazioni possibili da parte dell'utente, tramite utilizzo della webcam e/o scripting, richiedevano una risposta (output visivo) in real-time, per cui sono state fatte delle scelte progettuali fortemente influenzate da questo fattore che verranno analizzate in maniera dettagliata nei successivi paragrafi.
- **User Interface/Experience:** Dopo un'attenta analisi preliminare sulla tipologia di end-user per cui l'applicazione è rivolta e considerato anche il suo utilizzo prevalentemente su dispositivi desktop, si è deciso di replicare il vecchio design presente sull'applicativo in Java. Tuttavia lo sviluppo della UI è stato eseguito rispettando i principi del responsive design, perciò se in futuro si volesse ampliare l'utilizzo della web-app anche su dispositivi mobili, quali smartphone e/o tablet, basterebbe aggiustare di conseguenza i vari comportamenti delle diverse componenti grafiche senza ridefinire l'intera struttura dell'applicazione.
- **Portabilità:** Uno dei punti cardine per cui si è voluta progettare una web-app è il fatto di poter renderla indipendente dalla tipologia di piattaforma su cui l'utente andrà a utilizzarla e di ridurre il più possibile i vari vincoli di pacchetti software richiesti per poterla usufruire (es. per eseguire l'applicativo Java è richiesta la presenza della Java Virtual Machine).

2. Architettura

In questa sezione si andranno ad analizzare in maniera approfondita tutte le varie scelte progettuali intraprese che hanno portato alla realizzazione della web-app, con una particolare attenzione ai componenti principali messi a disposizione dall'applicativo e dalle librerie software utilizzate.

2.1. Componenti

Nelle fasi antecedenti dello sviluppo vero e proprio della web-app è stato stabilito che l'applicativo dovesse essere costruito rispettando una certa gerarchia strutturale. Questa scelta nasce dal semplice motivo di garantire una buona scalabilità dell'intero sistema in caso di future modifiche e/o update, per cui permette di integrare le nuove componenti senza intaccare quelle preesistenti.

La gerarchia sviluppata è composta principalmente da due livelli, il primo definisce le macro-sezioni dell'applicazione, mentre il secondo è impiegato per la definizione di tutti i micro-componenti che costituiscono il livello superiore della gerarchia.

Questo diverso livello di dettaglio che l'applicativo utilizza si tramuta per l'utente in una struttura grafica composta da schede, dette tab, ognuna delle quali permette di interagire con un singolo componente e/o funzionalità del sistema alla volta.

Il secondo livello di dettaglio della gerarchia, che va a specificare i singoli micro-componenti, è realizzato tramite l'introduzione di box grafici.

Il legame gerarchico che collega queste due tipologie di livelli di specializzazione è definito tale che ogni scheda (tab) sia composta da un certo numero di box, come illustrato in Figura 2.

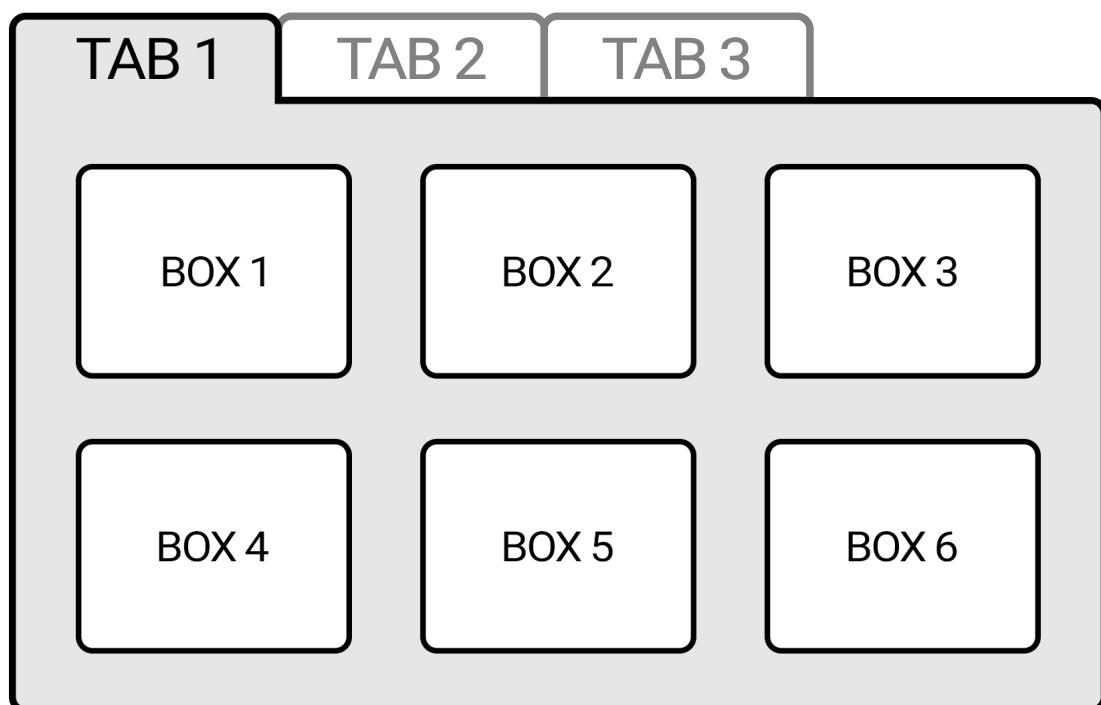


Figura 2 – Versione semplificata del layout utilizzato nell'applicazione.

Quando l'utente compie una certa interazione con il sistema, in altre parole aggiorna i campi dei singoli box, questi ultimi possono influenzare il comportamento di altre tab oltre a quella attualmente in uso.

Nei prossimi sotto paragrafi andremo a esaminare quali scelte progettuali sono state intraprese e come sono state realizzate inerenti ai principali micro-componenti della web-app.

2.1.1. Webcam

La webcam è una delle parti fondamentali dell'applicazione, nonché componente vitale messa a disposizione all'utente per interagire con tutto il sistema, per cui se ne richiede il possesso e l'autorizzazione ad usarla.

Il metodo implementativo utilizzato per catturare lo stream video della webcam (built-in o via usb) sfrutta la tecnologia WebRTC, ampiamente diffusa e supportata da tutti i maggiori browser in commercio.

Questo framework open-source permette, oltre all'acquisizione di audio e video, di specificare tramite l'utilizzo delle API Javascript ulteriori parametri e/o configurazioni che si vogliono apportare ai vari stream in entrata.

Nel nostro caso richiediamo di analizzare solamente la sorgente video messa a disposizione, disabilitando il corrispettivo comparto audio. Nel caso qualora qualcosa non andasse per il verso giusto durante questa fase, per esempio non fosse disponibile nessuna sorgente video o non si fosse acconsentito all'utilizzo della webcam (Figura 3), il sistema stamperà sulla console del browser e dell'applicativo dei messaggi d'aiuto per l'utente.

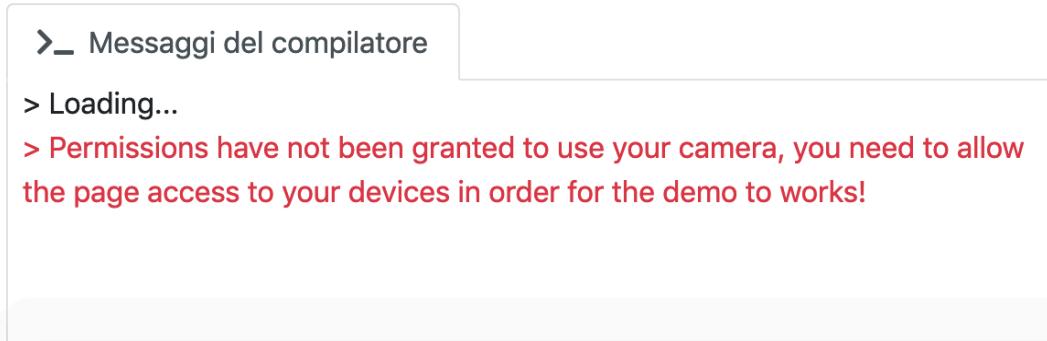


Figura 3 – Esempio di messaggio d'errore nel caso in cui non si fosse acconsentito all'uso della webcam

Per fornire un feedback visivo all'utente sullo stato della webcam si è deciso di mostrare il flusso video acquisito dalla webcam su un apposito canvas (Figura 4 e 5).



Figura 4 – Errore “No signal” (nessuna webcam trovata).

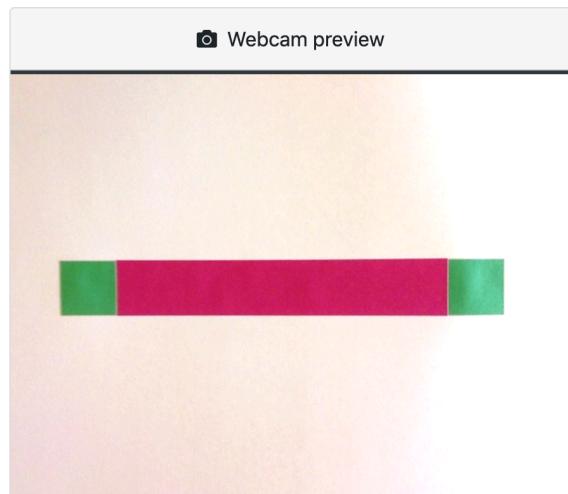


Figura 5 – Webcam in funzione e visione del relativo input.

Inoltre sono stati messi a disposizione svariati filtri grafici (Figura 6) da poter applicare allo stream video in modo da modificare le immagini acquisite migliorandole e facilitando l'utilizzo delle varie funzioni richieste dall'applicazione. Questi filtri grafici permettono la manipolazione di:

- **Saturazione (saturation):** Nel riferimento a immagini e/o video la **saturazione** è l'intensità di una specifica tonalità. Da sottolineare che alla diminuzione della saturazione corrisponde un progressivo affievolimento dei colori sino al raggiungimento della tonalità grigia.
- **Contrasto (contrast):** Nel riferimento a immagini e/o video il **contrasto** è il rapporto fra il valore del punto più luminoso e quello più scuro di un'immagine.
- **Luminosità (brightness):** Nel riferimento a immagini e/o video la **luminosità** è il quantitativo totale di luce che una sorgente e/o superficie appare emettere/riflettere.
- **Nitidezza (sharpness):** Nel riferimento a immagini e/o video la **nitidezza** è definita dal livello di contrasto che hanno due zone di diverso colore. Più il contrasto tra due zone è marcato, più queste ultime sembreranno maggiormente definite per l'occhio umano.

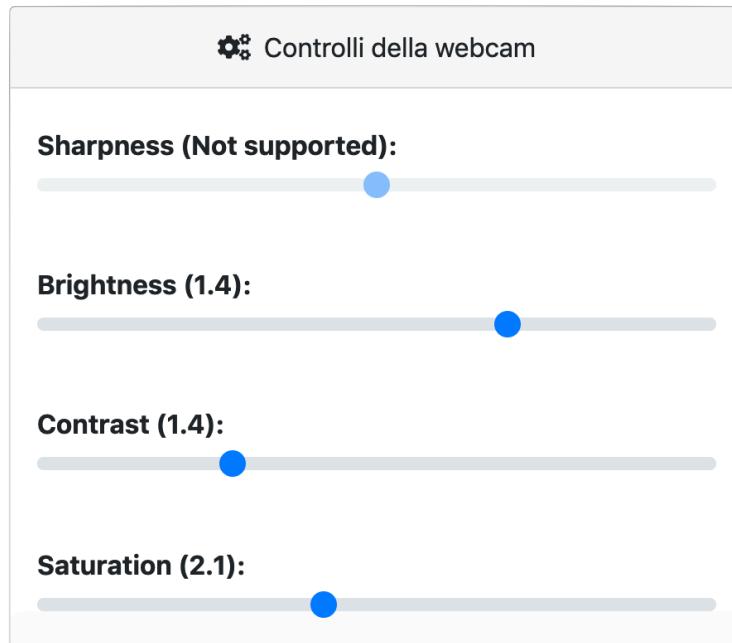


Figura 6 – Input di tipo 'range' per la manipolazione dei vari filtri video.

L'implementazione usata per i diversi filtri da applicare allo stream video si basa su filtri CSS (Cascading Style Sheets). Si è deciso di utilizzare questo metodo implementativo anziché usufruire delle API di WebRTC (**Media Streams API**), che permettono di modificare i dati direttamente dalla fonte, quindi dalla webcam, poiché gran parte dei browser non supportano in parte e/o completamente queste specifiche features.

2.1.2. Editor di scripting

Il secondo metodo che l'utente ha a disposizione per interagire con la web-app è fornito da una serie di editor di sviluppo. Questi editor permettono di scrivere, in piena libertà, del codice Javascript/jQuery che verrà poi allegato all'applicativo con lo scopo di assolvere a una serie di task che andranno ad influire sull'aspetto generale dell'intera applicazione.

Ogni editor presente sull'applicativo si basa sul progetto open-source Ace. La libreria in questione permette la creazione e gestione di multipli editor di sviluppo standalone in ambito web.

Si è scelto di utilizzare questa libreria principalmente perché mette a disposizione tutte le features che si trovano sui moderni editor di sviluppo come Sublime, Vim, etc. senza però sacrificare le performance generali.

Le principali features di Ace che sono state utilizzate per offrire un'esperienza di utilizzo al meglio sono state:

- **Syntax Highlighting**: Utilizzato ampiamente negli editor di testo odierni, specialmente quelli legati all'ambito di sviluppo, permette di mostrare il

codice sorgente in colori e font differenti a seconda delle keyword scritte. Uno dei benefici principali nell'uso di questo metodo è incrementare la leggibilità del codice, in questo caso JavaScript, fornendo aiuti grafici tra i quali quello di mostrare i vari errori sintattici commessi.

- **Autocompletamento live del codice:** Questo metodo consiste nel predire il resto di una parola/frase che l'utente ha intenzione di scrivere solamente anche dopo pochi caratteri digitati. Tutto questo va a beneficio di una più veloce interazione uomo-macchina, che nel nostro caso specifico si tramuta, nello sviluppo di codice Javascript.
- **Auto-indentazione:** Metodo estremamente diffuso che consiste nell'utilizzare spazi/tab per posizionare ogni riga di codice a seconda della propria logica strutturale, oltre che aumentare la leggibilità generale del programma.

Oltre alle sopracitate caratteristiche se ne sono volute creare altre completamente personalizzate per permettere un'interazione semplice e veloce. Tra queste features possiamo trovare: salvataggio tramite download del proprio codice (pulsante “*Salva*”), compilazione/incorporamento dello script alla web-app (pulsante “*Compila*”) e la possibilità di riportare il codice scritto allo stato originale (pulsante “*Ricarica default*”).

2.2. Librerie

Nello sviluppo dell'applicativo si sono utilizzate una serie di librerie, ovvero delle collezioni di risorse software, che hanno permesso l'implementazione di componenti chiave della web-app senza doverne creare di nuove e completamente custom.

Nello specifico si sono utilizzate le seguenti librerie:

- **Bootstrap (v4.3):** Framework che permette la creazione di siti web responsive grazie all'utilizzo di un sistema di layout a griglia e componenti grafiche pre-costruite.

Si è voluto ugualmente utilizzare questa libreria seppur la web-app non targetizzi chiaramente i dispositivi mobile, poiché oltre a fornire una facile implementazione del front-end permette la possibilità di gestire le componenti grafiche anche per dispositivi con ridotta dimensione dello schermo (mobile) senza dover riscrivere e/o aggiungere alcun componente, ma bensì manipolandone i rispettivi comportamenti tramite le apposite classi HTML messe a disposizione, a beneficio di un possibile futuro update dell'applicativo.

- **Ace**: Questa libreria è stata utilizzata in modo intensivo per quanto riguarda l'implementazione e gestione dei vari editor di sviluppo presenti sull'applicativo. Le varie scelte che hanno portato all'utilizzo di questa specifica libreria software sono riconducibili al fatto che sia liberamente utilizzabile (free to use), mette a disposizione le principali features presenti nei più moderni editor nativi e il tutto con un ridotto consumo di risorse computazionali.

Per una maggiore analisi sull'utilizzo di questa libreria nella web-app si consiglia di consultare la sezione “**2.1.2 – *Editor di scripting***” di questo documento.

- **OpenCV (Open Source Computer Vision Library)**: Questa libreria mette a disposizione una vasta gamma di algoritmi per l'elaborazione di immagini e video appositamente studiati per le applicazioni real-time e con un focus particolare per l'efficienza computazionale.

OpenCV non mette a disposizione direttamente una versione JavaScript della propria libreria, per cui per poterla applicare e utilizzare in un ambiente web si è dovuto utilizzare '**emscripten**', un compiler LLVM-to-JavaScript (LLVM: Low Level Virtual Machine), che ha permesso di convertire il bicode LLVM, generabile dai file scritti in C/C++ di OpenCV, in **asm.js**.

Asm.js è un sottoinsieme a basso livello altamente ottimizzato di JavaScript che permette quindi una compilazione ed ottimizzazione del codice offrendo una velocità prestazionale quasi paragonabile ad una nativa.

Il fattore decisivo per cui si è voluta utilizzare questa libreria, invece di sviluppare delle funzioni ad-hoc direttamente in JavaScript, è legato, come ribadito in molteplici occasioni, al fatto delle prestazioni generali, infatti, senza questa libreria il carico di lavoro non permetteva un'adeguata esperienza d'utilizzo della web-app (il frame rate della webcam si attestava a circa 1 fps, in confronto dei 30 attuali).

3. Workflow

In questa sezione si andranno ad analizzare in maniera approfondita tutti i possibili use case dell'applicazione, ciò significa che andremo ad esaminare tutte le possibili interazioni che possono avvenire fra l'utente e l'applicativo.

3.1. Homepage

Al primo avvio, la schermata principale della web-app si presenta come in Figura 7

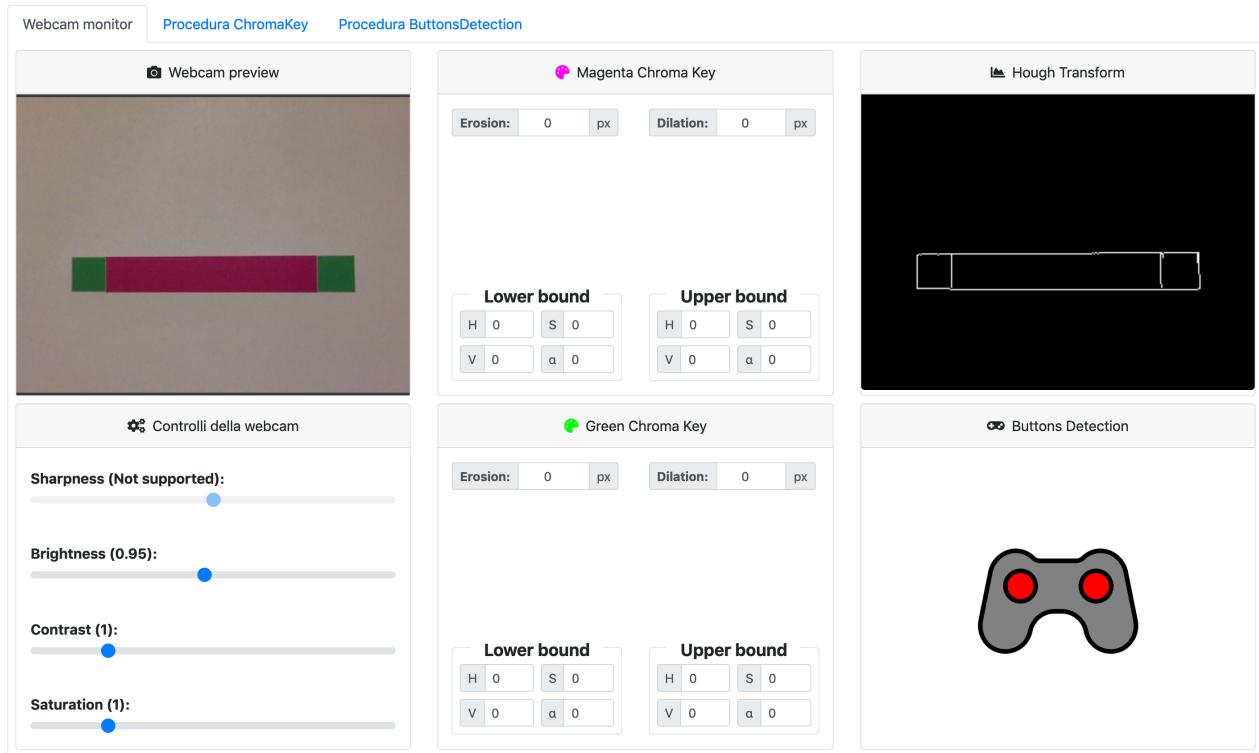


Figura 7 – Home page all'avvio dell'applicativo

L'homepage dell'applicazione è stata strutturata in modo tale che sia un hub di tutto lo stato del sistema, ovvero ogni qualvolta si interagisce con una determinata scheda/tab legata ai vari task da compiere, la schermata principale si aggiornerà mostrando in tempo reale all'utente un feedback visivo sul risultato delle nuove modifiche apportate.

Come si può apprezzare dalla precedente immagine, la schermata principale è composta da sei box, ognuno dei quali ricopre le seguenti funzioni:

- Webcam Preview:** Questo canvas è stato destinato alla sola rappresentazione real-time delle immagini catturate dalla webcam, in questo modo l'utente può facilmente capire in qualsiasi momento quali immagini stia elaborando l'applicativo.
Come anticipato nella sezione “2.1.1 - Webcam”, le uniche modifiche che l'utente può apportare in modo tale da cambiare/alterare l'aspetto finale delle

immagini dello stream avviene tramite l'utilizzo dei filtri grafici applicabili presenti nel box “**controlli della webcam**”.

- **Controlli della webcam:** Come già preannunciato questo box è riservato alla manipolazione delle immagini catturate dalla webcam attraverso l'utilizzo di filtri CSS, per cui le uniche interazioni possibili avvengono tramite drag-and-drop dei vari input range.

Si è voluto inoltre fornire un feedback testuale (Figura 8 e 9) relativo ad ogni filtro disponibile, che può assumere un valore numerico (valore attuale) nel caso sia stato possibile applicare quel determinato filtro, altrimenti in caso contrario verrà visualizzata una stringa d'errore (es. “Not supported” per il filtro nitidezza, poiché non esiste tale filtro in ambito CSS. Si è voluto comunque lasciare questo filtro poiché qualora i moderni browser supporteranno a pieno le API WebRTC si potrà usufruire anche di questo filtro).

Brightness (1.1):

Figura 8 – Valore attuale applicato allo stream video riguardante la luminosità

Sharpness (Not supported):

Figura 9 – Messaggio d'errore in caso in cui il filtro non fosse supportato

I vari intervalli di valori ammissibili da parte di ogni filtro supportato sono definiti quanto segue:

- Per quanto riguarda la ‘luminosità’ i valori concessi variano da un minimo di 0, che equivale ad un’immagine tutta nera, sino ad un valore massimo di 2. Il valore di default, corrispondente all’immagine originale, viene impostato a 1. La granularità messa a disposizione prevede degli step di incremento/decremento di 0.05 .
 - I valori relativi al ‘contrasto’ variano da un range che parte da 0 fino ad un massimo di 5. Il valore di default, corrispondente all’immagine originale, viene impostato a 1. La granularità messa a disposizione prevede degli step di incremento/decremento di 0.05 .
 - Infine i valori relativi alla ‘saturazione’ variano da un range che parte da 0 fino ad un massimo di 5. Il valore di default, corrispondente all’immagine originale, viene impostato a 1. La granularità messa a disposizione prevede degli step di incremento/decremento di 0.1 .
- **Magenta/Green Cromakey:** Questi due canvas vengono impiegati per la rilevazione e visualizzazione, in maniera separata, dei due colori costituenti la striscia di input (vedasi ‘Figura 1’ – Colori magenta e verde) utilizzata da parte dell’utente.
Sono stati resi disponibili per ognuno dei due box una serie di input (lower bound, upper bound, erosion e dilation – Figura 10 e 11) che permettono

all'utente, cambiandone i propri valori, di poter apportare modifiche sostanziali l'immagine resa disponibile dalla webcam.

| | |
|--------------------|-----|
| Lower bound | |
| H 0 | S 0 |
| V 0 | a 0 |

| | |
|--------------------|-----|
| Upper bound | |
| H 0 | S 0 |
| V 0 | a 0 |

Erosion: 0 px

Dilation: 0 px

Figura 10 – Campi input 'lower' e 'upper' bound

Figura 11 – Campi input 'erosion' e 'dilation'

Nello specifico i campi mostrati in Figura 10, come suggeriscono le rispettive etichette descrittive, servono a limitare il range cromatico dell'immagine catturata dalla webcam in modo tale da poter visualizzare solamente i colori che si trovano all'interno dell'intervallo di valori specificato.

Come rappresentazione del colore si è scelto modello HSV (Hue, Saturation, Value) che permette l'identificazione in maniera univoca di un determinato colore specificandone i propri parametri di colore (0 - 180), saturazione (0 - 255) e valore (0 – 255) come illustrati nello scema di Figura 12.

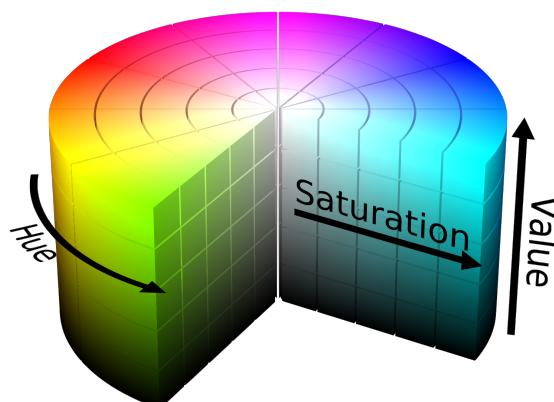
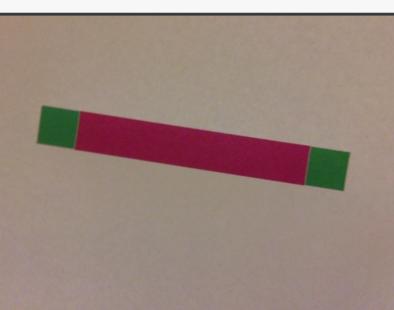


Figura 12 – Modello colore HSV

Ad applicativo appena avviato tutti questi campi sono impostati a valore zero, ossia identificano solamente il colore nero; sarà poi a cura dell'utente impostare in modo opportuno ciascuno di questi input cosicché i relativi canvas mostrino solamente uno dei due colori della striscia di input (vedasi Figura 13 – Magenta: [(120,50,70),(135,255,255)]; Verde: [(36,10,25),(75,255,255)]).

Webcam preview



Magenta Chroma Key

Erosion: 13 px Dilation: 13 px



Lower bound: H 120 S 50
V 70 a 0

Upper bound: H 135 S 255
V 255 a 0

Green Chroma Key

Erosion: 11 px Dilation: 11 px



Lower bound: H 36 S 10
V 25 a 0

Upper bound: H 75 S 255
V 255 a 0

Figura 13 – Rappresentazione filtraggio cromatico corretto

Si noti inoltre che per togliere il possibile rumore presente nelle immagini visualizzate nei canvas si sono resi disponibili ulteriori due campi input (Figura 11) che permettono di ridurre (**erosion**) e in seguito espandere (**dilation**), dello specificato numero di pixel, le sezioni d'immagine filtrate dagli input '**lower bound**' e '**upper bound**'.

- **Buttons Detection:** Questo box viene utilizzato per rappresentare, in versione stilizzata, un joypad che, nel caso in cui venisse completato in maniera corretta il task richiesto nella scheda "**Procedura Buttons Detection**", si animerà in modo tale da riprendere fedelmente l'inclinazione e lo status dei due quadrati di colore verde della striscia di input.
Tutti i possibili stati che il joypad può assumere sono riassunti nella seguente Figura 14:

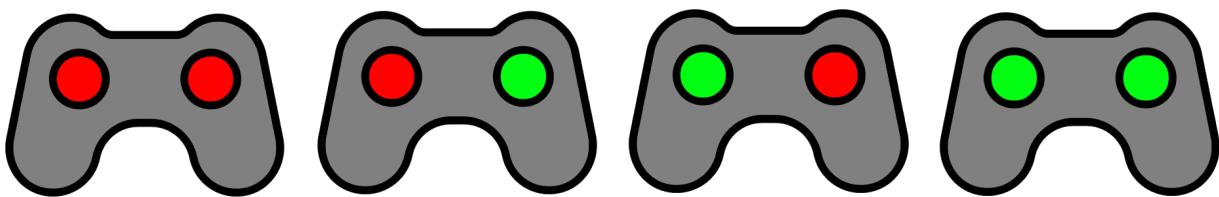


Figura 14 – Tutte le possibili rappresentazioni/status del joypad

Nel caso in cui l'utente non impletasse il task richiesto (stato di default) o, in caso contrario, coprisse i due quadrati di colore verde con le mani, il joypad assumerebbe una configurazione dei pulsanti tutta rossa, qualora invece si coprisse soltanto uno dei due quadrati verdi, i pulsanti del joypad si colorerebbero in modo alternato, di verde e di rosso, concorde con la striscia di input. Infine, nel caso in cui fossero visibili entrambi i quadrati verdi della striscia di input e si avesse implementato correttamente la funzione (task) richiesta, il joypad assumerebbe una configurazione dei pulsanti tutta verde.

- **Hough Transform:** In quest'ultimo box (si faccia riferimento alla Figura 7, box in alto a destra) troviamo una la rappresentazione grafica della trasformata di Hough, una tecnica molto diffusa nell'ambito della computer vision per l'individuazione di forme anche nell'eventualità che queste ultime siano un po' distorte grazie ad un sistema di votazione (l'immagine viene rappresentata in una matrice inizialmente impostata con tutti i campi a zero, poi mano che l'algoritmo avanza incrementa le celle della matrice corrispondenti ai punti dell'immagine che identificano i vertici di una possibile figura/forma).

Inizialmente si era ipotizzato un task in cui veniva richiesto all'utente l'implementazione dell'algoritmo della trasformata di Hough, ma per cause prestazionali si è deciso infine di togliere questa parte interattiva per salvaguardare l'esperienza di utilizzo dell'intera web-app, lasciando però la rappresentazione come spunto di riflessione per l'utente.

Una volta che l'utente avrà completato in maniera corretta entrambi i task richiesti, che verranno illustrati in maniera esaustiva nelle successive sezioni di questo documento, la homepage dell'applicativo si presenterà nella sua forma completa come mostrato in Figura 15.

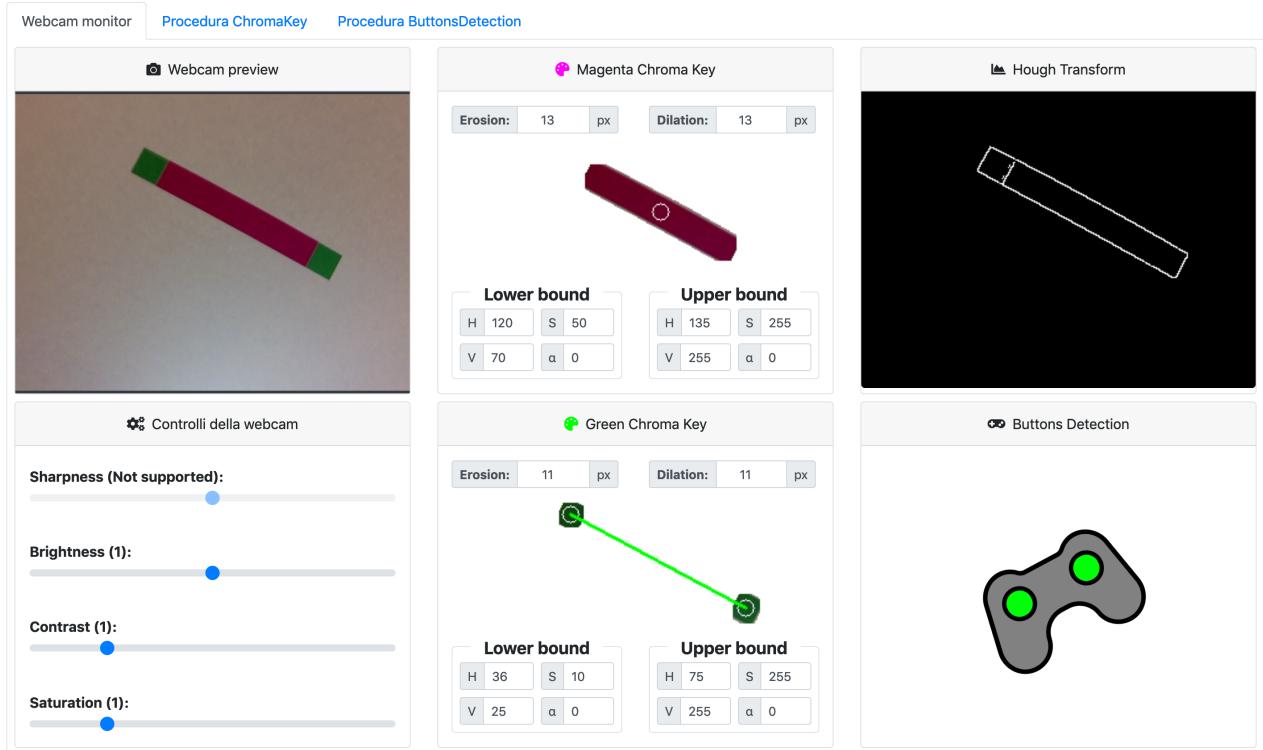


Figura 15 – Homepage nel caso in cui entrambi i task siano stati completati in maniera corretta.

3.2. Procedura “Chroma Key”

Questa seconda scheda/tab è stata sviluppata interamente per l'implementazione, da parte dell'utente, del primo dei due task richiesti, che consiste nello sviluppare una funzione che restituisca tutti i baricentri di ogni forma/figura identificata tramite l'applicazione dei filtri '*lower bound*' e '*upper bound*' presenti nella schermata principale (si faccia riferimento alla Figura 7) dei due box di riferimento.

Questa funzione quindi, una volta implementata, verrà utilizzata per visualizzare sui relativi canvas presenti nella homepage tutti i vari centroidi delle varie forme rilevate dalla webcam/filtri (Figura 16).

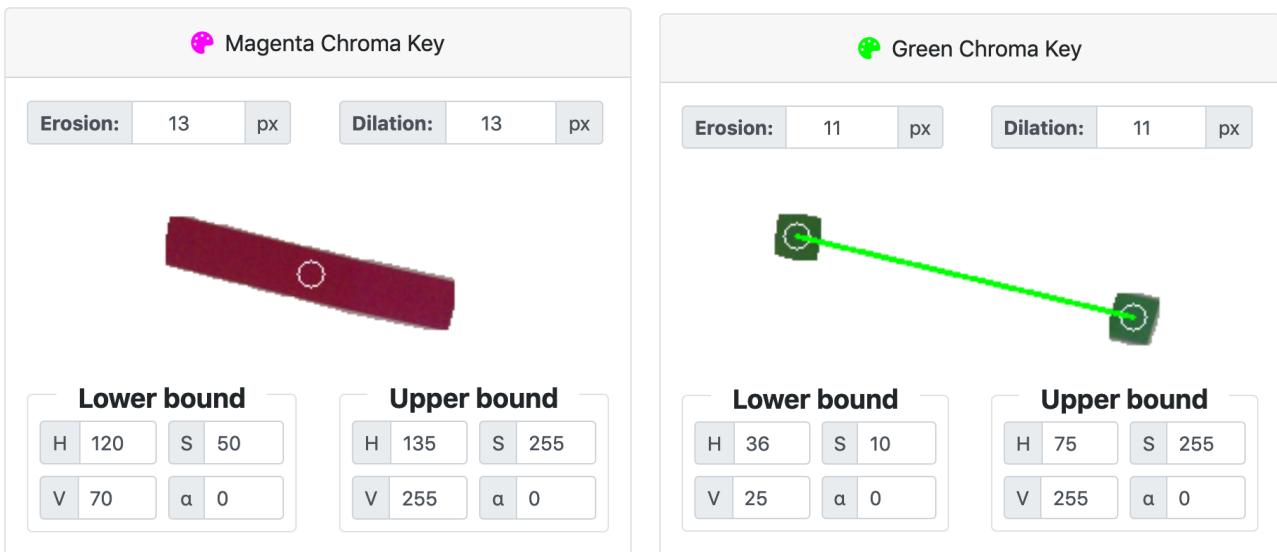


Figura 16 – Visualizzazioni, nei vari canvas presenti nella homepage, dell’output (corretto) della funzione richiesta dal primo task

L’implementazione della funzione richiesta da parte dell’utente si basa su un unico parametro, che viene passato automaticamente, e che consiste in un’array di ‘*momenti d’immagine*’ (in inglese, ‘image moments’), ovvero una serie di particolari medie d’intensità dei pixel che compongono l’immagine acquisita dalla webcam.

Un fattore importante per cui vengono utilizzati i ‘*momenti*’ di un’immagine è per il fatto che offrono una serie di proprietà che permettono di risalire, tramite l’applicazione di semplici formule matematiche, all’area, al baricentro e ad altre informazioni relative all’orientamento della figura presa in esame.

Per agevolare il completamento di questo task si sono volute aggiungere una serie di suggerimenti/aiuti visualizzabili tramite la pressione del pulsante ‘*Info*’ presente sul menù dell’editor di sviluppo (Figura 17).

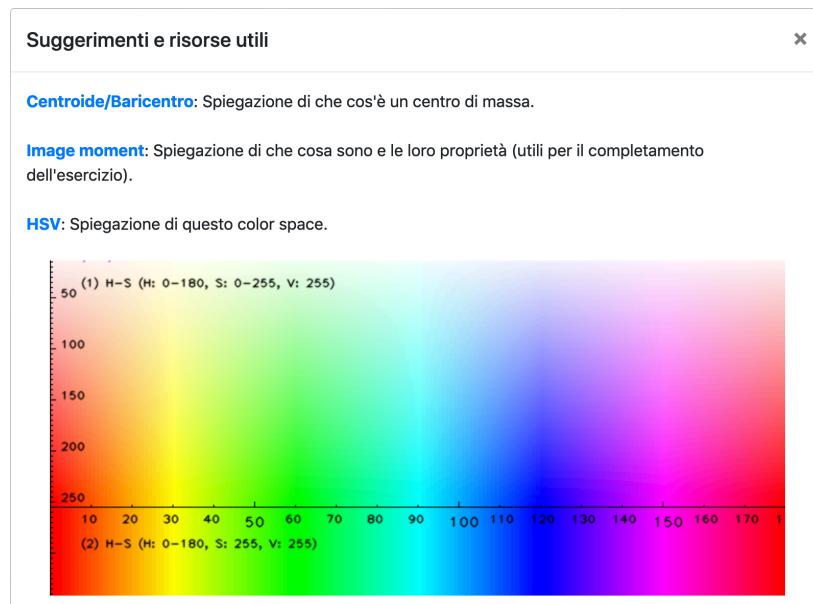


Figura 17 – Scheda di pop-up con vari suggerimenti (link esterni) al fine di completare il task richiesto

Infine, sempre sul menù dell'editor di sviluppo si è messo a disposizione la possibilità di poter visualizzare, ed incorporare automaticamente all'applicazione, una delle tante possibili implementazioni della funzione richiesta attraverso la pressione del pulsante '**Mostra soluzione**' (Figura 18).

```

1  */**
2  * La procedura 'find_centroids' deve restituire un array (di array) nel quale ogni elemento rappresenta le coordinate
3  * X e Y del baricentro di ogni forma/figura identificata tramite l'applicazione dei filtri 'Lower Bound' e
4  * 'Upper Bound' presenti nella scheda 'Webcam monitor'.
5  * Per ulteriori chiarimenti e/o informazioni consultare la scheda 'Info'
6  *
7  * parametro moments : array contenente i valori del 'Central Moments' di ogni forma/figura
8  */
9
10 function find_centroids(moments){
11     var centroids = [];
12     let i;
13     for(i=0;i<moments.length;i++){
14         let cX = moments[i].m10 / moments[i].m00;
15         let cY = moments[i].m01 / moments[i].m00;
16         centroids.push([cX,cY]);
17     }
18     return centroids;
19 }
20

```

> Messaggi del compilatore
> Building...
> Build successful

Figura 18 – Scheda/Tab 'Procedura Chroma Key' con relativa Implementazione della funzione richiesta

3.3. Procedura "Buttons Detection"

Questa terza, ed ultima, scheda/tab invece è stata sviluppata per permettere l'implementazione del secondo dei due task richiesti all'utente, che consiste nello sviluppare una funzione che restituisca l'immagine corretta sullo status e sulla relativa angolazione/inclinazione dei bottoni del joypad (si faccia riferimento alla Figura 14), identificati dai due quadrati di colore verde presenti sulla striscia di input. L'output di questa funzione verrà poi applicato al canvas "**Buttons Detection**" presente nella homepage.

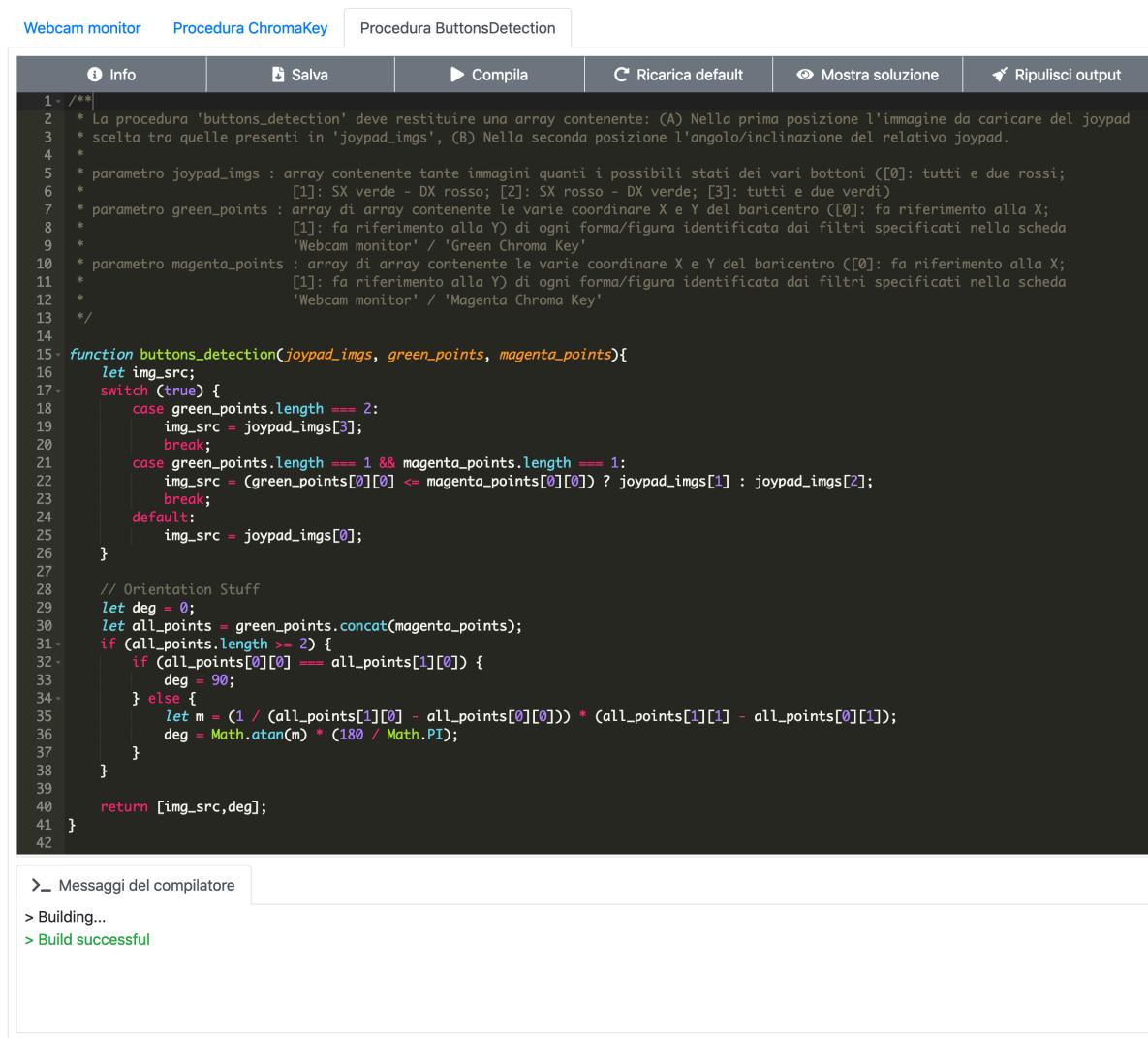
Come illustrato per la precedente scheda "Chroma Key", anche in questa situazione sono stati messi a disposizione una serie di aiuti utili al fine di completare l'esercizio richiesto, che in questo caso riguardano l'applicazione della formula di geometria analitica per il calcolo dell'equazione della retta passante per due punti (Figura 19).

Suggerimenti e risorse utili

Retta passante per 2 punti: Spiegazione di come calcolare la retta passante per due punti (e quindi ricavare l'annesso coefficiente angolare).

Figura 19 - Scheda di pop-up con vari suggerimenti (link esterni) al fine di completare il task richiesto

Infine, anche in questo caso, è possibile visualizzare ed incorporare automaticamente all'applicazione, una delle tante possibili implementazioni della funzione richiesta attraverso la pressione del pulsante '**'Mostra soluzione'**' (Figura 20).



```

Webcam monitor Procedura ChromaKey Procedura ButtonsDetection
Info Salva Compila Ricarica default Mostra soluzione Ripulisci output

1- /**
2 * La procedura 'buttons_detection' deve restituire una array contenente: (A) Nella prima posizione l'immagine da caricare del joypad
3 * scelta tra quelle presenti in 'joypad_imgs', (B) Nella seconda posizione l'angolo/inclinazione del relativo joypad.
4 *
5 * parametro joypad_imgs : array contenente tante immagini quante i possibili stati dei vari bottoni ([0]: tutti e due rossi;
6 * [1]: SX verde - DX rosso; [2]: SX rosso - DX verde; [3]: tutti e due verdi)
7 * parametro green_points : array di array contenente le varie coordinate X e Y del baricentro ([0]: fa riferimento alla X;
8 * [1]: fa riferimento alla Y) di ogni forma/figura identificata dai filtri specificati nella scheda
9 * 'Webcam monitor' / 'Green Chroma Key'
10 * parametro magenta_points : array di array contenente le varie coordinate X e Y del baricentro ([0]: fa riferimento alla X;
11 * [1]: fa riferimento alla Y) di ogni forma/figura identificata dai filtri specificati nella scheda
12 * 'Webcam monitor' / 'Magenta Chroma Key'
13 */
14
15- function buttons_detection(joypad_imgs, green_points, magenta_points){
16    let img_src;
17    switch (true) {
18        case green_points.length === 2:
19            img_src = joypad_imgs[3];
20            break;
21        case green_points.length === 1 && magenta_points.length === 1:
22            img_src = (green_points[0][0] <= magenta_points[0][0]) ? joypad_imgs[1] : joypad_imgs[2];
23            break;
24        default:
25            img_src = joypad_imgs[0];
26    }
27
28    // Orientation Stuff
29    let deg = 0;
30    let all_points = green_points.concat(magenta_points);
31    if (all_points.length >= 2) {
32        if (all_points[0][0] === all_points[1][0]) {
33            deg = 90;
34        } else {
35            let m = (1 / (all_points[1][0] - all_points[0][0])) * (all_points[1][1] - all_points[0][1]);
36            deg = Math.atan(m) * (180 / Math.PI);
37        }
38    }
39
40    return [img_src,deg];
41}
42

>__ Messaggi del compilatore
> Building...
> Build successful

```

Figura 20 - Scheda/Tab 'Procedura Buttons Detection' con relativa Implementazione della funzione richiesta

4. Conclusioni

Durante lo sviluppo di questa tesi si sono riscontrati, in svariati momenti implementativi, innumerevoli problemi quasi tutti legati alle performance dell'applicativo; per cui per offrire un'esperienza d'utilizzo confortevole si sono dovute adottate misure che hanno permesso di sopperire a questi problemi, ma con il trade-off di limitare notevolmente le possibili interazioni da parte dell'utente.

Uno dei metodi utilizzati per fronteggiare questa mancanza di prestazioni generali è stato utilizzare il più possibile le funzioni messe a disposizione dalle librerie software (descritte nella sezione “*2.2 - Librerie*”) anziché optare per un approccio di sviluppo custom, ovvero con funzioni sviluppate da zero. In questo modo le performance sono incrementate notevolmente, ma come evidenziato nel paragrafo precedente, hanno influenzato pesantemente le interazioni che l'utente può compiere con l'applicativo, poiché queste ultime sono legate dalla disponibilità (e/o limitatezza) delle stesse API delle librerie software.

Infine, si vuole sottolineare che vi sono moltissime possibili evoluzioni/update apportabili alla web-app descritta in questa tesi, tra cui:

- Aggiunta di una possibile schermata di ‘scoreboard’ con la lista di tutti gli studenti/utenti che hanno utilizzato l'applicativo ed hanno implementato una versione funzionante dei vari task richiesti, con la possibilità di visualizzare i relativi script sviluppati;
- Aggiunta di ulteriori task e/o box con nuove funzionalità sempre legate al mondo della computer vision;
- Sviluppo/Modifica in WebAssembly della web-app in modo tale da migliorare ulteriormente le prestazioni generali;
- Gamification della piattaforma in modo da aumentare l'user engagement e di conseguenza anche l'apprendimento di nuove materie/conoscenze.

5. Bibliografia/Appendice

Verrà di seguito riportato un elenco di collegamenti ipertestuali (link) riguardanti le fonti utilizzate durante lo sviluppo e la stesura di questa tesi.

- **WebRTC:** webrtc.org
- **Saturazione cromatica:** it.wikipedia.org/wiki/Saturazione_cromatica
- **Contrasto:** it.wikipedia.org/wiki/Contrasto
- **Luminosità:** [it.wikipedia.org/wiki/Luminosità_\(percezione\)](http://it.wikipedia.org/wiki/Luminosità_(percezione))
- **Acutance:** en.wikipedia.org/wiki/Acutance
- **Media Streams API:** developer.mozilla.org/en-US/docs/Web/API/Media_Streams_API - Capabilities_and_constraints
- **Ace editor:** ace.c9.io
- **Syntax highlighting:** en.wikipedia.org/wiki/Syntax_highlighting
- **Indentazione:** it.wikipedia.org/wiki/Indentazione
- **Libreria software:** [en.wikipedia.org/wiki/Library_\(computing\)](http://en.wikipedia.org/wiki/Library_(computing))
- **Bootstrap:** getbootstrap.com
- **OpenCV:** opencv.org
- **Asm.js:** asmjs.org
- **HSV:** en.wikipedia.org/wiki/HSL_and_HSV
- **Trasformata di Hough:** en.wikipedia.org/wiki/Hough_transform
- **Image moment:** en.wikipedia.org/wiki/Image_moment