UNIVERSITA' DEGLI STUDI DI PAVIA

FACOLTA DI INGEGNERIA

DIPARTIMENTO DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA ELETTRONICA E INFORMATICA

Digital Anastylosis of Fresco challeNgE (DAFNE): Sviluppo del dataset e del relativo software

Relazione discussa in sede di esame finale

dal candidato:

RITA DI CECCA

Matricola 436999

Docente tutore: Prof. VIRGINIO CANTONI

A.A. 2017/2018

Indice

Introduzione

Il progetto svolto per il lavoro di tesi riguarda l'ambito della Computer Vision e in particolare della Pattern Recognition per la simulazione di frammentazione e ricostruzione di affreschi, cioè l'anastilosi digitale. La Computer Vision è una disciplina che richiede l'utilizzo di algoritmi che, a partire da un'immagine digitale in input, restituisce un'immagine modificata oppure un dato numerico o tabellare rappresentativo di una particolare caratteristica dell'immagine in input.

In questo contesto, sono stati progettati i tre software per l'anastilosi di affreschi e realizzati come contributo per un concorso internazionale chiamato "Digital Anastylosis of Frescoes challeNgE (DAFNE)". Quest'ultimo si pone come obiettivo la ricerca del migliore procedimento in grado di ricostruire più metri quadrati di un affresco a partire da migliaia di frammenti, di cui alcuni spuri provenienti da altri affreschi.

Questo concorso nasce per selezionare uno strumento utile ai restauratori impegnati nel centro Italia, dove diverse chiese sono state distrutte in seguito ai terremoti degli ultimi anni. Infatti, l'obiettivo a lungo termine sarebbe quello di poter utilizzare la soluzione proposta dal vincitore del concorso, per riposizionare i frammenti nella posizione corretta. Un esempio pratico è quello della chiesa di Norcia, dove sono stati rinvenuti molti frammenti e dove è già iniziato un lavoro preliminare di raccolta delle immagini di ogni singolo pezzo.

Fragment Image e Rebuild Image nascono per realizzare il dataset necessario per il concorso. In particolar modo, Fragment Image a partire dall'immagine .jpg o .png di un affresco, produce delle immagini (cioè i frammenti) in formato .png e altre informazioni necessarie ai concorrenti per la ricerca dell'algoritmo di ricostruzione. Rebuild Image, invece, a partire da ciò che viene prodotto dal programma precedente e in generale dalla proposta di soluzione dei concorrenti, restituisce un'immagine che mostra come i vari frammenti sono disposti sull'affresco.

Infine, l'interfaccia grafica è stata realizzata perché a questo concorso non partecipano solo università e centri di ricerca in Computer Science, ma anche ragazzi con disabilità come l'autismo. Questa scelta è stata fatta sia per rendere partecipi questi giovani e dare loro la

possibilità di cimentarsi in questo progetto, sia perché hanno delle abilità straordinarie nel pattern recognition, ricostruzione di puzzle ecc. Per loro, è stato realizzato questo software interattivo con cui possono muovere con le dita i frammenti e posizionarli dove ritengono giusto all'interno di un'immagine, che costituirà la soluzione del "puzzle" e fornirà automaticamente una tabella dove ogni frammento sarà abbinato alle coordinate e alla rotazione da applicargli per posizionarli nel posto giusto.

Nel **Capitolo 2** verrà esposto lo stato dell'arte attuale per quanto riguarda l'anastilosi digitale di affreschi.

Nel **Capitolo 3** verranno illustrate delle semplici nozioni sull'elaborazione di immagini applicate alla realizzazione dei tre software citati, mentre nel **Capitolo 4** verranno descritti in maniera approfondita quest'ultimi.

Infine, nel **Capitolo 5** verrà esposto come è stato prodotto il dataset per il concorso.

Capitolo 2

Stato dell'arte

L'anastilosi è un problema di elaborazione geometrica che ha importanti applicazioni in diversi campi come l'archeologia, il restauro artistico e la medicina legale. Per esempio, gli archeologi spendono molto tempo per ricomporre, a partire da frammenti, antichi dipinti al loro aspetto originale, gli specialisti forensi, invece, ricompongono manualmente documenti o immagini danneggiati per ottenere delle prove. Sviluppare algoritmi robusti per l'assemblaggio automatico può ridurre notevolmente il tempo impiegato per le ricostruzioni ed aumentare l'efficienza. Il valore nell'assemblaggio di oggetti rotti, o addirittura distrutti, a livello storico e culturale è legato al lavoro di ricostruzione di un puzzle. L'approccio classico nella risoluzione di puzzle è usare le informazioni riguardo la forma dei pezzi. Per questo motivo, sono state stilate alcune caratteristiche principali riguardo ad essi:

- orientamento: non necessariamente noto a priori
- connessione: definisce la possibilità di "buchi" o pezzi mancanti
- contorno esterno: definisce la forma di ogni pezzo, di solito noto a priori
- unicità: il puzzle può avere una o più soluzioni

La connessione in particolar modo è importante poiché può capitare che, a causa di influenze ambientali (erosione), piccoli pezzi vengano persi introducendo dei "buchi" tra i pezzi stessi.
Negli anni sono stati sviluppati diversi algoritmi, la maggior parte centrati sulla ricerca di bordi adiacenti tra frammenti, seguendo i loro contorni geometrici. Altri, invece, mappano la posizione e l'orientamento dei pezzi basandosi sulla correlazione con l'affresco originale. Se questo non è discriminante, ci si basa sulla semantica dei frammenti stessi, il loro contenuto cromatico oppure la continuità dei contorni.
Diversi tentativi sono stati fatti negli ultimi 60 anni. Per esempio, dal 1944 sono stati condotti diversi esperimenti nella ricostruzione di una cappella molto importante (A. Mantegna, Cappella Ovetari, Chiesa degli Eremitani in Padova), distrutta in 78861 pezzi da un bombardamento alleato nella Seconda Guerra Mondiale (1944). Nel 2006 è stata realizzata una ricollocazione digitale dei pezzi, ma solo l'8% di questi ha trovato una giusta locazione [1-7]. Un particolare della ricostruzione di un affresco di Mantegna è illustrato in Figura 2.1.

Figura 2.1 Particolare della ricostruzione di un affresco di Mantegna della Cappella degli Ovetari di Padova

Capitolo 3

L’anastilosi digitale

3.1 Definizione di immagine

Un’immagine è una funzione bidimensionale $f(x, y)$ dove x e y sono coordinate spaziali, e l’ampiezza di f per una certa coppia di valori (x, y) è detta *intensità* o livello di grigio o di colore dell’immagine in quel punto [9]. Se i valori di x, y e f sono discreti possiamo definire l’immagine come immagine digitale. Quest’ultima è composta da un numero finito di elementi detti pixel definiti dalla posizione all’interno dell’immagine e dal loro valore. Per questo motivo, un’immagine digitale è rappresentata come una matrice di pixel.
Il formato più diffuso è quello RGB: tre canali di colore (rosso, verde, blu) da 8 bit ciascuno che consente di riprodurre 16.7 milioni di colori. Inoltre, esiste una variante delle immagini RGB detta RGBA, dove A sta per alpha, un quarto canale che indica l’opacità del pixel. Se il valore di alpha per un certo pixel è zero esso sarà trasparente, altrimenti potrà essere opaco o con gradi diversi di trasparenza in base al valore di questo canale.
Il formato RGBA è stato utilizzato nei tre software, in particolare in Fragment Image per realizzare le immagini .png dei frammenti con uno sfondo trasparente in modo tale da poterli poi sovrapporre all’immagine originale convertita in bianco e nero.

3.2 Il diagramma di Voronoi

Un diagramma di Voronoi (Fig. 3.1), detto anche tassellatura di Voronoi, è la partizione di uno spazio (in questo caso di un’immagine) definito dalle distanze rispetto a un insieme finito di punti. Questo algoritmo è utilizzato in diversi campi scientifici: geografia, cristallografia, chimica, fisica ecc. Oltre ad essere una struttura versatile di partizionamento dello spazio, i diagrammi di Voronoi sono anche esteticamente piacevoli dal punto di vista artistico e giocano un ruolo unico dal punto di vista interdisciplinare.

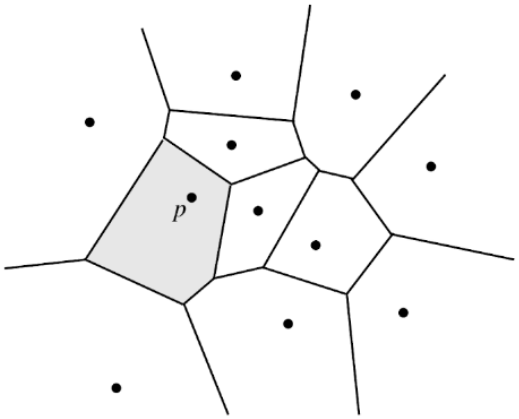

Figura 3.1 Diagramma di Voronoi di punti situati in uno spazio Euclideo

Con l'avvento dei computer moderni, l'importanza dei diagrammi di Voronoi è stata riscoperta. Infatti, vengono utilizzati per strutturare, rappresentare e visualizzare dati multidimensionali ma non solo: hanno applicazioni anche nel campo della robotica, computer graphics, classificazione di dati e clustering.

Considerando S un set di $n \geq 3$ punti *a, b, c* … nel piano Euclideo R^2, per i punti $a = (a_1, a_2)$ e $b = (b_1, b_2)$ la loro distanza Euclidea è data da

$$d(a, b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}.$$

Dato il segmento che collega *a* e *b*, la bisettrice *B* di quest'ultimi, chiamata *separatore*, è il luogo dei punti in R^2 alla stessa distanza da *a* e *b*. Inoltre, *B* è la linea perpendicolare del segmento $\overline{ab}$ e separa il semipiano vicino a *a* dal semipiano vicino a *b*. Le regioni che si creano con il set di punti iniziali sono poligoni convessi. La parte di confine in comune tra due regioni è detta *bordo*, se contiene più di due punti. Il diagramma di Voronoi è, quindi, l'unione di più bordi [8].

Un esempio di diagramma di Voronoi che utilizza la distanza Euclidea è illustrato in Figura 3.2.

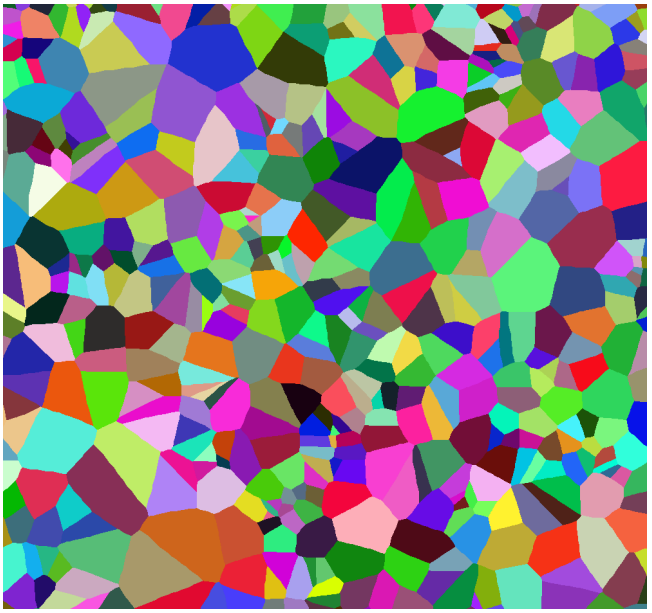

Figura 3.2 Diagramma di Voronoi utilizzando una distanza Euclidea

Esiste però un'altra distanza utilizzabile, detta *Manhattan* o *assoluta*, data da:

$$d(a, b) = \left|\left(a_1 - b_1\right) + \left(a_2 - b_2\right)\right|.$$

Con quest'ultima è possibile realizzare delle partizioni sia convesse che concave, come illustrato in figura 3.3. Come si può notare, i tasselli sono sia convessi che concavi.

Figura 3.3 Diagramma di Voronoi realizzato con distanza Manhattan

Nel software Fragment Image sono state utilizzate entrambe le distanze per realizzare una frammentazione degli affreschi il più realistica possibile.

3.3 L'operatore di Canny

L'obiettivo è quello di identificare delle discontinuità in un'immagine, in particolar modo i contorni. Questa ricerca si basa sulle caratteristiche dell'immagine stessa, per esempio sui livelli di grigio, la texture, la profondità della scena ecc.

Intuitivamente, un *contorno* è un set di pixel interconnessi che giacciono sul confine tra due regioni. Più nello specifico, è il luogo dove avviene un rapido cambiamento della funzione di intensità dell'immagine.

L'algoritmo che è stato utilizzato è quello di Canny [11] che si basa su tre passi:

- ricerca del gradiente della luminosità di un'immagine
- selezione dei massimi locali
- individuazione dei contorni mediante una soglia con isteresi

Prima di qualsiasi operazione, l'immagine di cui si vuole trovarne i contorni viene sottoposta a convoluzione con un *filtro gaussiano*. Un filtro è caratterizzato da un intorno e da un'operazione che deve essere eseguita sui pixels dell'immagine in quell'intorno. Il processo di filtraggio genera una nuova immagine spostando il filtro lungo tutta l'immagine. In particolare, il filtro gaussiano è un filtro di *smoothing* in quanto rimpiazza ogni pixel con la media pesata dei pixels in un suo intorno, secondo una funzione Gaussiana (vedi Fig. 3.4):

$$G_{\sigma}(x,y) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad \text{con } \sigma > 0$$

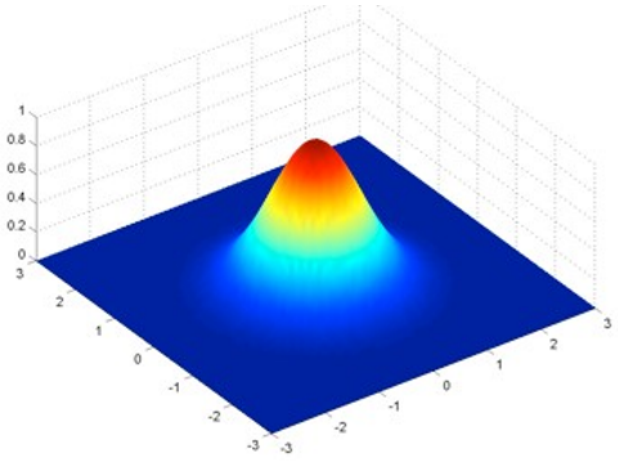

Figura 3.4 La funzione Gaussiana con $\sigma = 1$

Il risultato è un'immagine con una leggera sfocatura (*smoothing* appunto) in cui nessun pixel è affetto da disturbi in modo significativo.
Dopo l'applicazione di questo filtro, si procede con i passi elencati precedentemente.
Per la ricerca del gradiente di luminosità di un'immagine, dato che un contorno può puntare verso una direzione qualsiasi, si applicano dei filtri per individuare i contorni orizzontale, verticale e diagonali dell'immagine. Per ogni pixel, viene assunta come direzione quella relativa al filtro che dà il risultato maggiore, cioè la direzione in cui si ha il massimo gradiente di luminosità dell'immagine.
La soppressione dei non-massimi è un ulteriore passaggio che serve a discriminare con più certezza se un pixel fa parte di un contorno oppure no. Infatti, il gradiente non è sufficiente: solo i punti corrispondenti a dei massimi locali sono considerati appartenenti ad un contorno e possono essere presi in considerazione. Il massimo locale si ha se la derivata del gradiente è nulla. Quindi, al termine di questa fase, si avranno solo i punti che fanno parte di un bordo.
Con l'ultimo passaggio si estraggono i contorni dalla mappa generata dal passo precedente. Per ottenere ciò è necessario definire due soglie, una bassa e una alta, che vengono confrontate con il gradiente in ciascun punto di contorno:

- se il valore del gradiente del pixel è maggiore della soglia alta, il pixel è considerato come bordo vero
- se il valore del gradiente del pixel è al di sotto della soglia bassa, il pixel è scartato
- se il valore del gradiente del pixel si trova tra le due soglie, il pixel è considerato come bordo solo se è connesso ad un pixel che ha come gradiente un valore al di sopra della soglia alta

Al termine di questo passo si ottiene un'immagine binaria dove ciascun pixel è marcato come appartenente o no ad un contorno.
Nel software Fragment Image è stato utilizzato l'algoritmo di Canny per individuare i contorni dei frammenti generati con l'algoritmo di Voronoi.

3.4 Operazioni morfologiche

La morfologia matematica è uno strumento per estrarre componenti di immagini che sono utili per la rappresentazione e descrizione di una forma di una regione, come bordi e scheletri.

Per questo motivo, ha un riscontro nella soluzione di problemi nel campo dell'elaborazione d'immagini [10]. Noi useremo le operazioni morfologiche su immagini binarie, cioè immagini che possono avere solo due possibili valori per ogni pixel. Tipicamente, sono il bianco e il nero, in particolare il bianco per i punti dell'immagine e il nero per lo sfondo.

Le due operazioni primitive sono la *dilatazione* e l'*erosione*. Vengono utilizzate in diversi algoritmi e hanno un'ampia gamma di utilizzi, come la rimozione del rumore, la ricerca di discrepanze in un'immagine, l'isolamento o unione di elementi separati in un'immagine, ecc.

Entrambe le operazioni si basano sulla convoluzione di un elemento, detto *elemento strutturante*, lungo tutta l'immagine. L'elemento strutturante può avere qualsiasi forma: generalmente si utilizzano un cerchio, un quadrato o una croce.

Esistono anche combinazioni di queste due operazioni primitive, come per esempio l'*apertura* e la *chiusura*.

3.4.1 Dilatazione

La dilatazione è definita da:

$$A + B = \{ c \in E^n \mid c = a + b,\ a \in A,\ b \in B \}$$

Siccome l'elemento strutturante B è applicato a tutti i punti dell'immagine A, con la dilatazione si estende l'area dell'immagine (vedi Fig. 3.5).

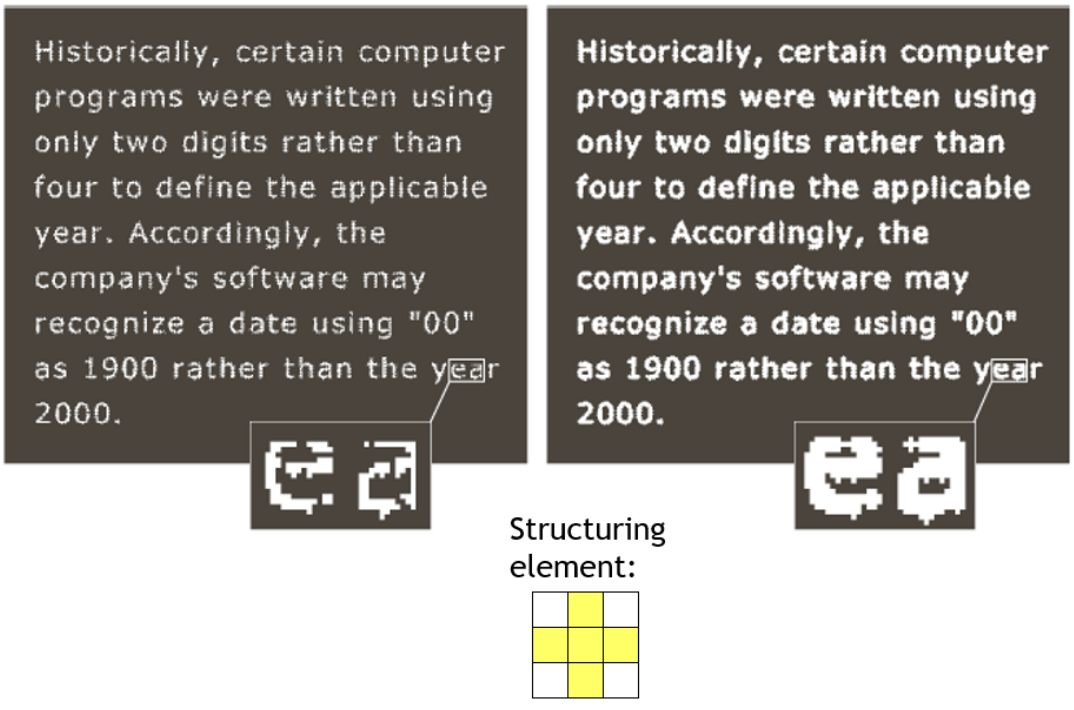

Figura 3.5 Esempio di dilatazione su un'immagine

3.4.2 Erosione

L'erosione è definita da:

$$A - B = \left\{ c \in E^n,\ c + b \in A,\ per\ ogni\ b \in B \right\}$$

Questa operazione è duale alla precedente, appartengono all'immagine finale solo i punti per i quali l'elemento strutturante B traslato sugli stessi resta tutto contenuto nell'immagine A. Con l'erosione otteniamo l'effetto contrario alla dilatazione: diminuzione di zone utili dell'immagine (da qui "erosione"), come si può vedere in Figura 3.6.

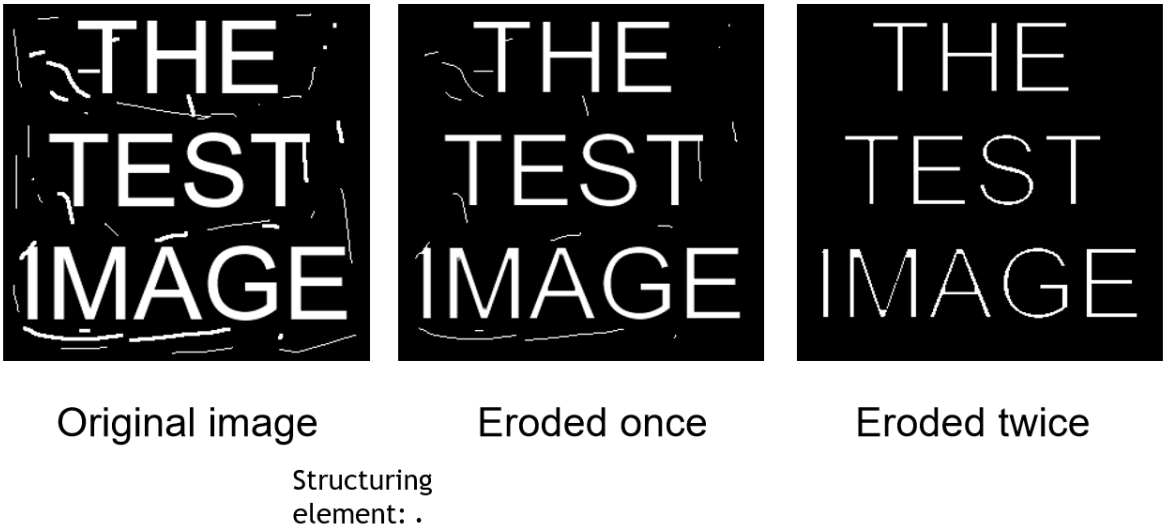

Figura 3.6 Esempio di erosione su un'immagine

L'erosione è stata utilizzata per ridurre la dimensione dei frammenti all'interno del software Fragment Image, in modo da rendere ancora più realistica la simulazione dei danni causati per esempio da un terremoto ad un affresco.

3.4.3 Apertura

Le due operazioni morfologiche precedenti possono essere combinate per realizzare un'altra operazione morfologica detta *apertura*. Essa consiste in un'erosione seguita da una

dilatazione. L'apertura è utilizzata per rimuovere il rumore da un'immagine, nell'esempio togliere pixel spuri e in generale parti piccole (vedi Fig. 3.7). Per questo motivo, è stata utilizzata nel software Fragment Image.

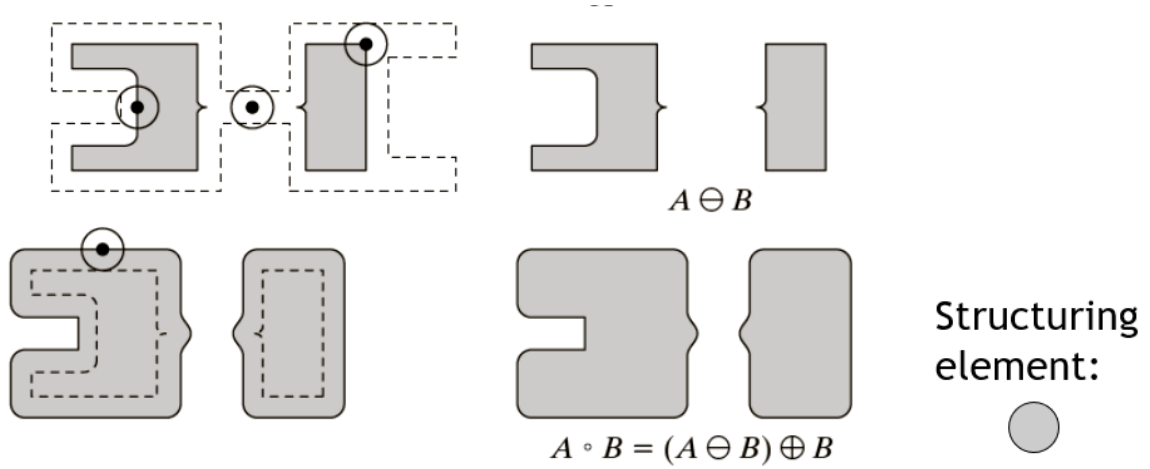

Figura 3.7 Esempio di apertura su un'immagine

3.4.3 Chiusura

Come per erosione e dilatazione, la chiusura è il duale dell'apertura. Essa consiste in una dilatazione seguita da un'erosione. Anch'essa è utilizzata per la rimozione del rumore all'interno dell'immagine, in particolare per riempire "i buchi" all'interno di un'immagine, come si può vedere in Figura 3.8.

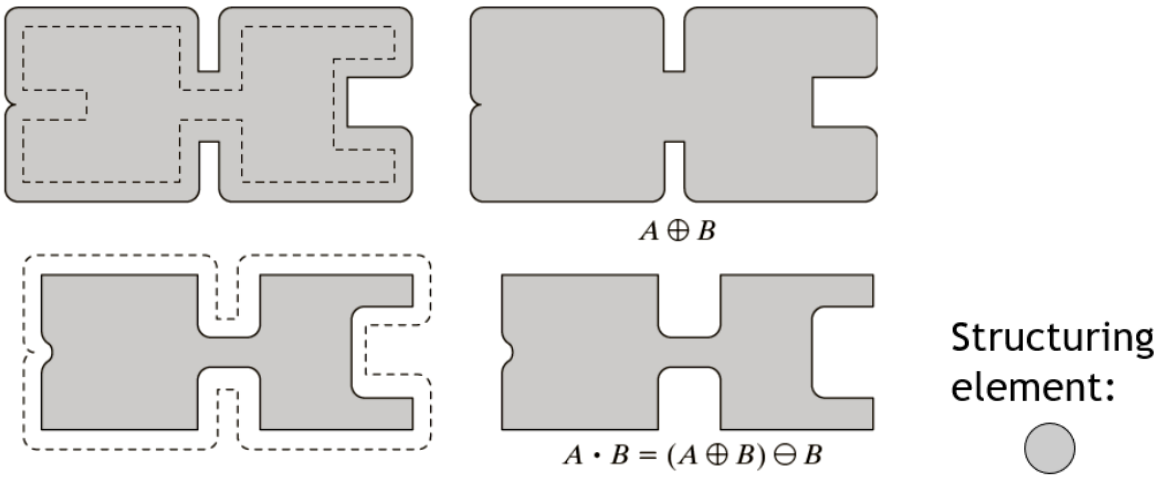

Figura 3.8 Esempio di chiusura su un'immagine

Capitolo 4

Descrizione dei tre software

4.1 Fragment Image

Il primo software che è stato realizzato è Fragment Image. L'obiettivo di quest'ultimo è quello di generare, a partire dall'immagine di un affresco scelto dall'utente, una serie di dati:

- una cartella denominata frag_eroded contenente i frammenti dell'affresco in input
- un'immagine denominata diagram_proba_map.png
- un'immagine denominata canny_image.png
- un'immagine denominata voronoi_color_map.png
- tre files txt : parameters.txt, fragment.txt e fragment_s.txt

Questi dati, che saranno a breve illustrati, sono necessari ai concorrenti per creare un algoritmo che sia in grado di ricostruire un'area maggiore di affresco.

Quest'applicazione è stata realizzata cercando di simulare una rottura che sia il più simile possibile ai danni causati, per esempio, da un terremoto. Diversi esempi di frammenti reali sono stati usati come riferimento, un esempio è mostrato nella Figura 4.1 e Figura 4.2.

Figura 4.1 Esempi di frammenti veri

Figura 4.2 Altri esempi di frammenti veri

I frammenti, per essere il più realistici possibili, devono rispettare alcune proprietà:

- varietà nelle dimensioni
- non devono presentare bordi acuti.
- possono essere sia convessi che concavi, ma non devono presentare due parti non connesse
- non devono essere troppo piccoli, per questo motivo vengono scartati se sono al di sotto di una certa soglia stabilita a priori
- la dimensione e la forma non dipendono dalla posizione all'interno dell'immagine

Figura 4.3 Esempio di frammento concavo

All'avvio dell'applicazione compare una schermata di console in cui viene richiesto l'inserimento dei parametri necessari per l'esecuzione e quindi per la frammentazione dell'immagine dell'affresco.

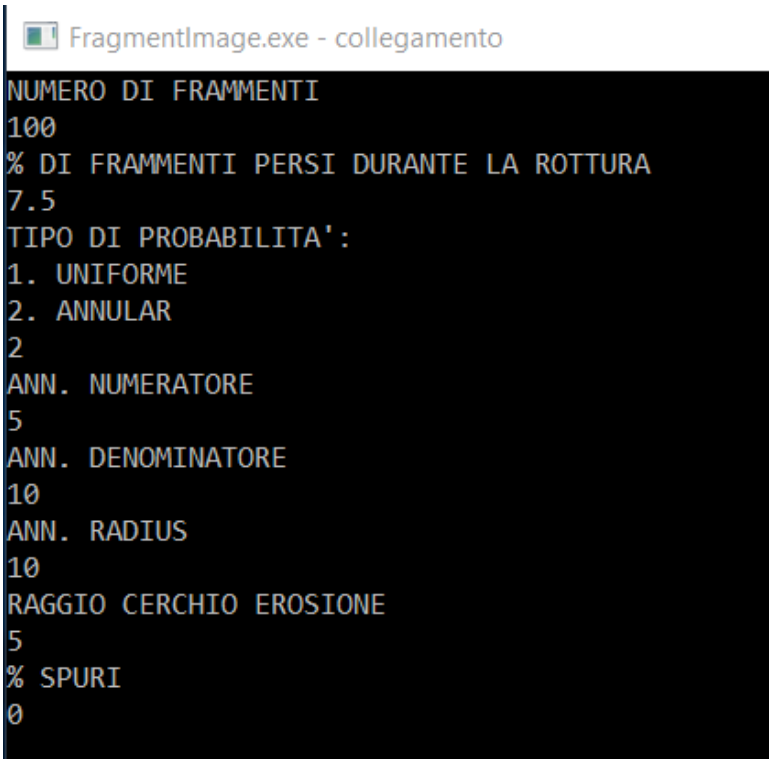

Figura 4.4 Esempio di esecuzione di Fragment Image

All'utente viene richiesto di inserire:

- il numero di frammenti che si vogliono ottenere dalla rottura dell'affresco
- la percentuale dei frammenti persi durante la rottura

- il tipo di statistica con cui verranno disposti i centri dei frammenti all'interno dell'immagine
- il raggio del cerchio per l'erosione che smusserà i contorni
- la percentuale di frammenti spuri provenienti da altri affreschi

Successivamente, viene richiesto all'utente di scegliere, tramite una GUI, l'immagine dell'affresco da frammentare.

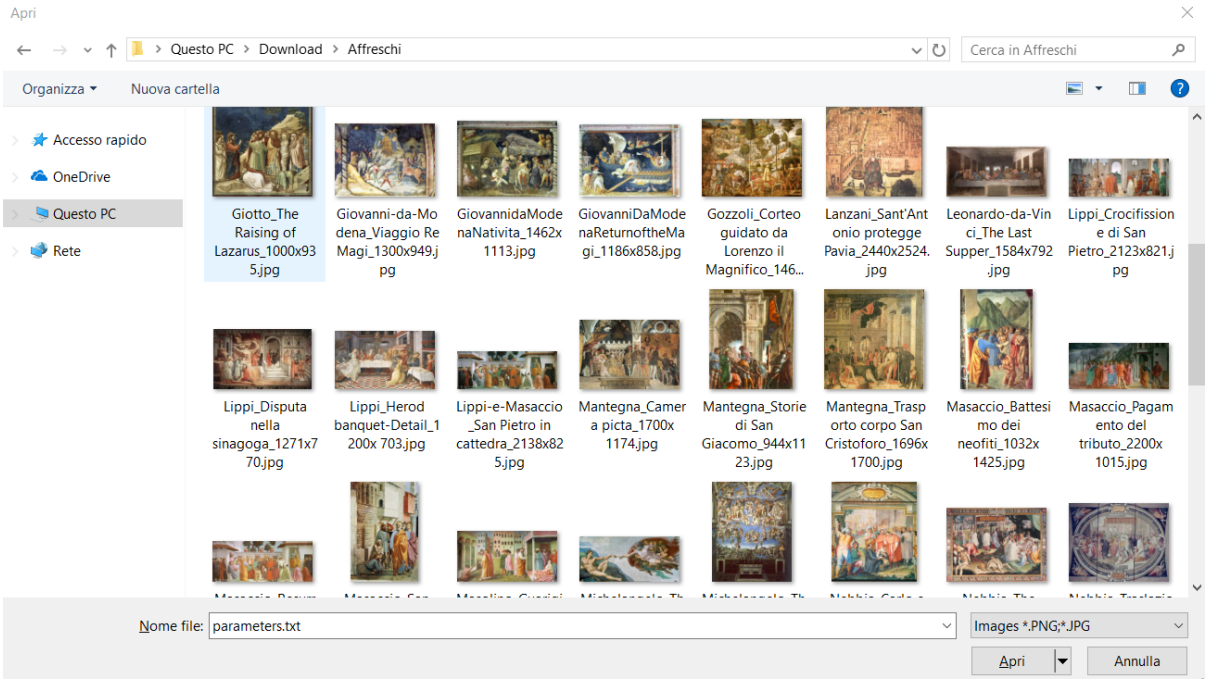

Figura 4.5 Finestra di dialogo per la scelta dell'affresco

Il primo passo è la costruzione dell'immagine diagram_proba_map.png. Quest'ultima rappresenta come sono stati presi i punti *p* per applicare l'algoritmo di Voronoi (come in figura 3.1). Sono state implementate due diverse probabilità: *uniforme* e *concentrata*.

La prima tipologia è quella più semplice in quanto vengono selezionati i punti all'interno di un'immagine secondo una distribuzione uniforme. È possibile selezionarla inserendo il valore "1" quando viene richiesto il tipo di probabilità all'avvio dell'applicazione.

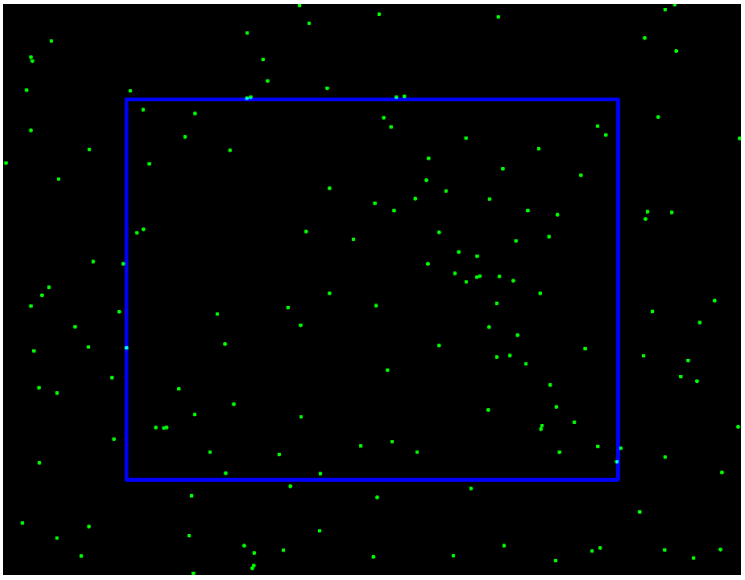

Figura 4.6 Esempio di diagram_proba_map.png con distribuzione uniforme

Il secondo tipo di probabilità utilizzata è quella concentrata, in modo tale da creare zone con un maggior numero di piccoli frammenti al contrario di altre in cui ci saranno di conseguenza frammenti più grandi. Così facendo, si crea una varietà nella dimensione. Per selezionarla, è necessario inserire il valore "2" all'avvio dell'applicazione quando viene richiesto il tipo di probabilità.

L'implementazione di quest'ultima è più complessa: ogni volta che viene selezionato un punto, si disegna una corona circolare bianca all'interno dell'immagine. Le corone circolari sono centrate nel punto che è appena stato preso. Successivamente, la probabilità di prendere un punto all'interno della corona circolare è aumentata.

Questa probabilità è controllata da alcuni parametri:

- denominatore della probabilità del cerchio *d*
- numeratore della probabilità del cerchio *n*
- raggio del cerchio esterno *r*
- bordo della corona circolare *b*

dove $\frac{n}{d}$ è la probabilità di prendere un punto che non sia all'interno della corona circolare,

$1 - \frac{n}{d}$ è la probabilità di prendere punto interno, $r - b$ il raggio interno.

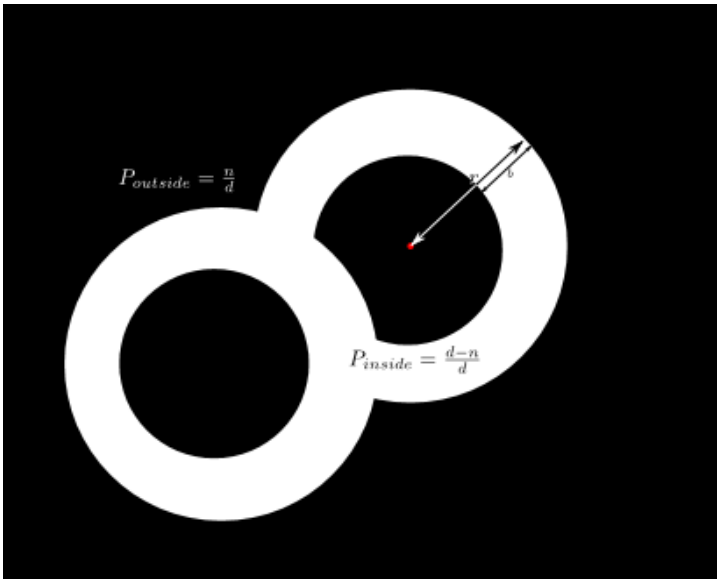

Figura 4.7 In chiaro l'area in cui può cadere il punto successivo con un'assegnata probabilità (realizzando una maggior concentrazione di punti) rispetto al complemento in spazio e in probabilità

Il risultato che si ottiene con una probabilità di questo tipo è mostrato in figura 4.8.

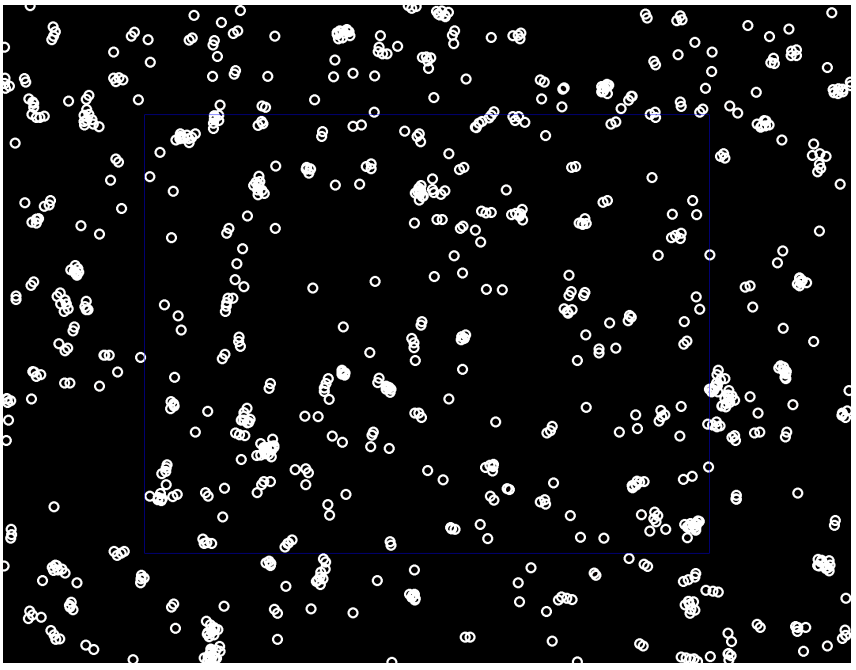

Figura 4.8 Esempio di diagram_proba_map.png con distribuzione concentrata

In entrambe le distribuzioni è stata considerata un'immagine più grande di un fattore 1.5 di quella presa in considerazione. Questo ingrandimento è stato realizzato per risolvere il problema delle aberrazioni sui frammenti più vicini al bordo dell'immagine. Ciò è dovuto al fatto che se il punto più vicino a quello selezionato è oltre il bordo stesso dell'immagine, viene a mancare e quindi condiziona le forme dei frammenti dell'intorno. Per questo motivo, considerando un'immagine più grande dell'originale, si considerano centri per possibili

frammenti anche al di fuori dell'immagine stessa, realizzando una frammentazione omogenea anche sui bordi.

Dopo la selezione dei centri dei frammenti, l'immagine viene ritagliata alle dimensioni effettive. Adesso, è possibile applicare l'algoritmo di Voronoi. Per fare questo, è stata utilizzata la funzione *distanceTransform* della libreria OpenCV, come si può vedere in Figura 4.9.

```
if (probability_type == 1)
    distanceTransform(initial_image, output, labels, CV_DIST_L1, CV_DIST_MASK_PRECISE, CV_DIST_LABEL_CCOMP);
if (probability_type == 2)
    distanceTransform(initial_image, output, labels, CV_DIST_L2, CV_DIST_MASK_PRECISE, CV_DIST_LABEL_CCOMP);
```

Figura 4.9 Implementazione della distanceTransform

Questa funzione restituisce delle matrici in output che contengono informazioni riguardo gli id (o label) che sono stati assegnati ad ogni frammento, dopo averli generati applicando Voronoi. Inoltre, è possibile passare come parametro CV_DIST_L1 nel caso in cui si voglia usare la distanza Manhattan, altrimenti CV_DIST_L2 per la Euclidea. In particolare, è stata scelta una distanza assoluta nel caso in cui si voglia utilizzare una probabilità di posizionamento dei centri di tipo uniforme, altrimenti l'euclidea nel caso di probabilità concentrata.

Una volta generati i frammenti, possiamo costruire il diagramma di Voronoi sulla base della segmentazione fatta dalla distanceTransform e ottenere delle immagini come in figura 3.2 e 3.3, ovvero l'immagine voronoi_color_map.png.
Da quest'ultima, è stato utilizzato l'algoritmo di Canny per delineare i contorni dei frammenti da sovrapporre successivamente all'immagine dell'affresco scelto dall'utente (vedi Fig. 4.10).

```
cvtColor(voronoi_color_map, voronoi_gray, COLOR_BGRA2GRAY);
Canny(voronoi_gray, voronoi_gray, 0, 1);
dilate(voronoi_gray, voronoi_gray, MORPH_DILATE);
cvtColor(voronoi_gray, sobel_bgr, COLOR_GRAY2BGR);
bitwise_or(sobel_bgr, image, out);
```

Figura 4.10 Implementazione di Canny e sovrapposizione dei contorni all'immagine dell'affresco

Essendo i frammenti anche molto piccoli e soprattutto vicini, sono state scelte volutamente due soglie per l'algoritmo di Canny molto basse e vicine tra loro in modo tale da identificare

tutti i contorni in maniera precisa. Successivamente, è stata applicata una dilatazione in modo da rendere i contorni più visibili.

Il risultato ottenuto per i due tipi di probabilità è mostrato in Figura 4.11 e 4.12.

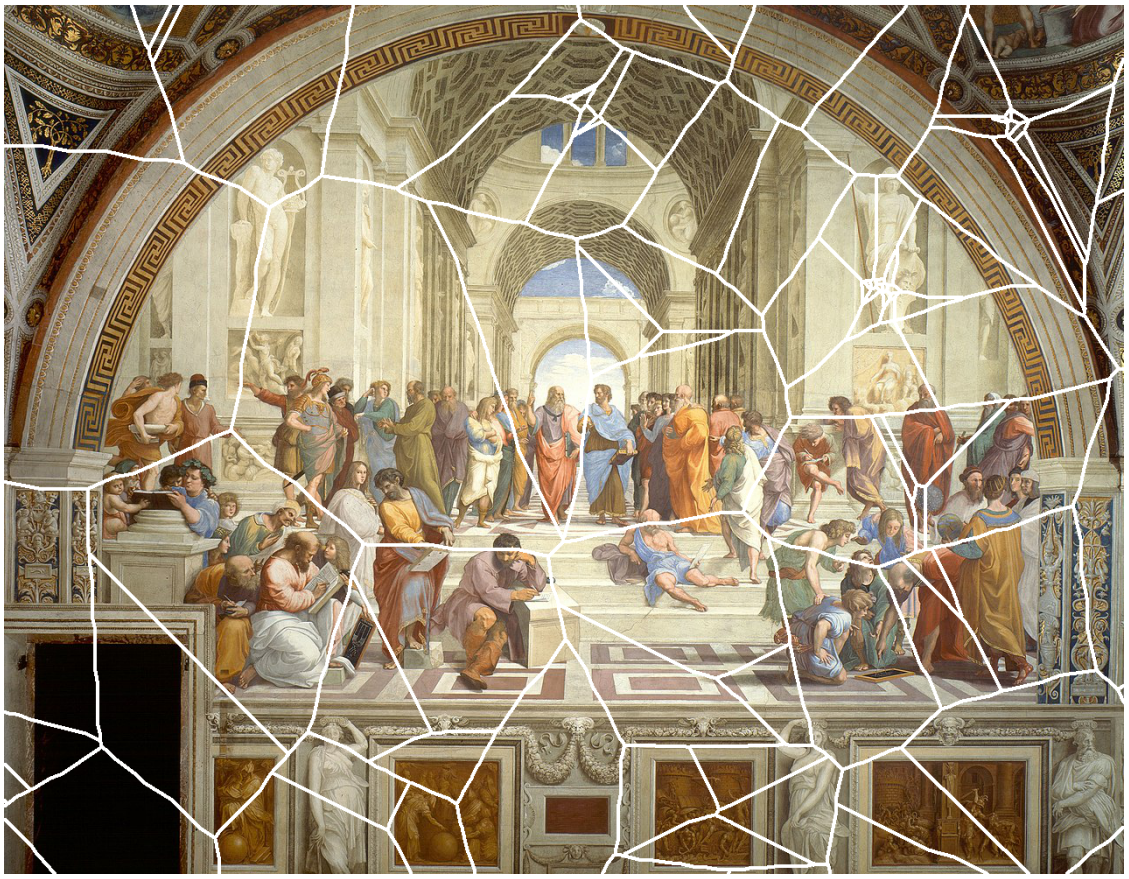

Figura 4.11 Canny_image di Raffaello “Scuola di Atene”, distribuzione concentrata

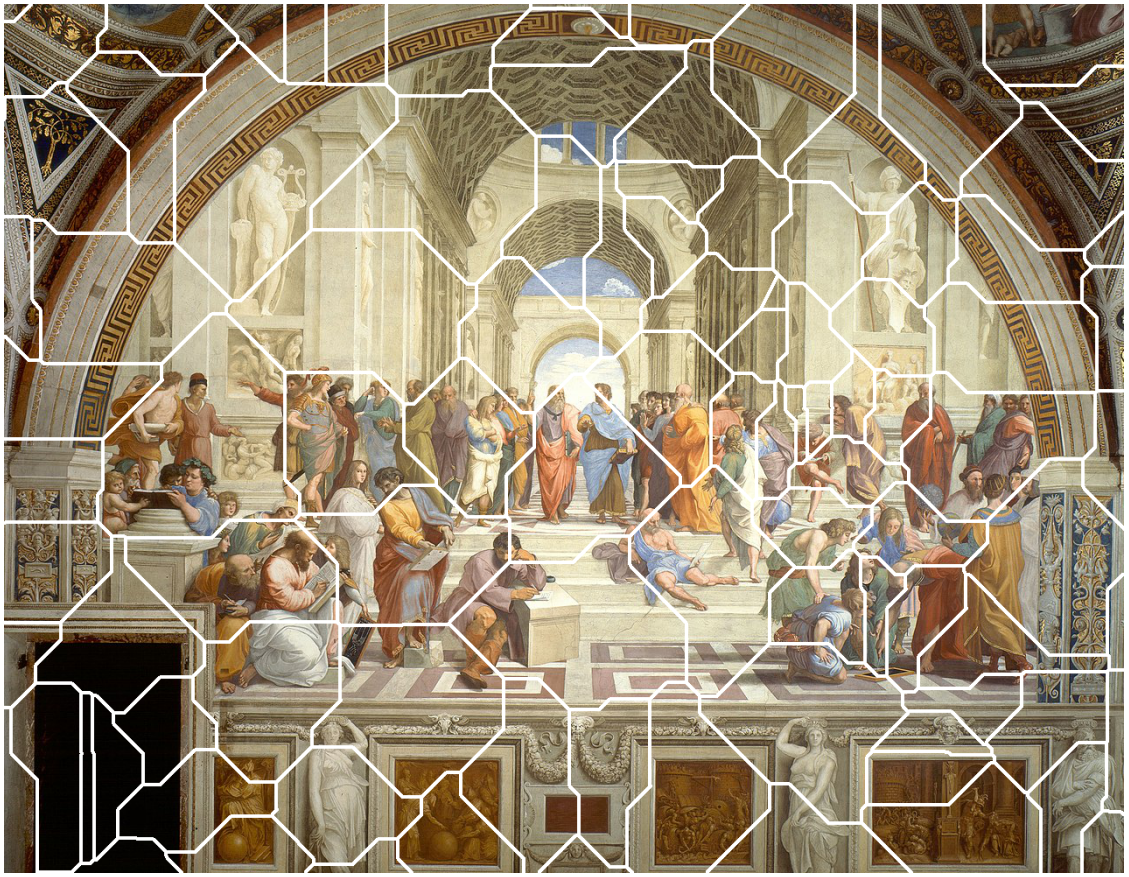

Figura 4.12 Canny_image di Raffaello “Scuola di Atene”, distribuzione uniforme

Mentre viene analizzata la matrice di output dell’algoritmo di Voronoi, oltre alla costruzione della voronoi_color_map, vengono anche creati i singoli frammenti. Il risultato di questa analisi è una serie di matrici, una per ogni frammento, delle dimensioni effettive del frammento stesso. Qui avviene un primo controllo sulle dimensioni: infatti, se la diagonale della matrice è al di sotto di un certo valore di soglia stabilito a priori, il frammento viene scartato. In questo modo, si evita di elaborare matrici che non verranno considerate e si migliorano le prestazioni. Infine, se il frammento ha superato questo controllo, viene ruotato di un angolo generico compreso tra 0 e 360 gradi. Anche ciò è stato implementato con lo scopo di realizzare una simulazione il più realistica possibile: durante un terremoto i frammenti non cadranno esattamente nella posizione corretta, ma ruotati. Inoltre, sicuramente una parte dell’affresco stesso diventa polvere, da qui la scelta di scartare, durante tutta l’esecuzione del codice, i frammenti troppo piccoli.

Per migliorare ulteriormente le prestazioni, in questo punto del codice viene effettuata la rimozione random di alcuni frammenti secondo la percentuale inserita all’avvio dell’applicazione. Viene generato un numero casuale compreso tra 0 e il numero di frammenti inseriti dall’utente e, attraverso un ciclo, si eliminano i frammenti dalla lista e quindi le

matrici dalla memoria. Non viene fatta alcuna distinzione nell'eliminazione. Anche questo passaggio è voluto: sicuramente, durante la rottura dell'affresco, alcuni pezzi saranno distrutti. Implementando la perdita dei frammenti in questa parte del software, possiamo alleggerire ancora di più le elaborazioni successive diminuendo il numero di matrici da dover analizzare.

A questo punto, uno dei modi per degradare i frammenti è quello di implementare l'operazione morfologica di erosione, descritta nel Capitolo 3. Questo passaggio è fondamentale perché è ciò che rende ancora più realistico il dataset prodotto: i frammenti dell'affresco che potremmo trovare all'interno di una chiesa dopo un evento naturale difficilmente saranno delle dimensioni originali, ma sicuramente più piccoli. In particolare, sono state effettuate due erosioni (vedi Fig. 4.13).

```
erode(binary_mask, erosion_dst, structural_element);
erode(erosion_dst, erosion_crc, circ);
```

Figura 4.13 Implementazione dell'erosione

La prima erosione viene effettuata non con un elemento strutturante noto come il cerchio, la croce ecc., bensì con il frammento più piccolo, tra quelli che sono rimasti dopo la prima eliminazione dei frammenti più piccoli. In questo modo, si realizza un'erosione più "casuale", diversa di esecuzione in esecuzione, creando dei frammenti con forme più particolari ed eliminando definitivamente quelli troppo piccoli, come si può vedere in Figura 4.14.

Figura 4.14 Confronto prima dell'erosione e dopo l'erosione. L'ultima immagine è una sovrapposizione delle prime due

Successivamente, viene effettuata una seconda erosione utilizzando un cerchio con un raggio della dimensione scelta dall'utente all'avvio dell'applicazione, per arrotondarne i bordi. In

questo modo, soddisfiamo una delle proprietà dei frammenti, cioè quella di non avere bordi appuntiti. Questo smussamento è illustrato in Figura 4.15.

Figura 4.15 Confronto prima e dopo l'erosione. L'ultima immagine è una sovrapposizione delle prime due

Inoltre, è stata applicata un'apertura, in modo tale da avere frammenti che consistano in un solo pezzo e non avere residui nel caso in cui per esempio una parte non dovesse essere più collegata al frammento stesso.

Anche dopo questa fase, viene fatta un'ulteriore rimozione di frammenti troppo piccoli: dopo l'erosione, potrebbero essere di nuovo al di sotto dell'area minima stabilita all'inizio.

L'ultimo passaggio per rendere la simulazione più realistica è l'aggiunta di frammenti spuri. Per fare ciò, è richiesto all'utente di scegliere, tramite una GUI, la cartella contenente i frammenti spuri. Quest'ultimi sono stati precedentemente selezionati e inseriti nella directory di riferimento.

Per uniformare tutti i frammenti, essi vengono rinominati e salvati in una cartella denominata frag_eroded in ordine di grandezza, in particolare in ordine decrescente.

L'ultimo passaggio è il salvataggio dei tre files .txt:

- in parameters.txt vengono salvati tutti i valori dei parametri utilizzati per l'esecuzione, compresi quelli inseriti dall'utente
- in fragment.txt vengono salvati in ordine id e le informazioni necessarie ai concorrenti sui frammenti effettivi dell'affresco: id, posizione (x, y) del centro del frammento rispetto all'immagine dell'affresco e angolo di cui è stato ruotato
- in fragment_s.txt sono presenti gli id dei frammenti spuri

Un esempio di output prodotto è illustrato in Figura 4.16.

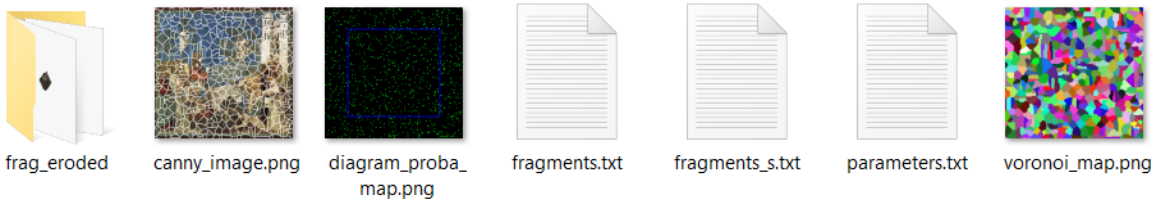

Figura 4.16 Esempio dell'output prodotto

4.2 Rebuild Image

Il secondo software realizzato è Rebuild Image, complementare a Fragment Image. Infatti, lo scopo di questo software è quello di mostrare come sono disposti i frammenti (generati precedentemente) nella posizione corretta. Per fare ciò, essi vengono sovrapposti all'immagine dell'affresco da cui sono stati generati, dopo che quest'ultima è stata convertita in bianco e nero. Inoltre, essendo i frammenti erosi, con Rebuild Image si può vedere chiaramente il lavoro finale: quanto è stata aggressiva l'erosione, se la posizione salvata nei file di testo è corretta, ecc. L'output di questa applicazione è un'immagine rebuilt_image.png che contiene l'immagine di riferimento in bianco nero sullo sfondo e i frammenti generati da Fragment Image a colori sovrapposti.

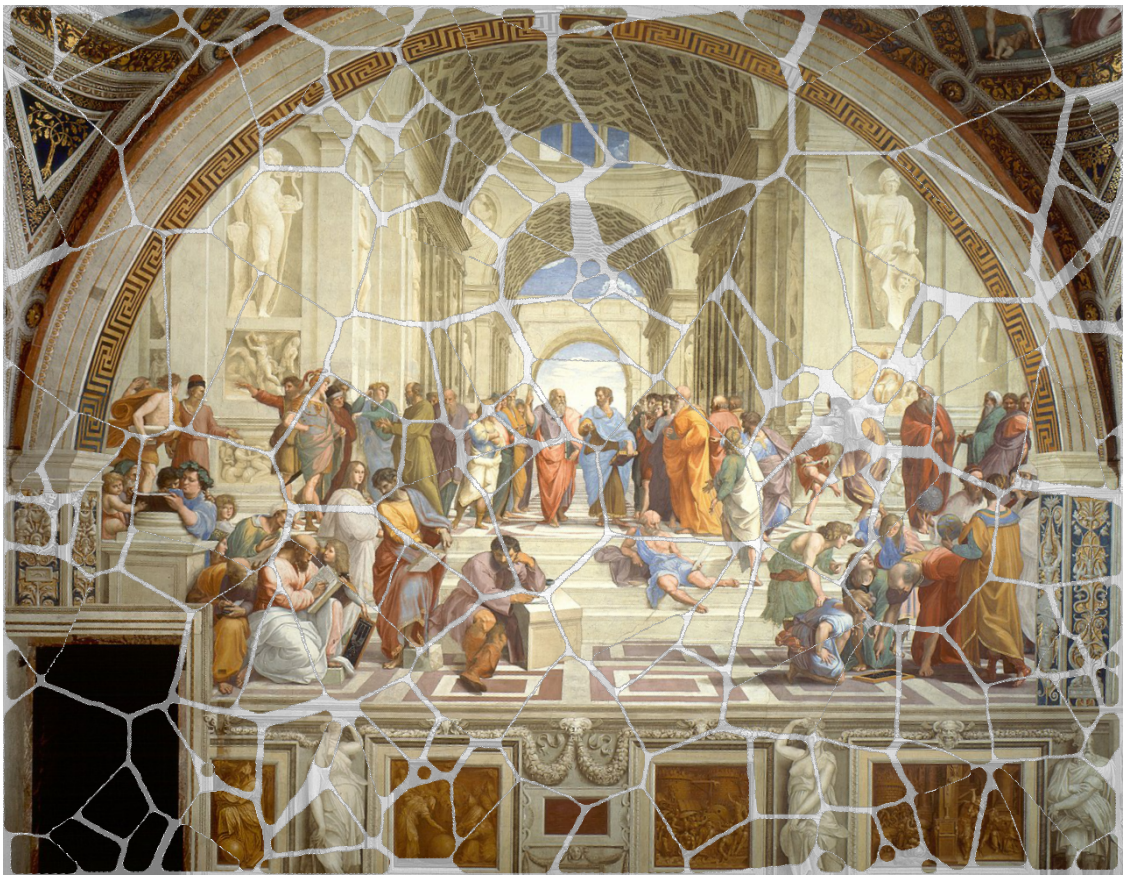

Figura 4.17 Esempio di ricostruzione dei frammenti su Raffaello, “Scuola di Atene”, distribuzione concentrata

In Figura 4.17, per esempio, si vede ad occhio che i frammenti combaciano quasi perfettamente, come se fosse un puzzle. Ciò è dovuto al fatto che è stata implementata un’erosione poco aggressiva (raggio del cerchio per lo smussamento piccolo) ed un numero di frammenti di partenza basso, in questo modo essendo quasi tutti di dimensioni grandi, non verranno eliminati nei controlli di dimensione. Infatti, le uniche zone in cui si hanno “buchi” sono dove c’è la concentrazione di frammenti piccoli.

Figura 4.18 Esempio di ricostruzione dei frammenti su Raffaello, “Scuola di Atene”, distribuzione uniforme

In Figura 4.18, invece, possiamo vedere ancora che i frammenti combaciano quasi come se fosse un puzzle, ma anche che le dimensioni sono pressoché uguali, proprio perché è stata utilizzata una distribuzione uniforme.

In entrambi i casi, solo i frammenti relativi all’affresco sono stati riportati sull’immagine, per cui gli spuri sono stati scartati.

Per l'implementazione di questo software è stato necessario:

- leggere il file di testo fragment.txt generato precedentemente
- leggere le immagini .png dei frammenti nella cartella frag_eroded
- ruotare e posizionare i frammenti nella coordinata (x, y) corretta

Affinché la posizione fosse la più corretta possibile, ogni frammento è stato salvato al centro di un rettangolo leggermente più grande del frammento stesso. Inoltre, anche per la ricostruzione è stata considerata l'immagine con un fattore di ingrandimento di 1.5. L'area in più è stata resa trasparente ed è denominata *padding*.
In questo modo, quando tutti i frammenti sono stati posizionati, quei pochissimi pixel che eccedono la dimensione dell'immagine vengono esclusi poiché l'immagine verrà ritagliata alle sue dimensioni effettive.

Per realizzare Fragment Image e Rebuild Image è stato utilizzato il linguaggio di programmazione a oggetti C++, l'ambiente di sviluppo o IDE (Integrated Development Environment) Microsoft Visual Studio e la libreria OpenCV.

4.3 Interfaccia grafica per la ricostruzione di affreschi

L'ultimo software realizzato è una GUI che permette la ricostruzione manuale di affreschi. Lo scopo di questa interfaccia grafica è di dare la possibilità ai ragazzi autistici di cimentarsi in questo progetto in modo del tutto interattivo e, se vogliamo, ludico.
Quest'applicazione è composta da due schermate principali: finestra iniziale e finestra di gioco.
La finestra iniziale è stata realizzata come mostrato in Figura 4.19:

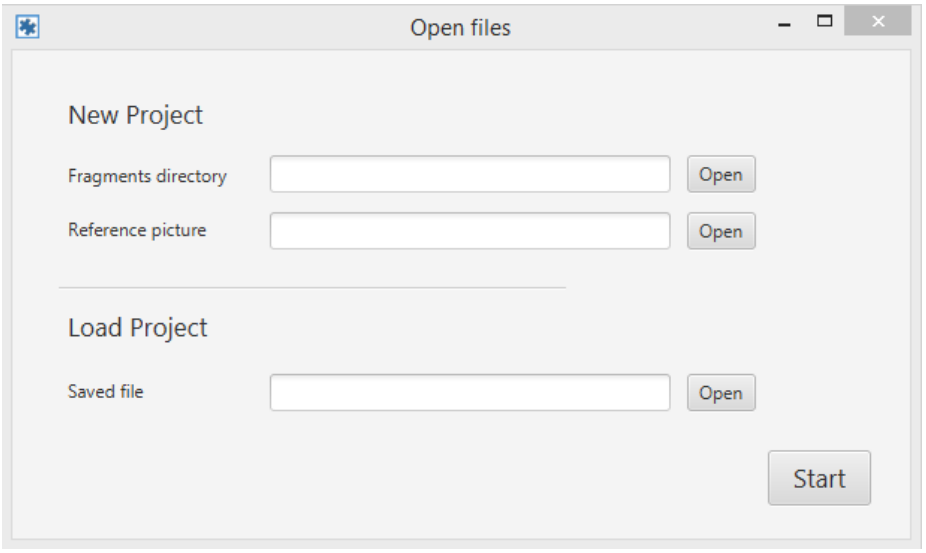

Figura 4.19 Interfaccia iniziale

Essa permette due possibilità: la ripresa di un progetto iniziato e salvato precedentemente oppure l'inizio di un nuovo progetto. Nel primo caso basta semplicemente cliccare su "Open" e caricare il file del salvataggio presente sul computer. Nel secondo caso è necessario dover scegliere la cartella dei frammenti che si vogliono ricomporre e l'immagine dell'affresco di riferimento.

Attraverso il bottone "Start" possiamo passare alla schermata di gioco vera e propria.

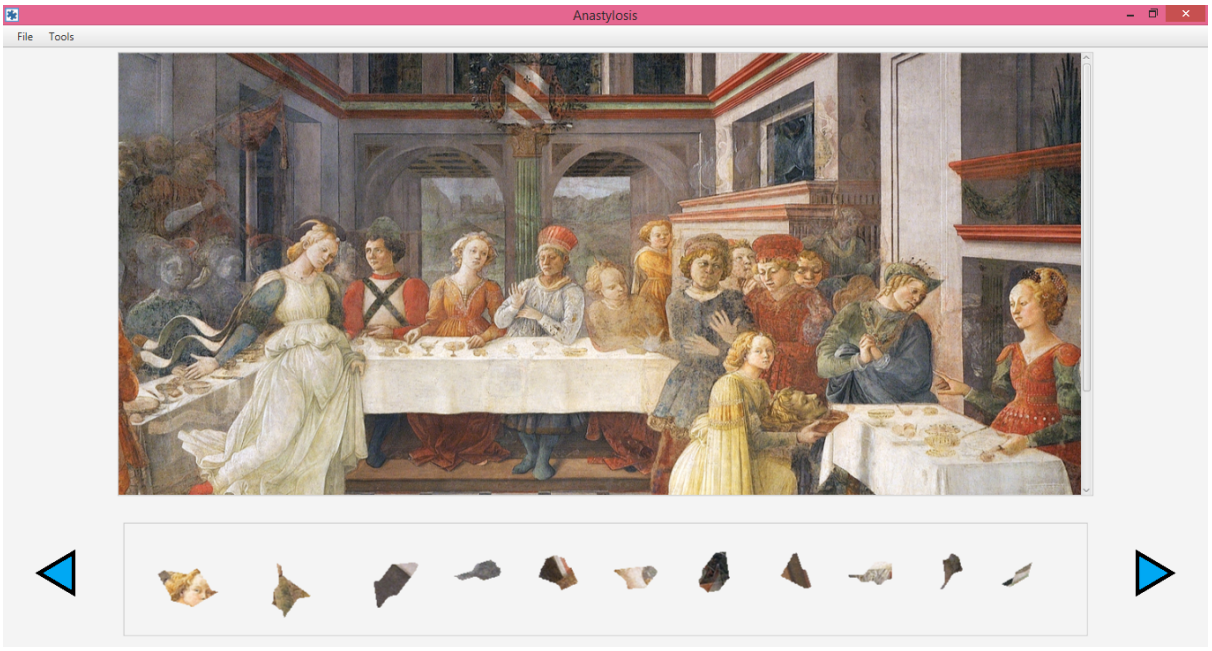

Figura 4.20 Schermata di gioco

La schermata mostrata in Figura 4.20 è composta da:

- un pannello principale contenente l'immagine di riferimento
- un pannello secondario contenente i frammenti di quell'immagine

- due frecce che permettono lo scorrimento della lista dei frammenti

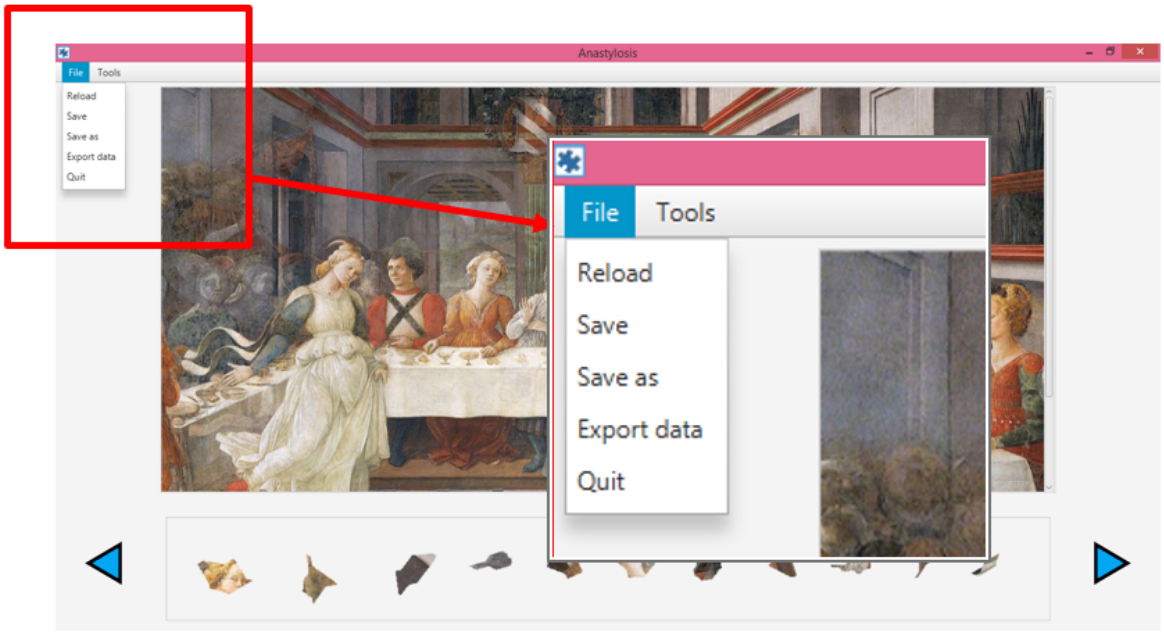

Figura 4.21 Menu della schermata di gioco

Inoltre, si può vedere che in alto a sinistra è presente un Menu a tendina (vedi Fig. 4.21) che permette di:

- caricare un altro progetto e abbandonare quello corrente
- salvare il progetto in corso
- salvare il progetto in corso con un altro nome
- export data permette di creare un file .txt contenente gli id dei frammenti, le coordinate (x, y) in base alla posizione in cui sono stati disposti sull'immagine di riferimento e l'angolo con cui sono stati ruotati
- abbandonare il gioco

Inoltre, vicino al bottone File, è presente "Tools", che dà la possibile di convertire l'immagine di riferimento dell'affresco in bianco e nero o di riportarla a colori. In aggiunta, si potrebbe lasciare lo sfondo bianco e fare la ricostruzione di un'immagine di cui non si conosce l'aspetto originario.

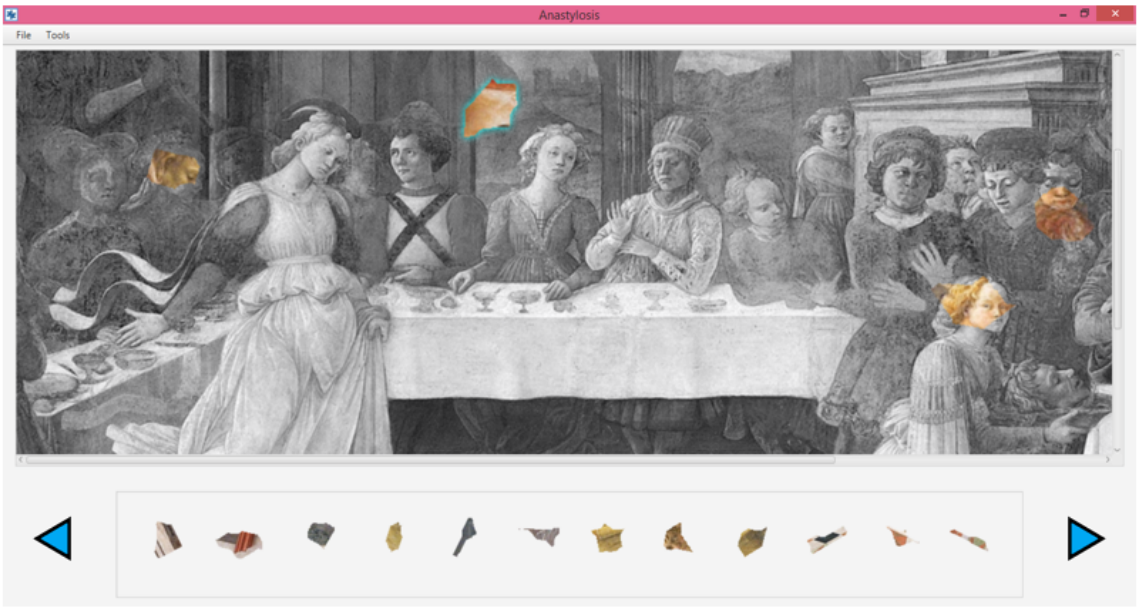

Figura 4.22 Esempio di esecuzione

La figura 4.22 mostra un esempio di esecuzione dell'interfaccia grafica. Per ricostruire l'affresco, i ragazzi possono giocare come più desiderano. Infatti, è possibile muovere i frammenti in ogni punto dell'immagine semplicemente trascinandoli col mouse dal pannello inferiore alla posizione voluta e, per rendere l'applicazione ancora più intuitiva, quando un frammento viene selezionato compare un'area celeste attorno. Se il cursore del mouse si trova su di esso, scrollandolo si può ruotare il frammento di qualsiasi angolo. Inoltre, in qualsiasi punto dell'immagine è possibile fare lo zoom. Se si volesse togliere un frammento dal pannello superiore, premendo il tasto destro del mouse si potrebbe riportarlo nel pannello inferiore. Tutte le funzionalità precedentemente descritte restano valide anche in presenza di uno schermo touch screen. È necessario semplicemente muovere le dita sullo schermo. Per la realizzazione di questo software è stato utilizzato il linguaggio di programmazione Java FX e l'ambiente di sviluppo IntelliJ Idea.

Capitolo 5

Realizzazione del dataset

Per la realizzazione del dataset necessario ai partecipanti sono stati considerati sessantatré affreschi:

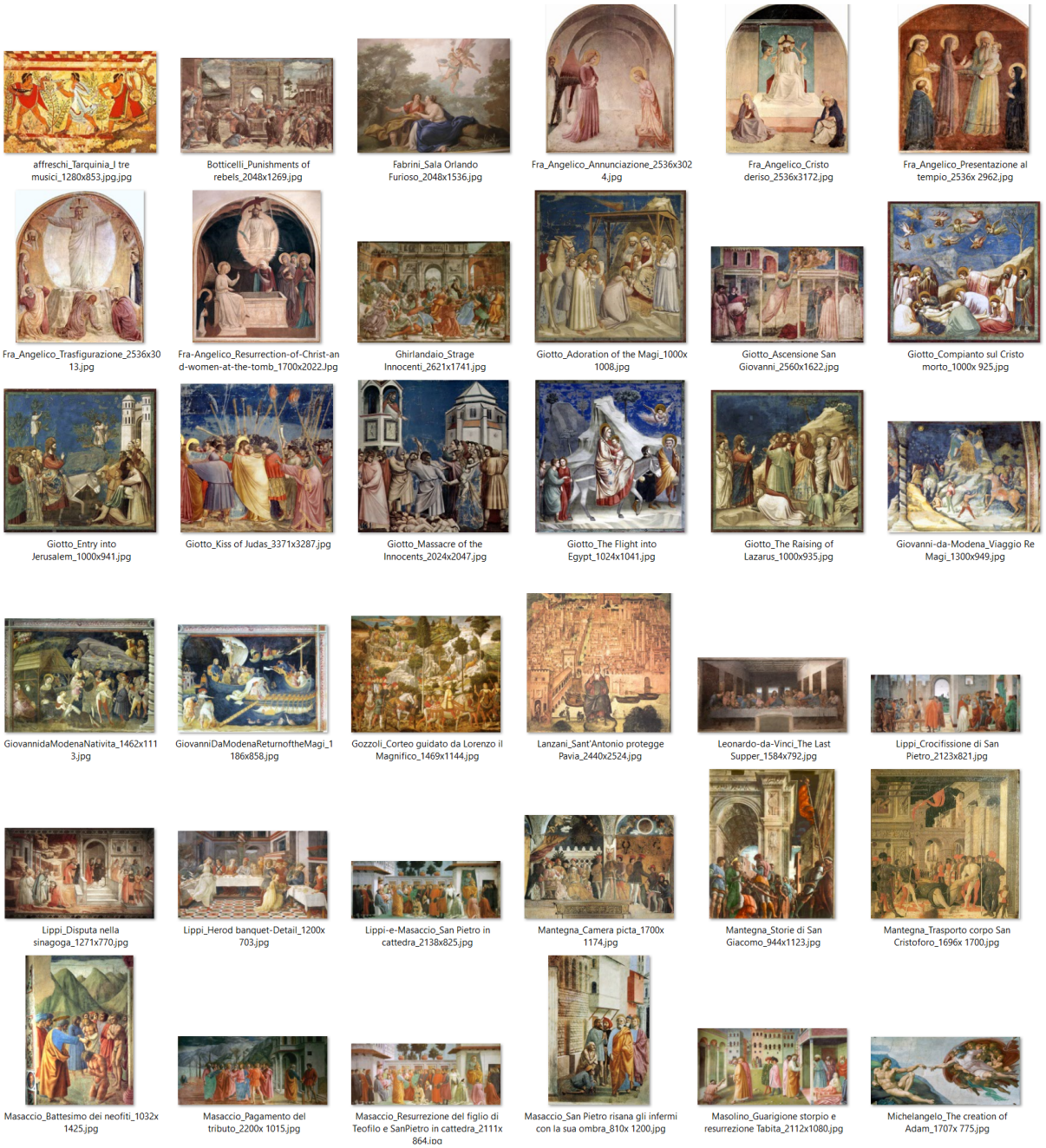

Figura 5.1 Prima parte di affreschi utilizzati per il dataset

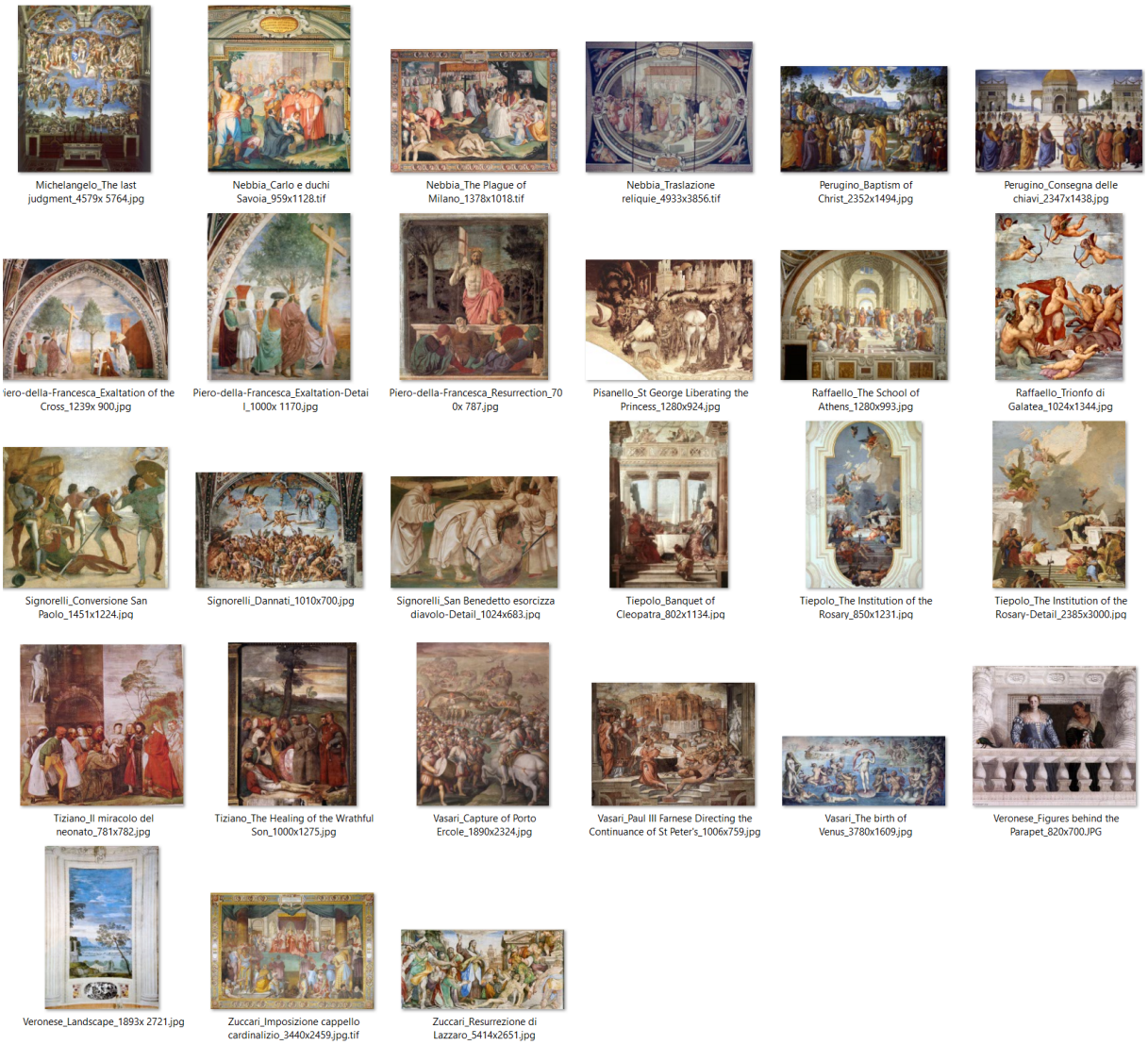

Figura 5.2 Seconda parte di affreschi utilizzati per il dataset

Per ogni affresco sono state realizzate diciotto esecuzioni ognuna con parametri diversi. In particolare:

- nove esecuzioni utilizzando la distribuzione concentrata e nove con quella uniforme.
- con percentuale di frammenti persi di valore 7,5%, 15% oppure 22.5%
- con percentuale di frammenti spuri di valore 0%, 7,5 % oppure 15%
- con raggio del cerchio per lo smussamento dei bordi deciso per ogni affresco in base al numero dei frammenti e alla dimensione dell'immagine
- probabilità di selezionare un punto al di fuori della corona circolare e raggio del cerchio più esterno nel caso di distribuzione concentrata

Più precisamente, sono state realizzate combinazioni tra i valori possibili di percentuale di frammenti persi e spuri, ovvero: per ogni percentuale di frammenti persi, tutti e tre i tipi di percentuale di quelli spuri. In questo modo quindi per ogni affresco sono state eseguite due

tipi di distribuzioni per tre tipi di perdita per tre tipi di quantità di frammenti spuri con variazioni sulla probabilità nel caso di distribuzione concentrata, in base alla dimensione dell'affresco e al numero dei pezzi.

Prima delle esecuzioni, è stato svolto un lavoro preliminare per assegnare ad ogni affresco un numero di frammenti e una dimensione del raggio di erosione congruo con la dimensione dell'immagine, in modo tale da non generare frammenti troppo piccoli o troppo grandi.
Affinché le esecuzioni potessero procedere velocemente per l'apertura del concorso, sono stati realizzati dei file batch contenenti le informazioni necessarie per le esecuzioni (parametri, percorsi delle immagini). In questo modo, è stato possibile avere tutte le casistiche in poco tempo senza richiedere la presenza fisica di un utente all'elaboratore.

FragmentImage.bat - Blocco note

File Modifica Formato Visualizza ?

```
set in_dir=C:\Users\ritad\Documents\Tesi\FragmentImage\data\Esecuzioni
set sp_dir=C:\Users\ritad\Documents\Tesi\FragmentImage\data\Spuri

set name_img_1=GiottoEntryIntoJerusalem

mkdir %in_dir%\%name_img_1%

FragmentImage.exe 500 7.5 1 0 0 0 8 0 %in_dir%\%name_img_1%.jpg %name_img_1% %sp_dir%
FragmentImage.exe 500 7.5 1 0 0 0 8 7.5 %in_dir%\%name_img_1%.jpg %name_img_1% %sp_dir%
FragmentImage.exe 500 7.5 1 0 0 0 8 15 %in_dir%\%name_img_1%.jpg %name_img_1% %sp_dir%
FragmentImage.exe 500 15 1 0 0 0 8 0 %in_dir%\%name_img_1%.jpg %name_img_1% %sp_dir%
FragmentImage.exe 500 15 1 0 0 0 8 7.5 %in_dir%\%name_img_1%.jpg %name_img_1% %sp_dir%
FragmentImage.exe 500 15 1 0 0 0 8 15 %in_dir%\%name_img_1%.jpg %name_img_1% %sp_dir%
FragmentImage.exe 500 22.5 1 0 0 0 8 0 %in_dir%\%name_img_1%.jpg %name_img_1% %sp_dir%
FragmentImage.exe 500 22.5 1 0 0 0 8 7.5 %in_dir%\%name_img_1%.jpg %name_img_1% %sp_dir%
FragmentImage.exe 500 22.5 1 0 0 0 8 15 %in_dir%\%name_img_1%.jpg %name_img_1% %sp_dir%

FragmentImage.exe 500 7.5 2 5 10 10 8 0 %in_dir%\%name_img_1%.jpg %name_img_1% %sp_dir%
FragmentImage.exe 500 7.5 2 5 10 10 8 7.5 %in_dir%\%name_img_1%.jpg %name_img_1% %sp_dir%
FragmentImage.exe 500 7.5 2 5 10 10 8 15 %in_dir%\%name_img_1%.jpg %name_img_1% %sp_dir%
FragmentImage.exe 500 15 2 5 10 10 8 0 %in_dir%\%name_img_1%.jpg %name_img_1% %sp_dir%
FragmentImage.exe 500 15 2 5 10 10 8 7.5 %in_dir%\%name_img_1%.jpg %name_img_1% %sp_dir%
FragmentImage.exe 500 15 2 5 10 10 8 15 %in_dir%\%name_img_1%.jpg %name_img_1% %sp_dir%
FragmentImage.exe 500 22.5 2 5 10 10 8 0 %in_dir%\%name_img_1%.jpg %name_img_1% %sp_dir%
FragmentImage.exe 500 22.5 2 5 10 10 8 7.5 %in_dir%\%name_img_1%.jpg %name_img_1% %sp_dir%
FragmentImage.exe 500 22.5 2 5 10 10 8 15 %in_dir%\%name_img_1%.jpg %name_img_1% %sp_dir%

pause
```

Figura 5.3 File batch utilizzato per le esecuzioni, in questo caso per l'affresco di Giotto "Entry into Jerusalem"

In Figura 5.3 è stato illustrato il file batch utilizzato per le esecuzioni. In questo caso, l'affresco di Giotto "Entry into Jerusalem" richiedeva un numero di frammenti pari a 500 sia nel caso di distribuzione uniforme che concentrata. A seguito, sono stati inseriti gli altri parametri necessari all'esecuzione presentati precedentemente. Gli ultimi tre, invece, contengono le informazioni riguardo ai percorsi relativi all'immagine da utilizzare e alla cartella degli spuri. Come si può vedere, essendo le esecuzioni diciotto per ogni affresco, sono

presenti diciotto linee all'interno del file batch, oltre a quelle necessarie per la creazione delle cartelle e delle variabili.

Dato che non tutti i file prodotti sono destinati ai concorrenti, ogni esecuzione prevede la creazione di due cartelle principali di cui una denominata "PRIVATE" che contiene ciò che non viene fornito adesso. Inoltre, ogni cartella è denominata dal nome dell'affresco e dalla data e l'ora in cui è stata fatta l'esecuzione.

Quanto esposto precedentemente è mostrato in Figura 5.4.

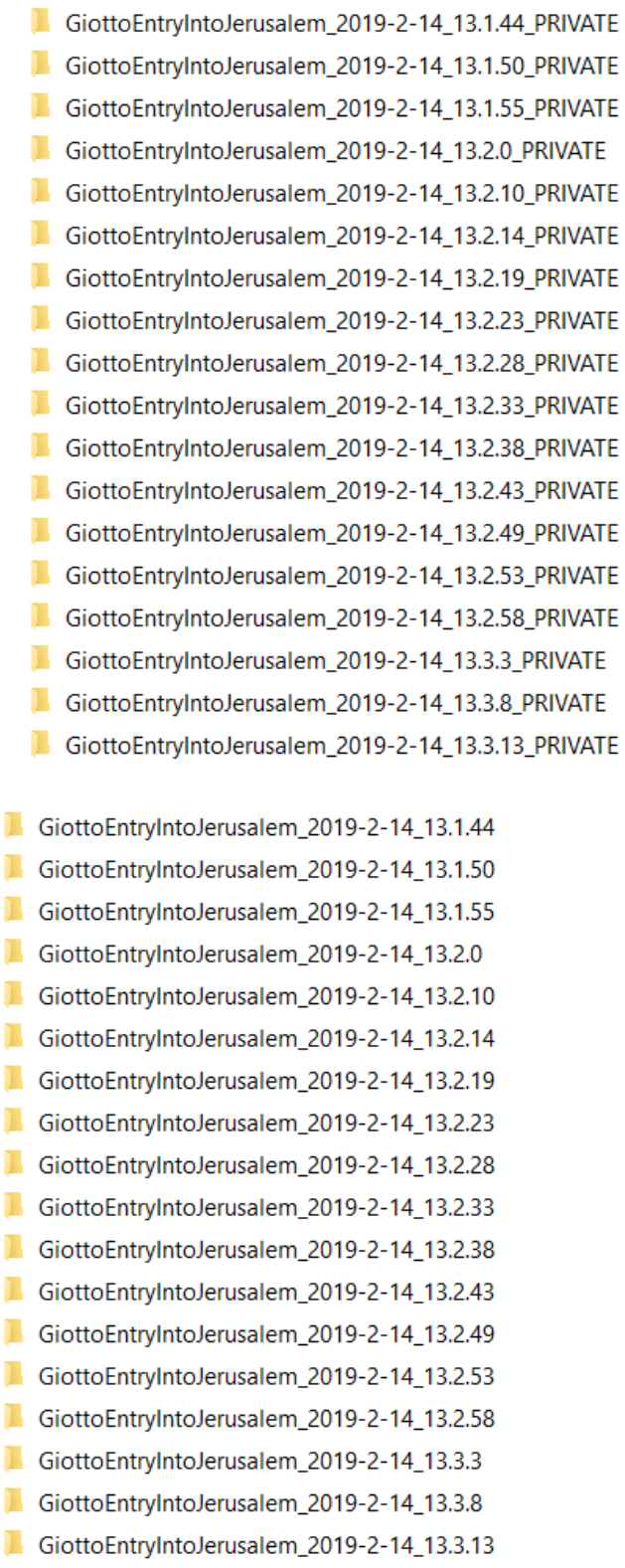

Figura 5.4 Esempio di esecuzione completa di Giotto “Entry into Jerusalem”

Capitolo 6

Conclusioni

I tre software sono stati sviluppati per realizzare il dataset necessario per il concorso DAFNE (Digital Anastylosis of Frescoes challeNgE) e permettere a ragazzi con disabilità, come l'autismo, di cimentarsi in questo progetto. L'output di Fragment Image e Rebuild Image sono dei file e delle immagini contenenti informazioni riguardo i pezzi (id, angolo, coordinate, contorni, distribuzione ecc.) dell'affresco da dover segmentare, passato in ingresso. L'interfaccia grafica, invece, permette la ricostruzione manuale dell'immagine dell'affresco semplicemente utilizzando un mouse o le dita, trascinando i pezzi sullo schermo.

Durante la realizzazione di Fragment Image e Rebuild Image sono state riscontrate difficoltà legate alla gestione delle immagini e alle prestazioni. In particolare, è stato necessario elaborare sia l'immagine in input che le matrici contenenti i pezzi per poter applicare le operazioni morfologiche ed operatori come Canny, per esempio dover convertire le immagini in binario, ritagliarle alle dimensioni effettive, convertirle in scala di grigio o a colori e così via. Dal punto di vista delle prestazioni, invece, è stato necessario studiare il flusso delle operazioni che permettesse un'esecuzione efficace ed efficiente senza occupare troppa memoria centrale o tempo. A questo scopo, è stato necessario analizzare il codice per far sì che:

- liberasse la memoria appena possibile, ovvero de-allocare le matrici non prettamente essenziali dalla memoria centrale
- venissero utilizzati pochi cicli for per scansionare ogni immagine poiché è dispendioso in termini di tempo
- venisse ridotto in termini di righe, cioè fare in modo di avere un software che, con pochi passaggi essenziali, riuscisse a produrre il valore o l'output richiesto e che fosse leggibile

Nella tabella sottostante sono illustrati i tempi di esecuzione dei software Fragment Image e Rebuild Image, eseguiti uno dopo l'altro, su un elaboratore con:

- 8 GB di RAM

- processore Intel® Pentium ® CPU 4415U @ 2.30 GHz
- Windows 10 Home a 64 bit
- SSD: Samsung MZVLW128HEGR-000L2

Numero di frammenti	Tempo di esecuzione medio
5000	33s
4000	26s
3000	21s
2000	15s
1000	8s
500	4s

Non sono stati riportati altri dati oltre al numero di frammenti in quanto non influiscono sui tempi di esecuzione, ciò che discrimina quest'ultimi è il numero di frammenti.
Nei tempi di esecuzione medi sono stati esclusi i tempi di immissione dei parametri in input.

Per quanto riguarda l'interfaccia grafica, invece, non sono state riscontrate difficoltà particolari se non quella di realizzare una GUI che fosse il più intuitiva possibile.

Questo progetto di tesi mi ha permesso di imparare tantissime cose sia di informatica che non. Ho imparato un linguaggio di programmazione nuovo, conoscenze sull'elaborazione di immagini e Computer Vision, a focalizzarmi sulle prestazioni, a scrivere un codice efficiente e leggibile, a leggere documentazioni sulle librerie e molto altro. Inoltre, ho avuto la possibilità di vivere in prima persona lo sviluppo di un progetto che è evoluto con il tempo e che mi ha messo di fronte ad obiettivi sempre nuovi, spingendomi ad imparare sempre di più, ed a collaborare con professori diversi, accettando i loro consigli.

Elenco delle figure

Bibliografia

[1] H. Shin, C. Doumas, T. Funkhouser, S. Rusinkiewicz, K. Steiglitz, A. Vlachopoulos, and T. Weyrich, Analyzing and simulating fracture patterns of Theran wall paintings., *ACM Journal on Computing and Cultural Heritag*e, Vol. 5, No. 3, Article 10: 1-14, October 2012.

[2] F. Kebler, R. Sablatnig, A Survey of Techniques for Document and Archaeology Artefact Reconstruction, *10th International Conference on Document Analysis and Recognition*, IEEE Computer Society, pp-1061-1065, 2009.

[3] W. Youguang, X. Li, M. Li, Color and Contour Based Reconstruction of Fragmented Image, *8th International Conference on Computer Science & Education (ICCSE 2013)*, IEEE Conferences, pp. 999-1003, 2013.

[4] M. Fornasier, Faithful Recovery of Vector Valued Functions from Incomplete Data. Recolorization and Art Restoration, F. Sgallari, A. Murli, N. Paragios (Eds.), SSVM 2007, LNCS 4485, Springer-Verlag Berlin Heidelberg, pp. 116-127, 2007.

[5] M. Fornasier, Nonlinear Projection Recovery in Digital Inpainting for Color Image Restoration, *J. Math Imaging Vis.*, 24, pp. 359-373, Springer, 2006.

[6] M. Fornasier, D. Toniolo, Fast,robust and efficient 2D pattern recognition for re-assembling fragmented images, *Pattern Recognition*, 38, pp. 2074-2087, Elsevier, 2005.

[7] F. Amigoni, S. Gazzani, S. Podico, A method for reassembling fragments in image reconstruction, *Proceedings International Conference on Image Processing*, IEEE Conferences, Vol. 3, pp. III-581, 2003.

[8] F. Aurenhammer, R. Klein, D. Lee, Voronoi diagrams and Delaunay triangulations, World Scientific, Singapore, 2013.

[9] V. Cantoni, S. Levialdi, B. Zavidovique, 3C VISION: Cues, Contexts, and Channels, Elsevier insights, London, 2011.

[10] J. Serra, G. Matheron, Image Analysis and Mathematical Morphology, Academic Press, London 1982.

[11] J. Canny, A Computational Approach to Edge Detection, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, No. 6, pp. 679-698, 1986.

Ringraziamenti

Vorrei ringraziare il Prof. Cantoni per avermi dato la possibilità di lavorare a questo progetto interessante, per aver sempre creduto in me e nelle mie potenzialità, soprattutto quando ne ho avuto più bisogno.
Un grazie speciale al Dott. Dondi per avermi seguito negli ultimi mesi, aiutandomi sempre e dandomi un sacco di consigli. Ho imparato davvero un sacco di cose grazie a te.
Infine, un grazie enorme all'intero staff del Laboratorio di Visione Artificiale, in particolar modo alla Dott. Setti e al Prof. Lombardi per la loro disponibilità.

Ringrazio la mia famiglia per tutto il supporto ricevuto in questi anni, per avermi dato la possibilità di proseguire gli studi e per avermi spinto a fare sempre di più.

Ringrazio Francesco perché in questi anni, seppur tra alti e bassi, sei stato la mia roccia. Mi hai spronato ad affrontare le mie insicurezze e gli ostacoli incontrati durante questo percorso.

Ringrazio Benni e Arianna perché mi hanno sopportato e supportato. Siete un riferimento e so che posso sempre contare su di voi. Non potevo chiedere amiche migliori.

Ringrazio i miei amici nonché compagni di corso per aver reso questi tre anni fantastici. In particolare, ringrazio i miei compagni di progetti Jack, Stefano, Mattia (quando finiamo il Trivial Pursuit?) e i miei compagni di uscite del sabato sera Dario, Carola, Simone, Martina (quando il prossimo sushi?).

Ringrazio Barba per avermi ascoltato in questi anni e per avermi dato forza, anche solo con un messaggio, quando ne avevo più bisogno.

Ringrazio Alessia per esserci stata sempre… non serve esser vicine per tenere forte un'amicizia.

Ringrazio i miei cari che purtroppo mi hanno lasciato troppo presto e non potranno esserci in questo giorno speciale, ma so che, ovunque essi siano, saranno sempre con me.

Infine, un ringraziamento va anche a me stessa. Come disse qualcuno… “A smooth sea never made a skillful sailor”.

Grazie di cuore a tutti.