



Gennaio 2020

PROGETTO COMUNICAZIONI MULTIMEDIALI

Software per la ricerca delle immagini affini

GRUPPO: G.M.A.
Giada Figliolini, Marco Cestari, Alissya Valer

SOMMARIO

1 INTRODUZIONE	2
1.1 Il gruppo	2
1.2 Obiettivi del progetto	2
2 IL PROGRAMMA.....	3
2.1 Descrizione del DataSet.....	3
2.2 Funzionamento del Confronto	3
2.2.1 Tipologie di confronto.....	3
2.3 cPICKLE.....	4
2.4 DataBase di Affinità.....	4
2.5 Linguaggi di programmazione.....	4
2.5.1 Interfaccia e funzioni ausiliarie.....	4
2.5.1 Algoritmi.....	5
2.6 Ponte tra i linguaggi	5
3 ALGORITMI.....	6
3.1 Premessa	6
3.2 Histogram	6
3.2.1 Esempio di applicazione ottimale - Histogram.....	7
3.2.2 Esempio di applicazione NON ottimale - Histogram.....	7
3.3 SSIM	8
3.3.1 Esempio di applicazione ottimale - SSIM.....	8
3.3.2 Esempio di applicazione NON ottimale - SSIM.....	9
3.4 SIFT	9
3.4.1 Esempio di applicazione ottimale - SIFT.....	10
3.4.2 Esempio di applicazione NON ottimale - SIFT	10
3.5 Tutti gli Algoritmi	11
3.5.1 Esempio di applicazione ottimale - ALL.....	11
3.5.1 Esempio di applicazione NON ottimale - ALL.....	11
3.6 Face Detection	12
4 INTERFACCIA	13
4.1 Premessa	13
4.2 Pagina Home	13
4.3 Pagina Impostazioni.....	15
4.2 Pagina Guida	16
5 CONCLUSIONE.....	16

1 INTRODUZIONE

1.1 IL GRUPPO

Il gruppo G.M.A. è composto da tre componenti:

- i. Giada Figliolini (matricola: 202121, mail: giada.figliolini@studenti.unitn.it)
- ii. Marco Cestari (matricola: 205306, mail: marco.cestari-1@studenti.unitn.it)
- iii. Alissya Valer (matricola: 200593, mail: alissya.valer@studenti.unitn.it)

Siamo tutti e tre studenti frequentanti il terzo anno di Ingegneria dell'Informazione ed Organizzazione d'Impresa.

La scelta di lavorare solamente in tre è data dal fatto che preferiamo essere pochi, al fine mantenere una visione chiara e condivisa dell'idea del progetto. Inoltre, avendo già lavorato assieme, sappiamo che ci troviamo bene nell'interagire tra noi.

1.2 OBIETTIVO DEL PROGETTO

L'obiettivo del progetto, come da consegna, è quello di realizzare un software che confronti un'immagine in input con una serie di altre immagini, al fine di selezionare le più affini a quella data.

Anzitutto, abbiamo deciso di realizzare un'interfaccia grafica a sostegno dell'utente, tramite la quale può interagire più comodamente e facilmente con il programma.

Per quanto riguarda gli algoritmi utilizzati, la nostra scelta è caduta su:

- Histogram
- SSIM
- SIFT
- Abbiamo ritenuto utile aggiungere Face Detection.

Ciò che ci ha guidato per tutto lo sviluppo dell'applicazione, è stato cercare di offrire all'utente la maggiore adattabilità e facilità di utilizzo del software. Abbiamo realizzato ciò ponendo particolare attenzione alla personalizzazione del programma e a tutte le funzioni che permettevano all'utente di compiere una scelta, creando degli algoritmi di configurazione per gestirle.

Tramite il codice, da noi realizzato, l'utente sarà quindi in grado di scegliere un'immagine in input e selezionare l'archivio d'immagini sul quale applicare il confronto, quali algoritmi impiegare nell'operazione, la lingua e altre impostazioni. In output appariranno un numero di immagini (deciso dall'utente), con le relative percentuali di somiglianza.

2 PROGRAMMA

2.1 DESCRIZIONE DATASET

Il DataSet che abbiamo creato, di supporto nello sviluppo del nostro programma, contiene immagini di nostra proprietà ed altre prese del web senza copyright. Grazie a questo archivio di immagini è stato possibile effettuare diverse prove dell'applicazione per testare il funzionamento delle varie componenti oltre che del software finale.

Le fotografie sono quasi tutte di alta qualità in quanto sono state scattate da smartphone o macchine fotografiche e sono salvate in formato JPG.

Il DataSet contiene all'incirca 220 immagini che raffigurano situazioni e oggetti molto diversi tra di loro, sono di dimensione variabile e con orientamento sia orizzontale che verticale.

Abbiamo voluto creare un programma funzionante con immagini molto varie tra di loro, perché abbiamo pensato che un utente esterno, facendo foto da vari dispositivi, non abbia immagini con grandezze e dimensioni identiche.

2.2 FUNZIONAMENTO DEL CONFRONTO

Il confronto avviene tra due immagini alla volta. Viene effettuato tramite una serie di algoritmi, il cui compito è ricercare somiglianze e differenze tra esse, fornendo successivamente una valutazione della comparazione, ossia una percentuale (che va da 0 a 100) che quantifica l'affinità tra le due immagini.

2.2.1 TIPOLOGIE DI CONFRONTO

Il nostro programma offre l'opportunità di scelta riguardo a come effettuare la comparazione, le varie possibilità sono basate sulle caratteristiche dell'immagine:

- **Ricerca per differenze:** è basato sull'algoritmo *SSIM*. Se selezionato permette di attuare un confronto tra immagini sulla base delle loro differenze; è utile per avere un'accurata valutazione della somiglianza tra le fotografie.
- **Ricerca per colore:** è basato sull'algoritmo *Histogram*. In base ai colori presenti nell'immagine in input effettua un paragone con immagini in archivio per ottenere in output quelle con la palette di colori più simile.
- **Ricerca per punti chiave:** è basato sull'algoritmo *SIFT*. Focalizza alcuni punti caratterizzanti dell'immagine in input e cerca di identificare se le altre immagini in archivio presentano le stesse somiglianze.
- **Tutti gli algoritmi:** utilizza tutti e tre gli algoritmi, ponderando i loro valori per cercare di ottenere il miglior output possibile.
- **Rilevazione dei Volti:** è un'ulteriore impostazione, basata sull'algoritmo di *Face Detection*. Permette all'utente di configurare se l'output presenterà dei volti o meno.

2.3 cPICKLE

Inizialmente il nostro programma calcolava tutte le affinità in run-time, impiegando un tempo di elaborazione troppo elevato (2 minuti circa).

Per ovviare a questa problematica e far sì che il programma rispondesse in modo più veloce ed efficiente, abbiamo pensato di introdurre i *file cPICKLE*, notando che grazie a questi, i tempi di ottenimento dei risultati diminuivano sensibilmente (circa 20 secondi). Questa implementazione permette di salvare alcuni valori intermedi, necessari al fine di facilitare l'applicazione degli algoritmi di confronto.

Questi file sono utili, in quanto permettono l'utilizzo dei valori senza doverli ricalcolare ad ogni confronto.

2.4 DATABASE DI AFFINITÀ

Inoltre, abbiamo implementato un Database, con l'obiettivo di salvare i risultati delle affinità tra l'immagine in input e le varie immagini presenti nell'archivio, in modo da permettere il passaggio dei risultati da Python a C#.

È un DataBase temporaneo, in quanto a ogni comparazione viene pulito e modificato con i dati ricavati dal nuovo confronto andando, però, a fare un UPDATE anziché creare un nuovo database da zero, per evitare un allungamento dei tempi e una minore efficienza.

Questo DataBase è composto da: nome dell'immagine con cui si è confrontata l'immagine in input, la percentuale di affinità tra le immagini, e il valore del Face Detection (numero di volti rilevati).

2.5 LINGUAGGI DI PROGRAMMAZIONE UTILIZZATI

2.5.1 INTERFACCIA e FUNZIONI AUSILIARIE

Per la realizzazione dell'interfaccia grafica è stato utilizzato il linguaggio di programmazione C# con Visual Studio Community.

Lo abbiamo scelto in quanto permette una creazione facilitata dell'interfaccia fornendo fin da subito, in maniera visuale, gli strumenti necessari per l'implementazione di un'interfaccia più user-friendly e personalizzata.

Anche le funzioni ausiliare, ossia quelle funzioni che permettono di creare e gestire la personalizzazione del programma, consentendo all'utente di effettuare determinate scelte, sono state implementate tramite C#, perché avendo realizzato l'interfaccia con lo stesso linguaggio le due aree si interfacciano in maniera più efficiente.

2.5.2 ALGORITMI

In merito agli algoritmi, invece, abbiamo preferito utilizzare Python con Visual Studio Code, in quanto fornisce molte librerie, utili per il confronto delle immagini.

2.6 PONTE TRA I LINGUAGGI

La scelta di utilizzare due linguaggi di programmazione ci ha portato alla necessità di progettare e creare un “ponte” tra i due, che ci permettesse di eseguire gli algoritmi in Python ma mostrare successivamente i risultati, e quindi le immagini più affini, sull’interfaccia grafica realizzata in C#.

La funzione di “ponte” nel nostro programma si chiama “*getResultsComparision*” e si trova nella classe “*interconnectorPy*”; in seguito, è possibile vedere la parte di codice citata con i vari commenti di spiegazione:

```
result = interconnectorPy.getResultsComparision();

if (result == "OK\r\n")
{
    // Esegui Comparazione con i dati salvati nel DB da Python
    // Se arriva qui sono SICURO che python abbia già finito di calcolare e salvare i risultati delle comparazioni
}

public static string getResultsComparision()
{
    // 1) Crea un nuovo processo (Command Promp)
    var psi = new ProcessStartInfo();
    psi.FileName = "python"; // Nome del programma da usare per lanciare gli argomenti

    // 2) Provide script and arguments

    var script = Application.StartupPath + @"\algorithms\" + Properties.Settings.Default.pref_algorithm + ".py"; // Nome del file Python da eseguire
    var arg1 = Properties.Settings.Default.default_input_image; // Argomento 1
    var arg2 = Properties.Settings.Default.default_dir; // Argomento 2
    var arg3 = Application.StartupPath; // Argomento 3
    psi.Arguments = $"{script}\" \"{arg1}\" \"{arg2}\" \"{arg3}\""; // Fusione dello script con gli argomenti

    // Esempio Psi.Argument -> "python filePython.py "C://Argomento1" "C://Argomento2" "C://Argomento3"

    // 3) Process configuration
    psi.UseShellExecute = false;
    psi.CreateNoWindow = true; // Permette di NON vedere una shell aprirsi ogni volta che si lancia
    psi.RedirectStandardOutput = true; // Reindirizza l'OUTPUT per poter essere letto
    psi.RedirectStandardError = true; // Reindirizza gli ERRORI per poter essere letti

    // 4) Execute process and get output
    var errors = "";
    var results = "";

    using (var process = Process.Start(psi)) // Lancio effettivo del comando
    {
        errors = process.StandardError.ReadToEnd(); // Lettura dell'Output
        results = process.StandardOutput.ReadToEnd(); // Lettura di eventuali Errori per DEBUG
    }

    return results;
}
```

3 ALGORITMI

3.1 PREMESSA

Abbiamo scelto tre tipi di algoritmi per fare il confronto tra le immagini: Histogram, SSIM, SIFT. Inoltre, abbiamo implementato un'ulteriore funzionalità: il Face Detection, per fornire all'utente una maggiore accuratezza nella scelta delle caratteristiche delle immagini in output.

La scelta di questi algoritmi è dovuta al fatto che volevamo fare una ricerca delle immagini più completa possibile, considerando e valutando tutti i principali componenti che un'immagine può contenere, facendo comunque una costante attenzione a non appesantire troppo il programma.

3.2 HISTOGRAM

L'algoritmo Histogram si basa sulla comparazione degli istogrammi dei colori presenti nelle varie immagini.

Innanzitutto, le immagini vengono ridimensionate in un quadrato di 1000x1000 pixel così da non falsare il risultato finale, successivamente vengono trasformate in una scala di colori di tipo HSV per garantire un'alta saturazione dei colori.

La motivazione riguardo alla decisione di adottare questa tipologia di scala di colori è data dal fatto, che in seguito a numerose prove e tentativi, abbiamo riscontrato una maggiore affidabilità dei risultati con HSV rispetto a BGR o GRAY.

Una volta trasformata l'immagine viene utilizzata la funzione *calcHist* della libreria cv2 (OpenCV) che restituisce il valore dell'istogramma.

L'istogramma viene quindi normalizzato tramite la funzione *normalize* di cv2, per ottenere risultati il più vicini possibile alla realtà.

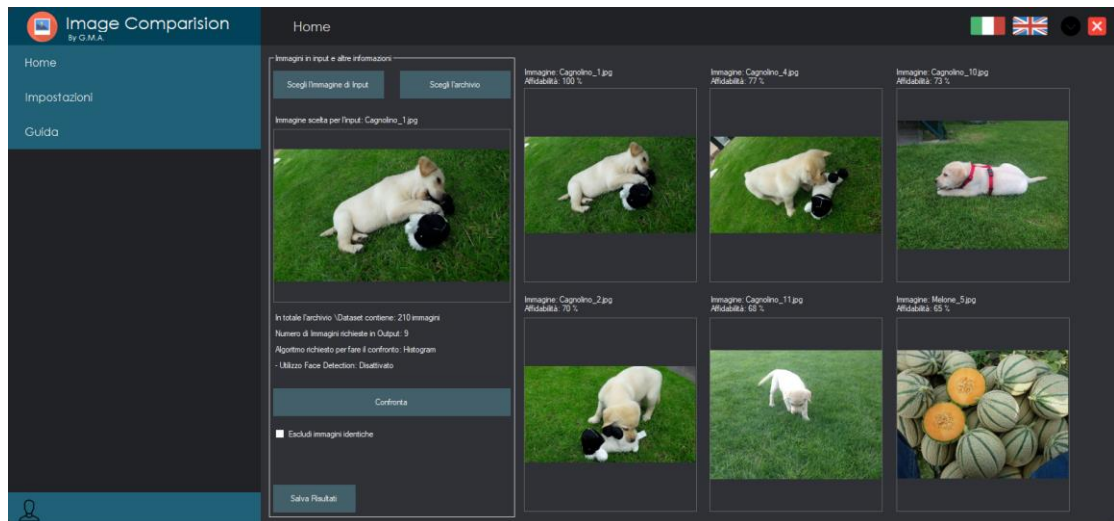
Infine, il risultato viene salvato nei file cPICKLE per poter essere letto successivamente, durante le comparazioni.

La comparazione viene effettuata attraverso la funzione *compareHist* che, dati in ingresso due istogrammi e il metodo di confronto (in questo caso *CORRELATION*), permette di ottenere un valore compreso tra 0 e 1 che, se moltiplicato per 100, fornisce una percentuale di somiglianza dei due istogrammi.

Abbiamo scelto questo algoritmo perché il colore è una delle caratteristiche principale che può avere un'immagine, l'Histogram ci ha permesso di analizzarlo (in tutte le sue gradazioni e sfumature) e confrontarlo.

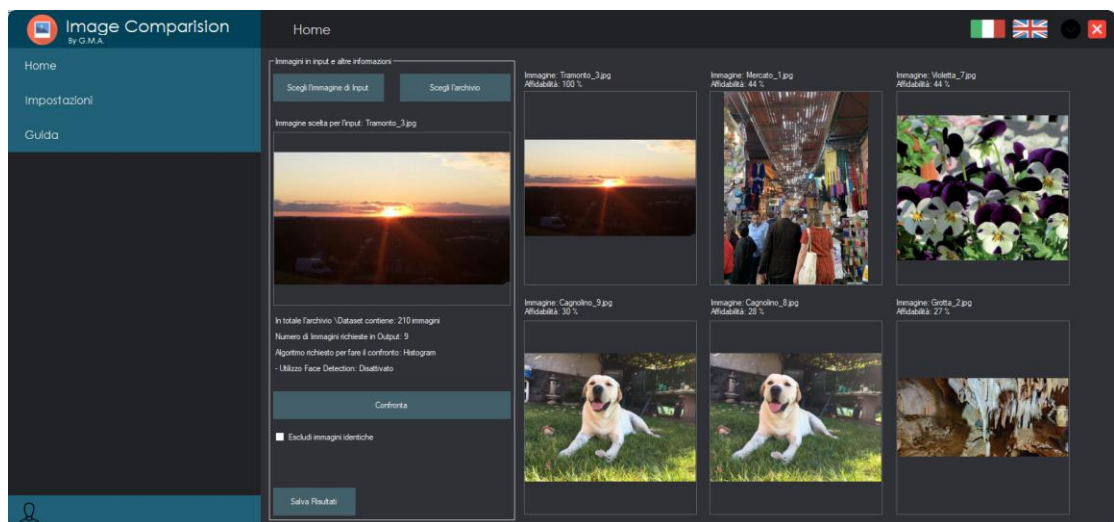
Inoltre, ha un'implementazione abbastanza semplice e computazionalmente non onerosa che non appesantisce troppo il programma.

3.2.1 ESEMPIO DI APPLICAZIONE OTTIMALE - Histogram



Utilizzando questa foto abbiamo notato che il risultato è molto buono in quanto le immagini si assomigliano tra di loro, infatti, mostrano lo stesso soggetto in varie posizioni. Secondo noi, questo algoritmo ha funzionato ottimamente perché la fotografia di input presenta pochi colori uniformi tra di loro e poche sfumature.

3.2.2 ESEMPIO DI APPLICAZIONE NON OTTIMALE - Histogram



Utilizzando questa fotografia abbiamo notato che il risultato non soddisfa le aspettative, in quanto le immagini in output si discostano molto dall'immagine di input. La spiegazione che abbiamo trovato per questo è che nell'immagine di input sono presenti colori con molte sfumature e il programma fatica nella comparazione dei loro istogrammi.

3.3 SSIM

L'algoritmo SSIM (Structural Similarity Index Measure) si basa sulla somiglianza effettiva delle immagini e quindi va a rilevare le differenze tra due fotografie, andando a sovrapporle e calcolando la differenza pixel a pixel per marcare le aree in cui l'immagine si discosta maggiormente.

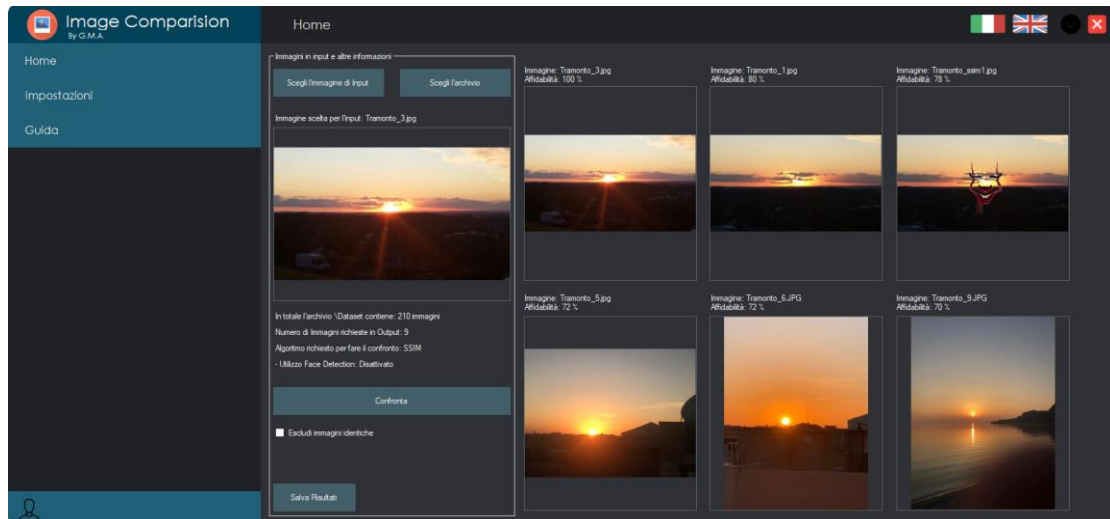
Anche in questo caso, come in quello dell'Histogram, abbiamo ridimensionato le immagini in un quadrato di 1000x1000 pixel, sia per migliorare il risultato finale ma, in particolare, per necessità in quanto questo algoritmo richiede che le immagini siano della stessa grandezza per riuscire a sovrapporle.

In seguito al ridimensionamento dell'immagine, questa viene convertita in scala di grigi per poi essere salvata nel file cPICKLE con l'obiettivo di utilizzarla successivamente.

Per effettuare il confronto si utilizza la funzione *structural_similarity* che, dando come input due immagini permette di ottenere un valore compreso tra 0 e 1, che moltiplicato per 100, ci consente di ottenere la percentuale di somiglianza tra le immagini.

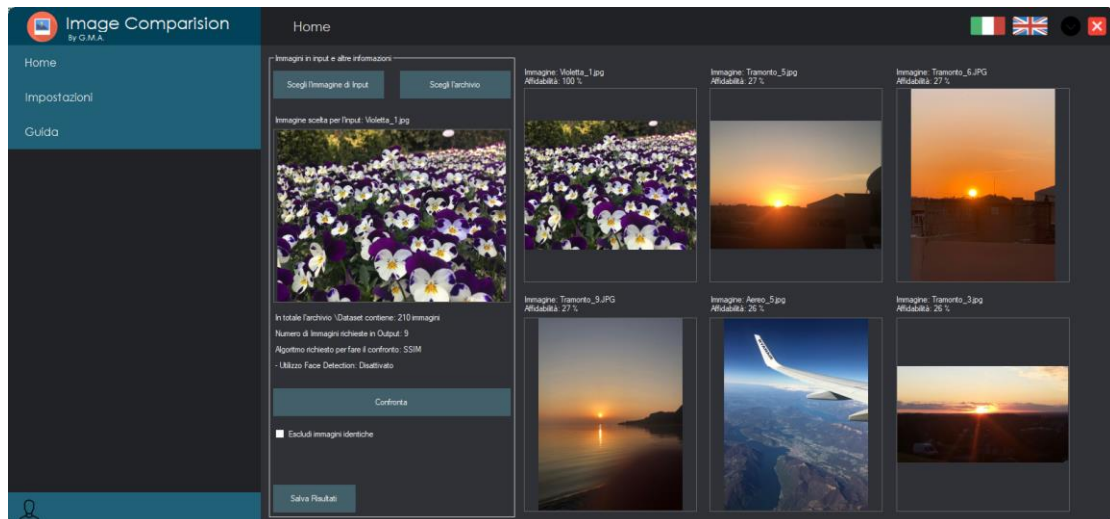
Abbiamo scelto questo algoritmo perché per noi la similarità della struttura è una parte fondamentale per la verifica dell'affinità tra le immagini.

3.3.1 ESEMPIO DI APPLICAZIONE OTTIMALE - SSIM



Utilizzando questa fotografia si può notare come il risultato sia buono, in quanto le immagini ottenute raffigurano tutte dei tramonti. Secondo noi, in questo caso, l'algoritmo ha funzionato ottimamente, infatti la struttura delle varie fotografie è molto simile. Tutte le immagini presentano nella parte inferiore una porzione molto scura, con elementi poco distinguibili tra di loro e una parte superiore con colori più chiari, contenenti in posizione centrale un elemento comune a tutte le fotografie (il sole).

3.3.2 ESEMPIO DI APPLICAZIONE **NON** OTTIMALE - SSIM



Utilizzando l'algoritmo su quest'immagine il risultato non ci soddisfa in quanto, le fotografie ottenute in output si discostano molto da quella di input. Una possibile spiegazione potrebbe essere che nell'immagine di input non sia presente una struttura chiara e definita.

3.4 SIFT

L'algoritmo SIFT (Scale Invariant Feature Transform) si basa sulla ricerca di pattern di pixel comuni tra due immagini.

È particolarmente adatto nella ricerca di strutture simili tra di loro, come possono essere monumenti o edifici.

Abbiamo innanzitutto convertito l'immagine in scala di grigi, e solo successivamente abbiamo utilizzato la funzione *detectAndCompute*, della libreria SIFT di cv2, per ottenere i *KeyPoint* e il *Desc* (cioè il descrittore dell'immagine).

In seguito a queste operazioni i due valori (*KeyPoint* e *Desc*) vengono salvati nei file cPICKLE per essere utilizzati successivamente.

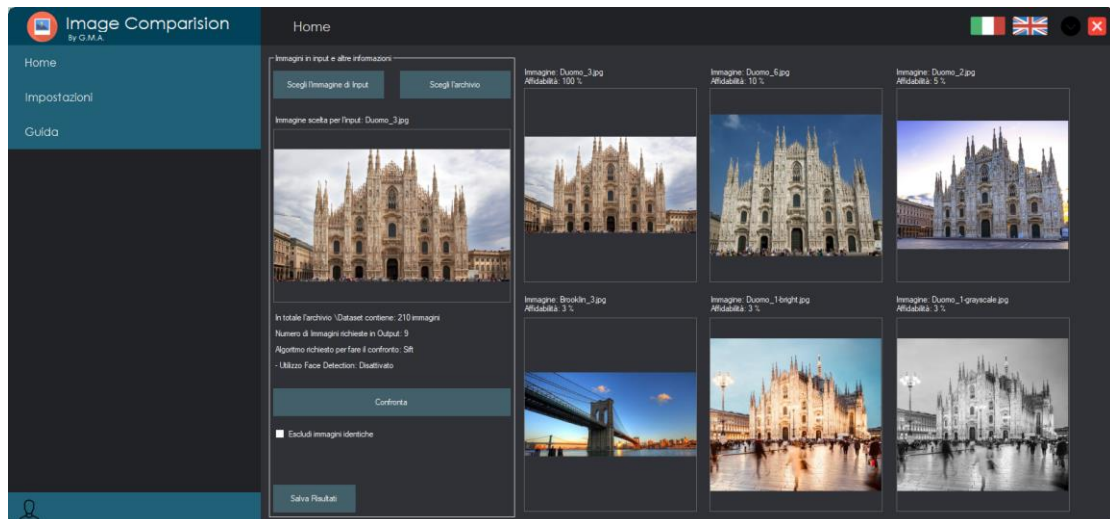
Per poter calcolare l'affinità tra le immagini si utilizza una funzione di cv2 chiamata *BFMatcher* che, attraverso il *BruteForce*, riesce a capire se vi sono dei pattern comuni tra le immagini.

Si va quindi a contare il numero di *KeyPoint* in comune tra le immagini (chiamati *GoodPoints*) e lo si va a dividere per il numero totale dei *KeyPoint*.

Moltiplicando per 100 questo valore si ottiene la percentuale di affinità.

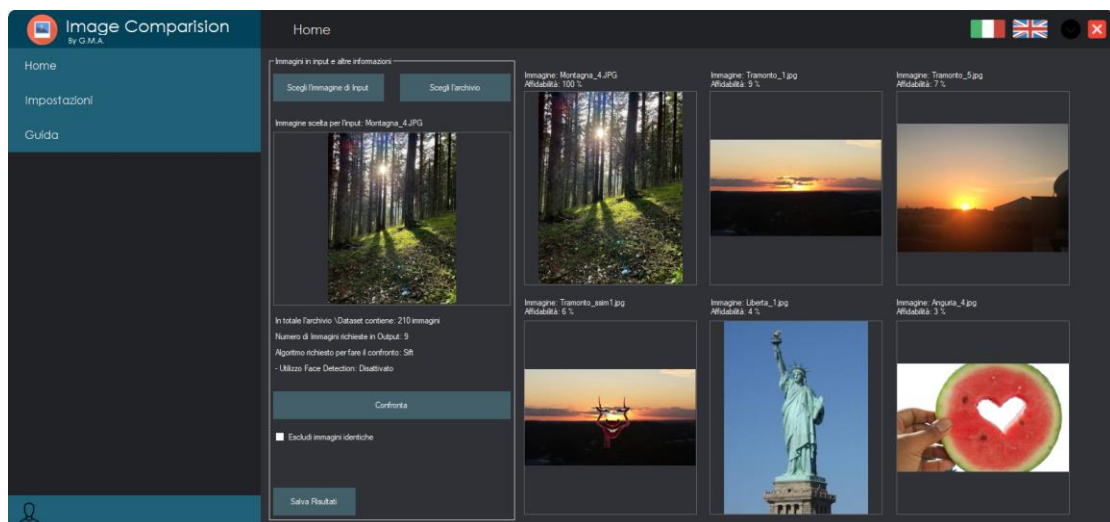
Abbiamo deciso di implementare questo algoritmo perché ci sembrava utile avere un terzo algoritmo che andasse a verificare se le immagini avessero dei pattern in comune tra di loro e quindi, ad esempio, degli oggetti con forme simili.

3.4.1 ESEMPIO DI APPLICAZIONE OTTIMALE - SIFT



Utilizzando questa fotografia si può notare che il risultato è molto buono, in quanto sono presenti diverse figure geometriche che si ripetono per ogni immagine (esempio: forma del tetto, le finestre ecc).

3.4.2 ESEMPIO DI APPLICAZIONE NON OTTIMALE - SIFT



Utilizzando l'algoritmo su quest'immagine il risultato non ci soddisfa in quanto, le foto che risultano in output si discostano troppo dall'input. Questo accade perché l'immagine di input non presenta delle forme chiare e definite ma un generico paesaggio.

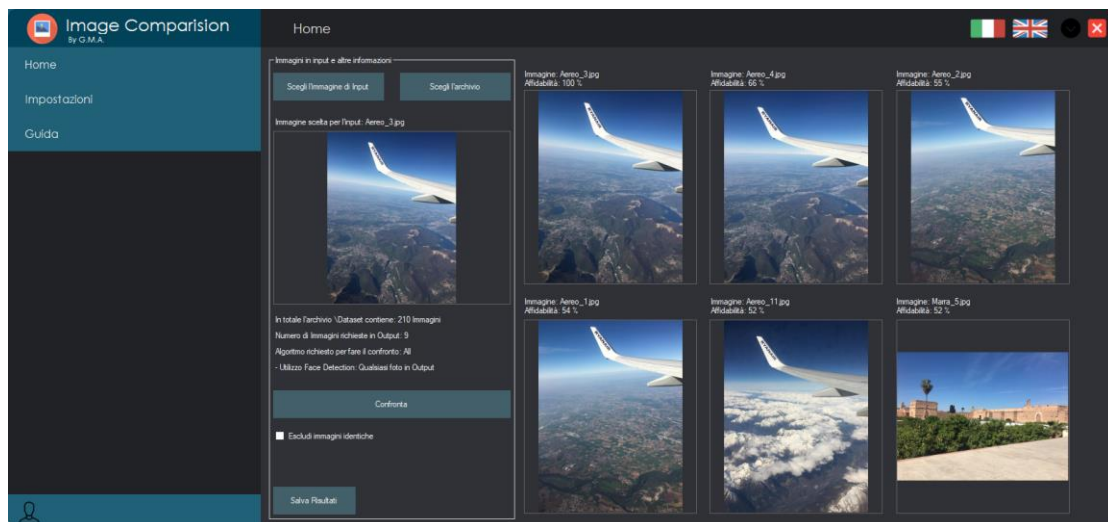
3.5 TUTTI GLI ALGORITMI

Questa opzione consente di selezionare tutti e tre gli algoritmi citati in precedenza.

I risultati dei tre algoritmi vengono ponderati sulla base della loro rilevanza, pertanto l'Histogram viene considerato per il 40% del suo valore, l'SSIM per il 35% e infine il SIFT per un 25%.

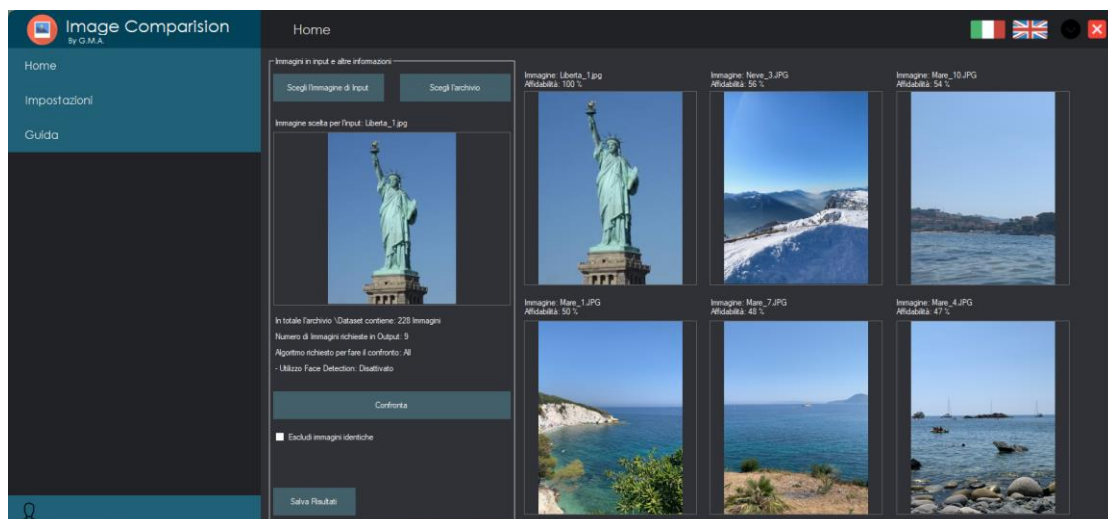
La scelta di questa suddivisione è data dal fatto che i colori fornisco un buon criterio nella verifica di somiglianza, mentre il risultato del SIFT è quello che procura un margine d'errore più elevato.

3.5.1 ESEMPIO DI APPLICAZIONE OTTIMALE - ALL



Dalle nostre prove è emerso che tramite l'utilizzo congiunto di tutti gli algoritmi si ottengano solitamente degli ottimi risultati. Questo risvolto è probabilmente dovuto al fatto che l'algoritmo sfrutta tutti i vantaggi dei singoli tre algoritmi e li combini assieme. In questo esempio si può notare come una fotografia, presentante più elementi non semplici da confrontare (come l'ala dell'aereo oppure il paesaggio), riesca ad essere analizzata in maniera tale da ottenere risultati così simili.

3.5.2 ESEMPIO DI APPLICAZIONE **NON** OTTIMALE - ALL



In questo esempio si può notare come gli output ottenuti non presentino caratteristiche abbastanza simili tra loro. Tale discostamento può essere frutto del peso più sostanzioso della comparazione di colori tramite l'Histogram, rispetto agli altri algoritmi, il quale, trovando un'elevata quantità di azzurro, tende a confondere le varie immagini con questa gradazione.

Purtroppo, questa funzione, sebbene sia quella che probabilmente ottiene risultati più attendibili, a volte presenta come esito valori discostanti tra loro. Infatti, le percentuali di utilizzo degli algoritmi sono costanti e utili nella maggior parte dei casi, ma in alcune fotografie servirebbero altre proporzioni.

3.6 FACE DETECTION

Il Face Detection è un algoritmo secondario in quanto non svolge la principale funzione del progetto, ovvero trovare una percentuale di affinità, ma fornisce solamente un check se nelle immagini sono presenti dei volti.

Abbiamo pensato di lasciare la libertà all'utente di utilizzarlo in tre modalità:

- *Default* → l'algoritmo in questo caso permette di ottenere in output immagini sia con volti sia senza (quindi qualsiasi immagine)
- *Face Detection* → l'algoritmo obbliga la funzione confronto a dare come output solo immagini CON dei volti
- *No Face Detection* → l'algoritmo obbliga la funzione confronto a dare come output solo immagini SENZA volti

Questo algoritmo, quindi, è utilizzato nel nostro programma per migliorare la precisione della ricerca, e fornire all'utente una maggior personalizzazione e possibilità di scelta.

4 INTERFACCIA

4.1 PREMESSA

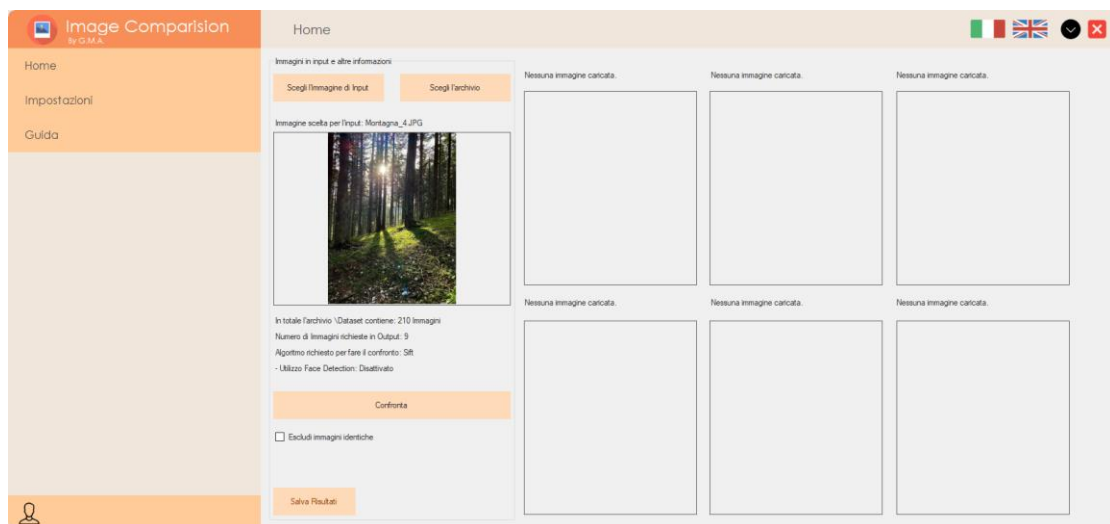
Abbiamo deciso di realizzare un'interfaccia grafica a sostegno del software che faciliti l'usabilità del programma da parte dell'utente.

La comodità del linguaggio utilizzato ci ha permesso di renderla semplice, chiara e colorata. Infatti, la nostra volontà non è solo quella di aiutare l'utente nella sua esperienza con l'applicazione, ma anche di renderla più piacevole e vivace.

L'interfaccia si divide in tre pagine: HOME, IMPOSTAZIONI E GUIDA:

- *Home*: è il cuore del programma dove sono presenti le funzioni e le informazioni per fare il confronto con l'immagine scelta in input.
- *Impostazioni*: progettata per gestire le possibili scelte che l'utente ha la possibilità di effettuare.
- *Guida*: è una pagina statica, che mostra una breve spiegazione dei vari algoritmi che possono essere selezionati e un consiglio in merito alla tipologia di immagini con cui funzionano meglio.

4.2 PAGINA HOME



Bottoni:

- *Scegliere immagine in input*: permette all'utente di scegliere l'immagine che vuole confrontare, tramite il *fileDialog* e quindi "l'esplora risorse" del proprio computer.
- *Scegli archivio*: consente la scelta da parte dell'utente della cartella contenente le immagini da utilizzare per il confronto. La selezione dell'archivio viene effettuata tramite il *dirFileDialog*.
- *Confronta*: attiva l'effettivo confronto tra l'immagine scelta in input e le varie immagini dell'archivio selezionato in precedenza e avvia l'esecuzione degli

algoritmi con l'obiettivo di trovare le immagini con maggiore affinità, le quali vengono le mostrate sull'interfaccia.

- *Salva risultati*: permette di salvare in una cartella, scelta dall'utente, i risultati ottenuti in output, ossia le immagini più simili a quella in input.

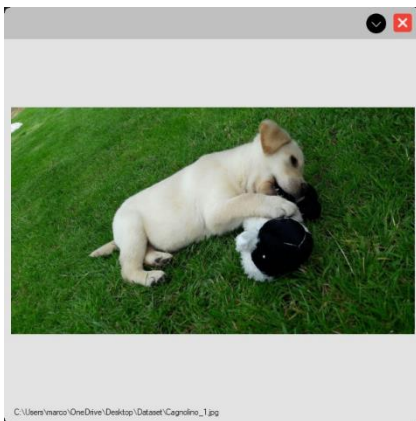
La checkbox *"Escludi immagini identiche"* offre la possibilità di includere nel confronto anche l'immagine stessa (scelta in input) oppure escluderla. Questo permette di verificare se l'immagine data è presente nella cartella selezionata per il confronto o no. Inoltre, è utile per constatare se il nostro programma funziona come dovrebbe; in tal caso l'immagine identica a quella in input dovrebbe essere trovata con una corrispondenza del 100%.

L'immagine scelta in input verrà raffigurata nel riquadro *"Immagine scelta per l'input"*, mostrando sopra l'immagine il relativo nome associato ad essa.

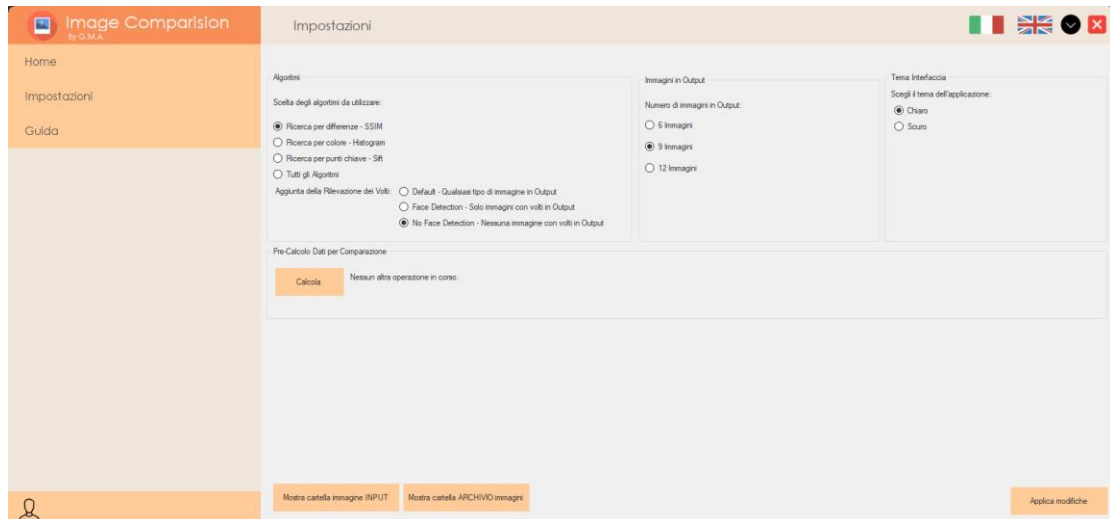
Inoltre, si possono vedere una serie di informazioni descrittive, utili per l'utente, che mostrano il *Numero di immagini presenti nell'archivio selezionato*, il *Numero di immagini richieste in output*, l'*Algoritmo scelto* e lo *stato del Face Detection*.

Sulla parte destra della pagina sono raffigurati i riquadri dove verranno mostrate le *Immagini più affini*, in seguito al confronto, con relativi nomi e percentuali di affinità.

Effettuando un doppio click su un'immagine appare una seconda *form* che raffigura la singola immagine desiderata con dimensioni maggiori e mostra la relativa path nella quale è salvata.



4.3 PAGINA IMPOSTAZIONI



La pagina *Impostazioni* è suddivisa in 3 aree che permettono all'utente di effettuare una serie di scelte:

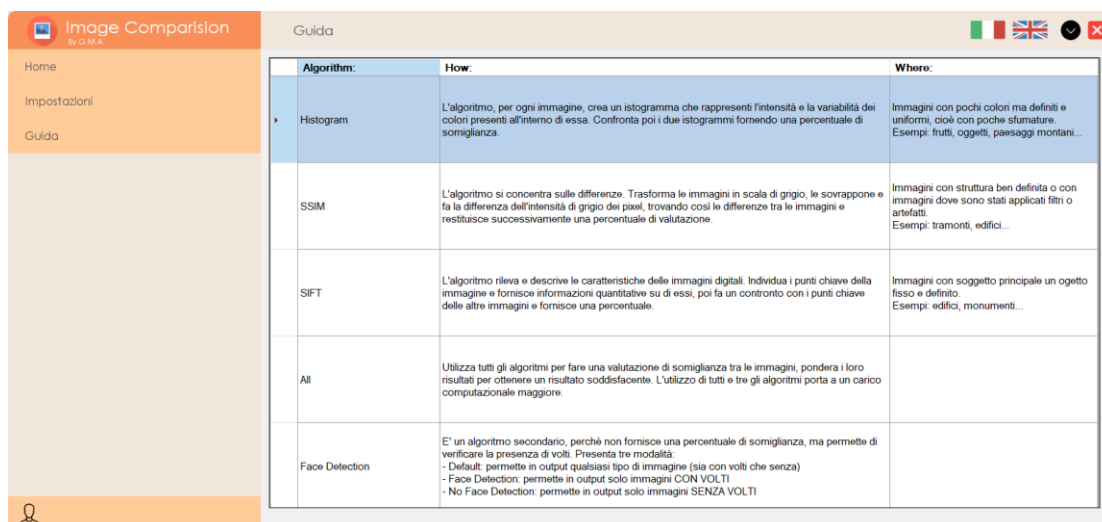
- **Algoritmi:** permette all'utente la scelta delle caratteristiche, e quindi dei vari algoritmi, su cui basare il confronto; ha anche la possibilità di selezionarli tutti e tre tramite il *RadioButton* "Tutti gli Algoritmi". Inoltre, è possibile scegliere se utilizzare l'algoritmo secondario "Face Detection" nel confronto oppure no. Sono presenti 3 opzioni in merito:
 - *Default – Qualsiasi tipo di immagine in Output*
 - *Face Detection – Solo immagini con volti in Output*
 - *No Face Detection – Nessuna immagini con volti in Output*
- **Tema Interfaccia:** consente di cambiare il tema (dello sfondo e delle scritte) dell'interfaccia rendendo i colori chiari o scuri in base alla preferenza.
- **Immagini in Output:** Consente di decidere il numero di immagini che verranno mostrate in output. Le possibilità sono: 6, 9 o 12 immagini. Ci sembrava un buon compromesso tra un numero sufficiente di risultati ma non troppo eccessivo da permettere un discostamento dall'input.

Ulteriormente, abbiamo implementato la possibilità di scegliere la *Lingua* del programma tra inglese e italiano. Questa opzione è modificabile in ogni pagina dell'interfaccia.

Bottoni:

- **Calcola:** è utile nel momento del calcolo dei dati per le successive comparazioni, permette infatti di creare i file cPICKLE.
- **Mostra cartella immagine di INPUT:** consente di mostrare la cartella dove si trova l'immagine che è stata selezionata per l'input.
- **Mostra cartella ARCHIVIO immagini:** mostra la cartella, selezionata come archivio, nella quale sono presenti le immagini da confrontare.
- **Applica modifiche:** permette di applicare tutte le scelte una volta effettuate dall'utente e salvarle per l'utilizzo dell'applicazione.

4.4 PAGINA GUIDA



Algorithm:	How:	Where:
Histogram	L'algoritmo, per ogni immagine, crea un istogramma che rappresenti l'intensità e la variabilità dei colori presenti all'interno di essa. Confronta poi i due istogrammi fornendo una percentuale di somiglianza.	Immagini con pochi colori ma definiti e uniformi, cioè con poche sfumature. Esempi: frutti, oggetti, paesaggi montani...
SSIM	L'algoritmo si concentra sulle differenze. Trasforma le immagini in scala di grigio, le sovrappone e fa la differenza dell'intensità di grigio dei pixel, trovando così le differenze tra le immagini e restituisce successivamente una percentuale di valutazione.	Immagini con struttura ben definita o con immagini dove sono stati applicati filtri o artefatti. Esempi: tramonti, edifici...
SIFT	L'algoritmo rileva e descrive le caratteristiche delle immagini digitali. Individua i punti chiave della immagine e fornisce informazioni quantitative su di essi, poi fa un confronto con i punti chiave delle altre immagini e fornisce una percentuale.	Immagini con soggetto principale un oggetto fisso e definito. Esempi: edifici, monumenti...
All	Utilizza tutti gli algoritmi per fare una valutazione di somiglianza tra le immagini, pondera i loro risultati per ottenere un risultato soddisfacente. L'utilizzo di tutti e tre gli algoritmi porta a un carico computazionale maggiore.	
Face Detection	E' un algoritmo secondario, perché non fornisce una percentuale di somiglianza, ma permette di verificare la presenza di volti. Presenta tre modalità: <ul style="list-style-type: none">- Default: permette in output qualsiasi tipo di immagine (sia con volti che senza)- Face Detection: permette in output solo immagini CON VOLTI- No Face Detection: permette in output solo immagini SENZA VOLTI	

A sostegno dell'utente è stata creata una specifica pagina dell'interfaccia, con lo scopo di illustrare in sintesi il *funzionamento* degli *algoritmi implementati*.

Inoltre, si è aggiunta la colonna "*Where*", affinché la scelta dell'utente possa essere il più ponderata, mirata ed efficiente possibile.

La decisione di aggiungere questa pagina è dovuta al fatto che non tutti i possibili utilizzatori dell'applicazione abbiano un'idea chiara di come funzionino i vari algoritmi.

5 CONCLUSIONI

Nel complesso siamo davvero soddisfatti del progetto da noi realizzato. Ci ha permesso di imparare cose non ancora conosciute e metterci in gioco, realizzando un software che probabilmente non ci ritenevamo in grado di creare. Infatti, non abbiamo mai avuto l'occasione di realizzare un progetto complesso e completo come questo (comprendente non solo codice ma anche l'interfaccia, le presentazioni, i grafici e il report finale), ne sapevamo da dove partire inizialmente.

Abbiamo riscontrato alcune problematiche durante lo sviluppo di questo progetto.

Anzitutto, il tempo di elaborazione degli algoritmi, dapprima run time, era troppo elevato (anche più di 2 minuti per confronto).

Per risolvere questo problema abbiamo utilizzato dei file chiamati cPICKLE che permettono di salvare i valori delle varie immagini in delle cartelle, così da essere disponibili anche per confronti successivi, e un database. Questo ha portato ad un tempo di pre-calcolo dei vari cPICKLE iniziale di circa 1 minuto e mezzo, però il tempo di comparazione dei confronti è diminuito da 2 minuti a pochi secondi (max 15 secondi).

Questo ha portato anche ad un ritardo rispetto alle scadenze previste dal grafico Gantt, ma visti i risultati ci possiamo ritenere soddisfatti dell'implementazione effettuata.

Per quanto riguarda gli algoritmi, quello più difficile da implementare è stato il SIFT, perché per funzionare richiede di salvare contemporaneamente non uno, ma due valori (KeyPoint e Desc). Questo algoritmo è quello che ha creato più problemi anche nei confronti dell'efficacia; infatti, si è notato che spesso i risultati in output erano troppo distanti dall'input, specialmente nelle immagini con pochi e confusi punti chiave su cui basare il confronto.

Al contrario, l'Histogram e l'SSIM sono gli algoritmi che sono risultati più fruttuosi, relativamente in merito alla velocità e all'efficacia dei risultati ottenuti.

Di fatto, l'Histogram riesce ad elaborare una elevata quantità di immagini in un lasso tempo molto breve e ottiene immagini molto simili all'input, purché i colori abbiano poche sfumature.

Per quanto riguarda l'SSIM riusciamo ad ottenere dei risultati soddisfacenti, se la struttura dell'immagine è simile con le altre fotografie dell'archivio. In merito al tempo di elaborazione del confronto, l'SSIM occupa una quantità di tempo maggiore rispetto agli altri due, in quanto non è ottimizzato per il multithreading (al contrario del SIFT che invece è ottimizzato).

Anche l'interfaccia ha occupato la sua parte di tempo, infatti, abbiamo dovuto dedicare diversi giorni prima di riuscire ad implementare tutte le funzionalità in modo chiaro e di facile comprensione da parte dell'utente.

Ci possiamo ritenere soddisfatti di come il nostro programma funzioni, anche se con alcune immagini gli algoritmi non si comportano come ci si aspetterebbe, dando come output delle fotografie troppo diverse rispetto all'input.

Inoltre, ci siamo accorti che la nostra continua ricerca della personalizzazione del programma ha portato a far diventare le impostazioni della ricerca molto tecniche e difficilmente capibili da utenti senza una conoscenza pregressa, se non con una spiegazione iniziale dell'applicazione. Per cercare di ovviare a questo problema abbiamo creato la pagina *Guida*, per aiutare e indirizzare l'utente verso l'algoritmo che soddisfi al meglio le sue necessità.

Le immagini del DataSet che abbiamo realizzato sono di media-alta qualità e molto diversificate tra loro ed è stato positivo constatare che il programma fosse funzionale ed efficiente con queste.

Ci siamo trovati bene nel gruppo e sebbene siamo in pochi e inizialmente inesperti in merito, crediamo di aver fatto nel complesso un ottimo lavoro e siamo soddisfatti di come abbiamo lavorato.

A causa della pandemia abbiamo dovuto affrontare un'altra sfida: il lavorare su un progetto comune "a distanza". Per risolvere questa difficoltà abbiamo lavorato con l'applicazione "Discord" che permette di fare videochiamate, anche in gruppo, e di condividere lo schermo del proprio computer. Per facilitarci lo scambio di dati abbiamo creato una cartella condivisa su un NAS per potervi accedere in qualsiasi momento da tutti i pc e avere tutti i dati sempre aggiornati in tempo reale.

Verso la conclusione del progetto e quando abbiamo dovuto fare l'integrazione tra la parte degli algoritmi e dell'interfaccia abbiamo anche utilizzato l'applicazione "AnyDesk", la quale ci ha permesso di collegarci, in remoto, al pc master (il computer computazionalmente più potente e veloce), il quale è stato configurato con tutte le librerie utili per il programma (ad esempio quelle di Python) e ha consentito a tutti di modificare e integrare il codice.

La creazione di questo progetto ci ha permesso di metterci in gioco, con nuovi compiti e di ragionare assieme per trovare le soluzioni alle problematiche sorte, spronandoci ad aiutarci a vicenda e collaborando come in un vero team.