PROGETTO PER IL CORSO DI PROGRAMMAZIONE E MODELLAZIONE AD OGGETTI

LABORATORIO DI ELABORAZIONE DI IMMAGINI IN SCALA DI GRIGI

CHRISTEL SIROCCHI

MATRICOLA 293310

A.A. 2019/2020

1. Specifica del software

Il progetto consiste nello sviluppo di un software per l'elaborazione di immagini in scala di grigi con interfaccia grafica implementata con Windows Form.

Lo sviluppo di strumenti per la visualizzazione e l'analisi di questa tipologia di immagini è di fordamentale importanza in quanto:

- tutte le immagini generate da radiazioni elettromagnetiche diverse dallo spettro visibile (X-Ray, infrarosso, microonde, ultravioletto) e altri tipi di sorgenti (fasci di elettroni, suono) sono in scala di grigi, e ricoprono ruoli fondamentali in medicina, ricerca, industria;
- l'occhio umano è in grado di distinguere in media 16 livelli di grigio contemporaneamente mentre è in grado di percepire milioni di colori diversi.

L'elaborazione delle immagini in scala di grigio consiste in:

- evidenziare dettagli;
- ridurre il rumore;
- generare immagini a falsi colori.

Il programma sviluppato mira ad implementare tutte le principali tipologie di elaborazione di immagini in scala di grigi.

L'utente del programma è in grado di:

- effettuare l'upload di immagini in vari formati;
- convertire immagini RGB in scala di grigi;
- applicare all'immagine ciascuna delle operazioni elencate in Tabella 1;
- visualizzare le immagini di input e output con le relative statistiche e gli istogrammi di intensità;
- salvare le immagini elaborate nel formato preferito;
- impostare l'output dell'elaborazione come nuovo input per elaborazioni composte.

Spatial Operations	Intensity Transformation	Spatial Filtering	Mathematical Morphology	False Colour
o Rotate o Mirror o Resize* - Bilinear Interpolation - Bicubic Interpolation	o Negative o Contrast Stretching o Logarithmic o Histogram Equalization o Thresholding* o Slicing* o Gamma Correction* o Piece-wise (Costum)* - Linear - Steps	o Smoothing Filter - Average* - Gaussian* - Median* o Sharpening Filter - Laplacian 1 - Laplacian 2 o High-Boost Filter - Laplacian 1 - Laplacian 2 o Embossing - Embossing 1 - Embossing 2 - Embossing Intense o Edge Detection - Sobel - Prewitt	o Erosion* o Dilation* o Opening* o Closing*	o Standard palettes - palette1 - palette2 - palette3 - palette4 - palette5 o Costum palette*

Tabella 1

2. Studio del problema

1) Scelta del filtro

Ciascuna operazione elabora la Bitmap dell'immagine di input e resituisce una Bitmap in output. Tutti i filtri implementano dunque un' interfaccia IFilter.

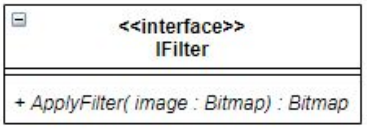

Diagramma 1

a) Operazioni Spaziali

Queste operazioni sono già implementate da metodi della classe Image. I filtri spaziali sono dunque classi che implementano l'interfaccia IFilter, il cui metodo ApplyFilter richiama il corrispondente metodo della classe Image con gli opportuni parametri.

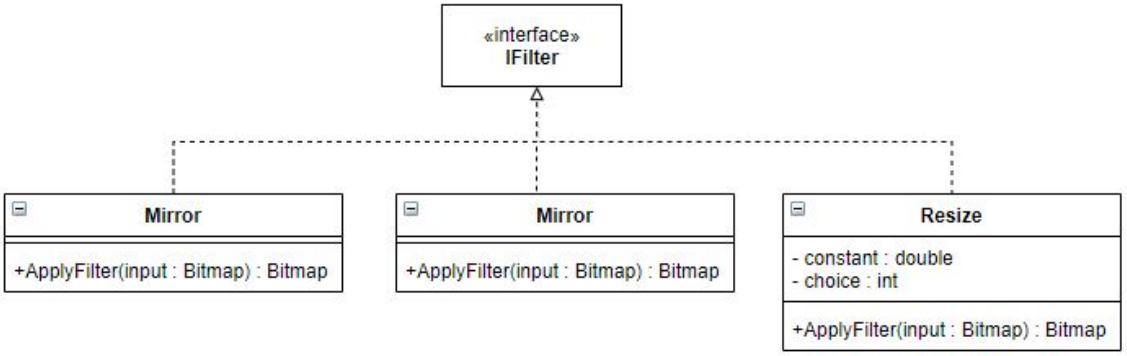

Diagramma 2

b) Trasformazioni di intensità

Queste operazioni consistono nel rimappare i valori di intensità dell'immagine di input utilizzando una "look-up table", che a ciascuno dei valori di intensità in scala di grigio nell'immagine di input fa corrispondere un valore di intensità nell'immagine di output. Le intensità possono assumere valori tra 0 e 255 e sono dunque rappresentabili con il tipo byte.

La classe LookupTable è dotata di un array di 256 valori di tipo byte che implementa la look- up table, e dei metodi ApplyFilter, che effettua il remapping delle intensità utilizzando la look-up table, e DrawGraph, che ne traccia il grafico. Classi derivate implementano look - up table specifiche che vengono generate dal costruttore della classe.

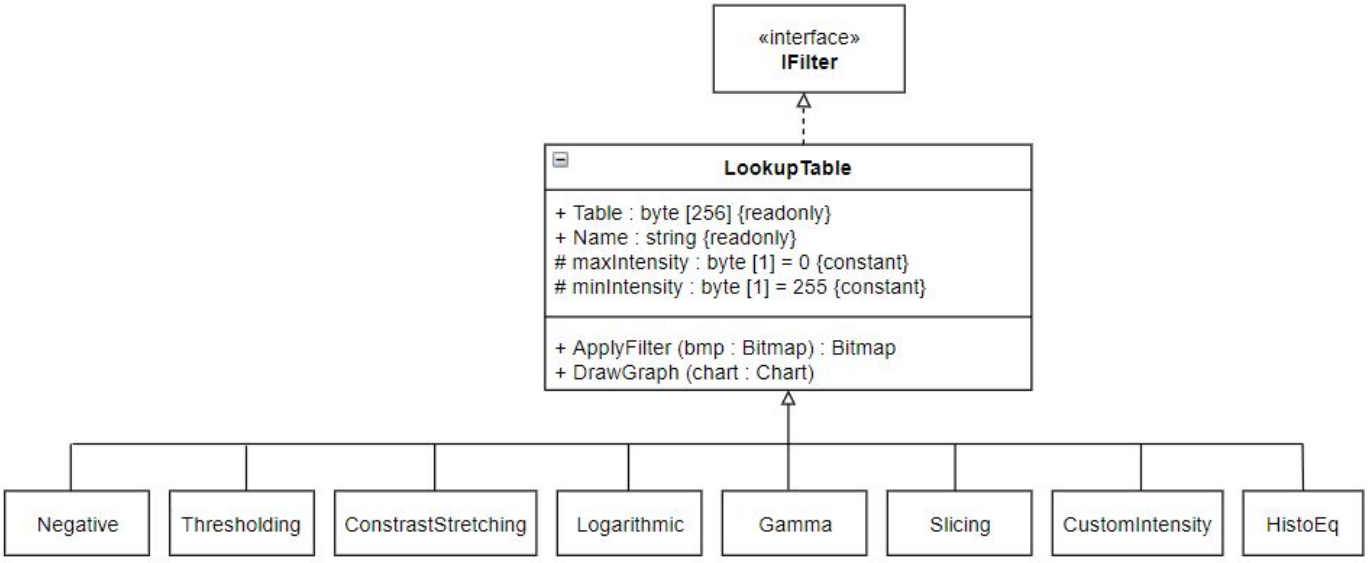

Diagramma 3

c) Filtraggio spaziale

Queste operazioni consistono nell'effettuare una convoluzione tra due matrici: la matrice dei pixel dell'immagine e una maschera o kernel. Il kernel è sovrapposto all'immagine in modo tale che il centro del kernel sia in corrispondenza del pixel che si vuole elaborare e il corrispondente pixel nell'immagine di output è calcolato come somma pesata dei prodotti di ciascun elemento del kernel con il corrispondente pixel dell'immagine sottostante.

Si distinguono due categorie di filtri:

- Filtri semplici, in cui l'immagine è convoluta una sola volta con un singolo kernel;
  esempi sono i filtri di media, gaussiani e laplaciani.
- Filtri composti, in cui l'immagine è convoluta con più kernel;
  esempi sono i filtri per l' "edge-detection", dove l'immagine è convoluta prima con un filtro semplice (ad esempio Sobel o Prewitt) poi con il filtro simmetrico al primo, e l'output è calcolato come il risultato della funzione distanza applicata all'output di ciascuna convoluzione.

La classe ConvFilter include gli attributi Kernel, Offset, Weight che caratterizzano la maschera di convoluzione e implementa il metodo ApplyFilter che effettua l'operazione di convoluzione. Classi derivate di ConvFilter implementano filtri specifici.

La classe CompositeFilter contiene due ConvFilters come "member objects".

La classe ausiliaria ConvMath implementa metodi statici utilizzati da altre classi per generare il proprio Kernel.

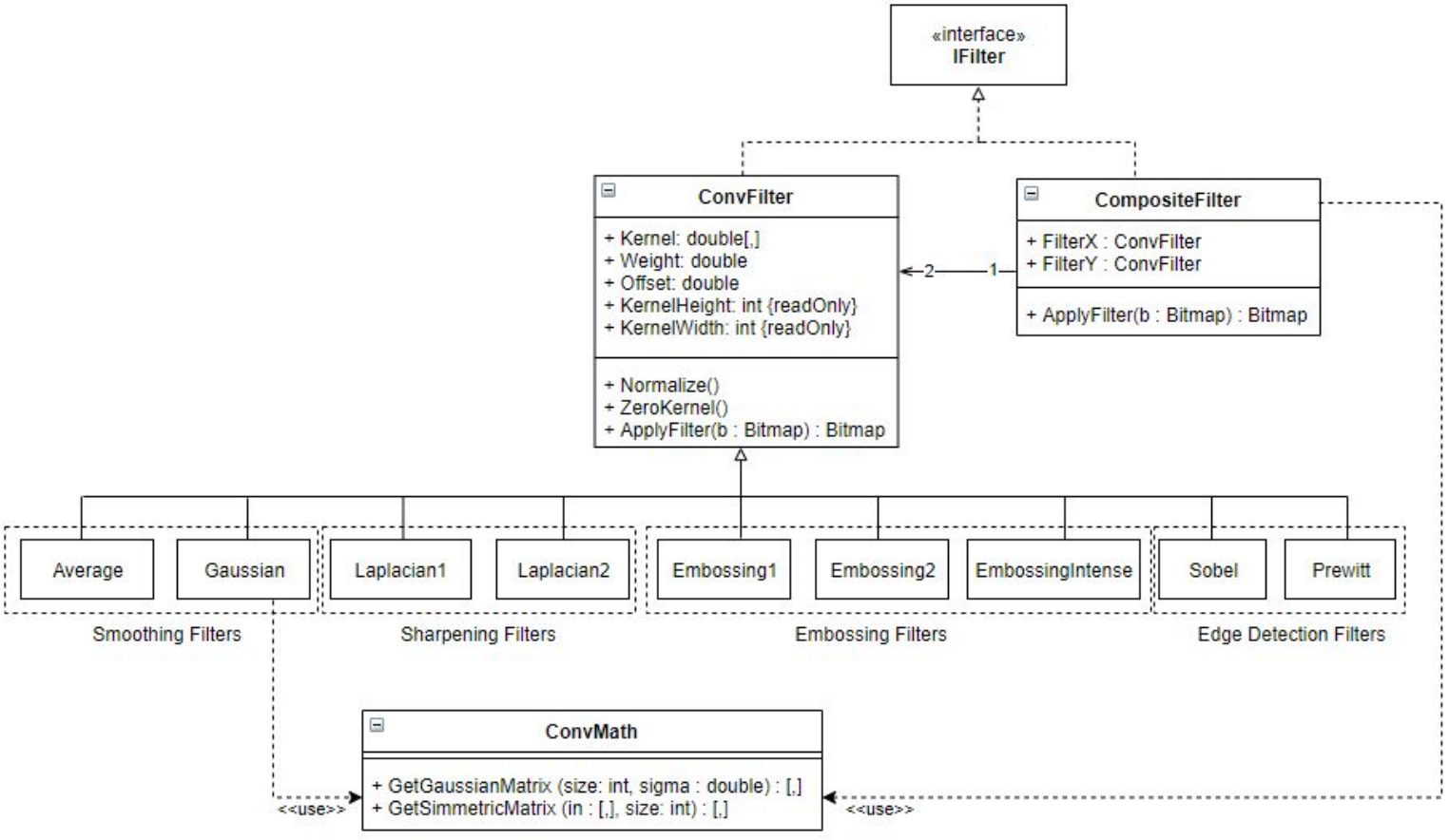

Diagramma 4

d) Morfologia matematica:

Applicare filtri mediani di smoothing o filtri di morfologia matematica consiste nel definire una maschera di una determinata dimensione e applicare statistiche d'ordine (come ad esempio valore massimo, minimo, mediano) ai pixel sottostanti alla maschera.

Anche in questo caso si distinguono :

- Filtri semplici, in cui si applica una statistica d'ordine una sola volta all'immagine.
  esempi sono il filtro mediano (valore mediano), il filtro di erosione (valore minimo), il filtro di dilatazione (valore massimo).
- Filtri composti, in cui si applicano n statistiche d'ordine in cascata;
  esempi sono il filtro di chiusura (filtro di erosione + filtro di dilatazione) e il filtro di apertura (filtro di dilatazione + filtro di erosione)

Poichè i filtri possono essere semplici o composti, e quest'ultimi possono essere costituiti a loro volta da filtri semplici e composti, si utilizza il pattern Composite per implementare le classi.

L'interfaccia IOrderStat deriva l'interfaccia IFilter e fornisce i metodi per accedere ai componenti dei filtro.

La classe astratta SimpleOrderStatsFilter implementa l'interfaccia e include gli attributi KernelSize, che definisce la dimensione della maschera, e il metodo astratto SelectPixel, che esprime il criterio secondo cui il pixel tra quelli sovrapposti dalla maschera sarà scelto come pixel nell'immagine di output.Le sue classi derivate concretizzano il metodo.

La classe CompositeOrderStatsFilter implementa anch'essa l'interfaccia IOrderStats e implementa i metodi per gestire filtri compositi. Le sue classi derivate definiscono filtri compositi specifici.

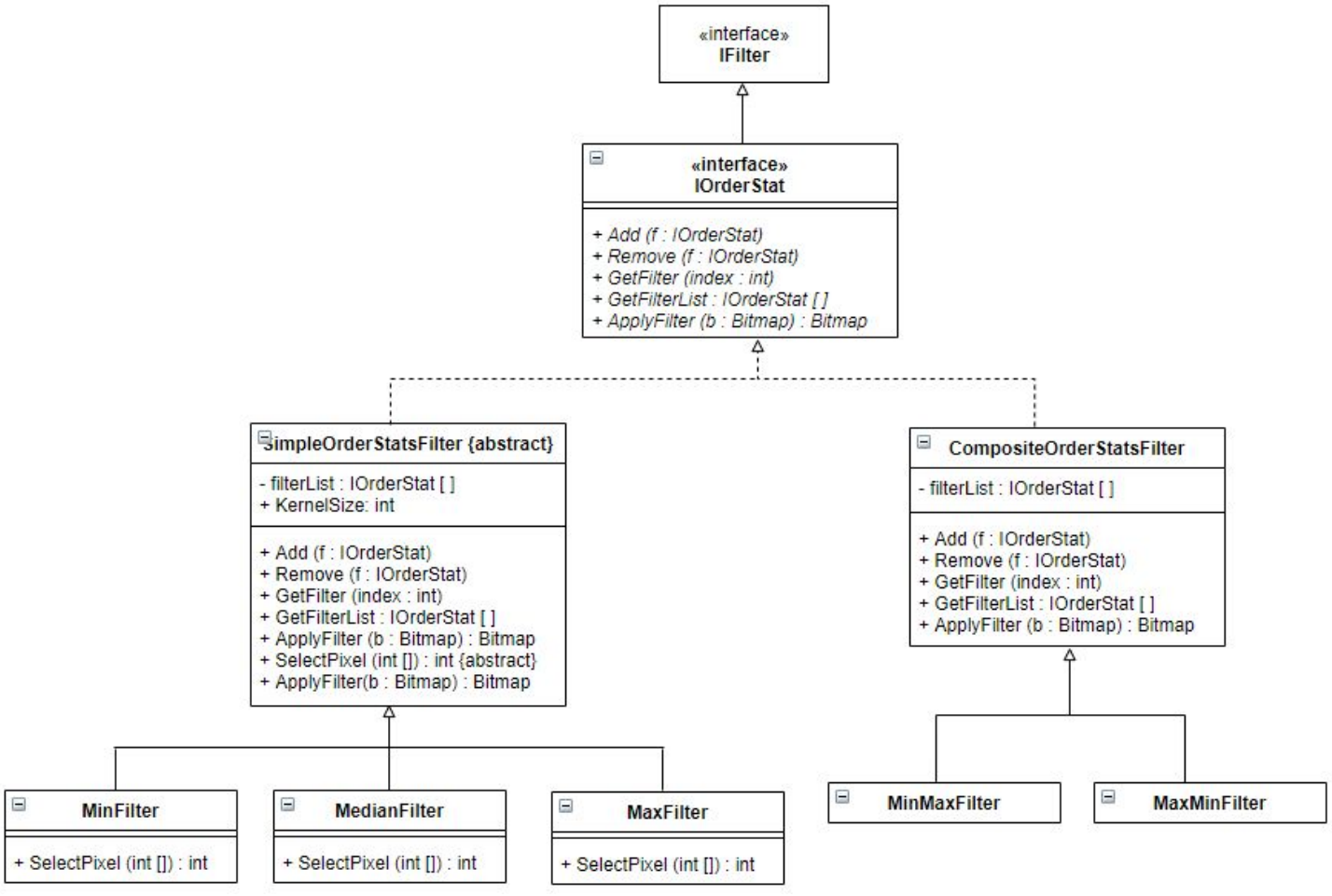

Diagramma 5

e) Falsi Colori

Similmente alle trasformazione di intensità, la conversione di immagini da scala di grigi a falsi colori consiste in un'operazione di remapping utilizzando una look-up table che a ciascuno dei valori di intensità in scala di grigio nell'immagine di input fa corrispondere un colore RGB nell'immagine di output.

La classe Palette è dotata di un array di 256 colori RGB che implementa tale look-up table. Poichè generalmente si produce una palette a partire da un numero limitato di colori, la classe è dotata di un attributo privato di tipo RemapPoints che memorizza una lista di colori e di un metodoche genera la palette a partire tali colori tramite interpolazione lineare.

La classe RemapPoints è dunque member object per la classe Palette. Le sue classi derivate definiscono palette predefinite o personalizzate.

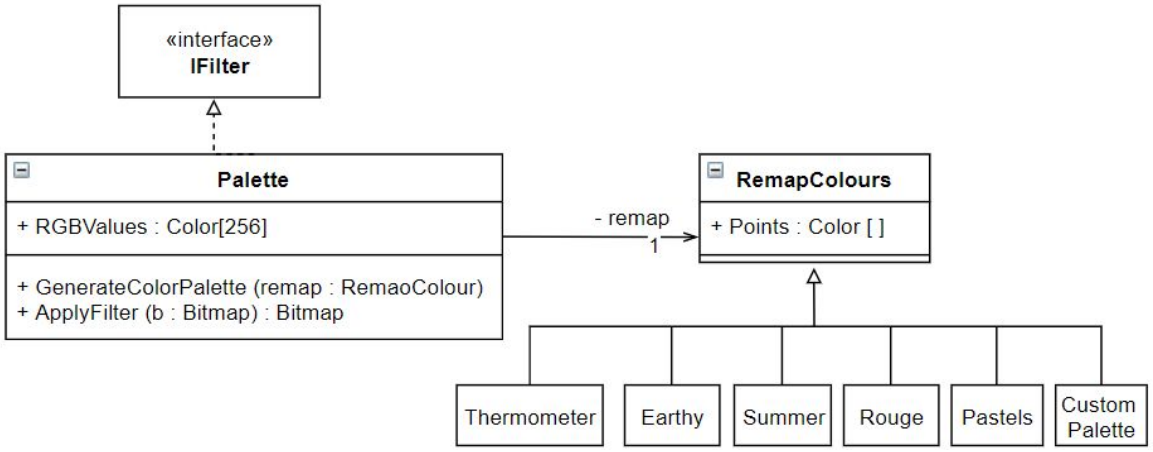

Diagramma 6

2) Visualizzare Immagini e Statistiche

Molte operazioni richiedono informazioni specifiche relative all'immagine (valori di intensità massima e minima, istogramma dei valori di intensità). É dunque necessario che tali informazioni siano sempre disponibili e visualizzabili.

La classe ImageStats calcola le statistiche relative a una Bitmap, ed è member object della classe ImageFile.

La classe ImageFile contiene Bitmap, nome e tutte le statistiche relative all'immagine e ha costruttori per generare l'oggetto a partire da file o da Bitmap.

La classe BitmapExtensions raccoglie metodi estensionali che si applicano alla classe Bitmap.

La classe DisplayManager si occupa della visualizzazione di immagine e statistiche e relativi grafici sul Form.

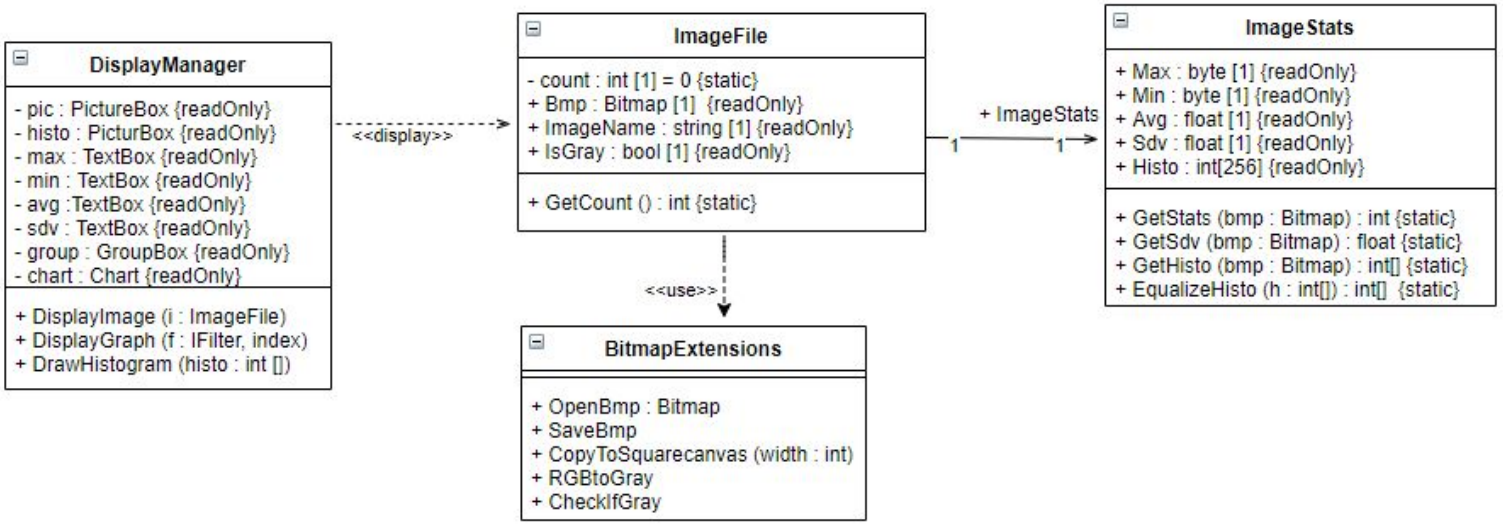

Diagramma 6

3) Generare il filtro selezionato

Nel selezionare l'operazione da eseguire, l'utente effettua due scelte: il tipo di operazione e il tipo di filtro. Si definisce dunque un nuovo tipo enum che racchiude i tipi di operazioni possibili.

```
public enum Transformation { spatial, intensity, convolution, morphology, colour };
```

La selezione da parte dell'utente aggiorna due indici, uno di tipo Trasformation e uno di tipo int.

Poichè molti dei filtri disponibili dipendono da parametri, solo in seguito alla scelta dell'utente e del parsing dei parametri dall'utente è possibile istanziare il filtro corretto. É dunque preferibile delegare l'istanziazione del tipo specifico di filtro ad una classe apposita tramite Factory method.

Si crea una classe di tipo Creator per ciscuno dei filtri che implementano l'interfaccia IFilter. Queste classi implementano l'interfaccia IFilterCreator e che crea un'istanza del filtro evocando il metodo FactoryMethod. Poichè alcuni filtri necessitano di parametri dall'utente e di statistiche sull'immagine, i parametri del metodo sono di tipo ImageFile e ParseManager, che si occupa del parsing di tutti i parametri, come verrà discusso in seguito.

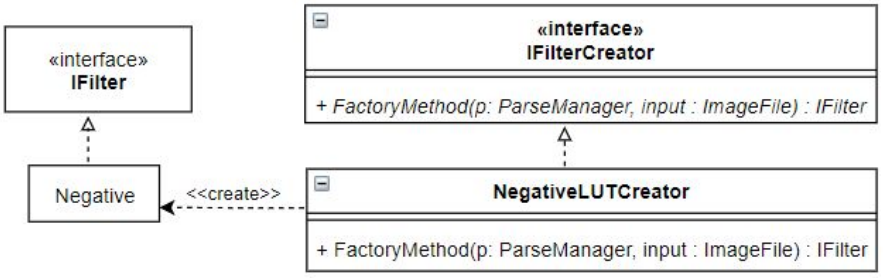

Diagramma 7

Si crea una struttura che raccoglie tutte le classi che implementano l'interfaccia IFilterCreator. Si sceglie di utilizzare un dizionario di liste che a ciascuna trasformazione associa la lista di tutti i creatori per i filtri di quel tipo. Per creare un'istanza del filtro giusto basta dunque trovare nel dizionario il creator giusto tramite i due indici ed evocare il metodo FactoryMethod in quel creator per istanziare il filtro desiderato.

```
IFilter filter = filterCreator[transformationIndex][selectionIndex].FactoryMethod(parseParameters, inputImage);
```

Data la complessità della struttura, si delega la sua creazione ad un Builder che è in grado di generare ciascuna lista individualmente ed il dizionario completo, e produce il prodotto desiderato a seguito degli ordini dati da un director.

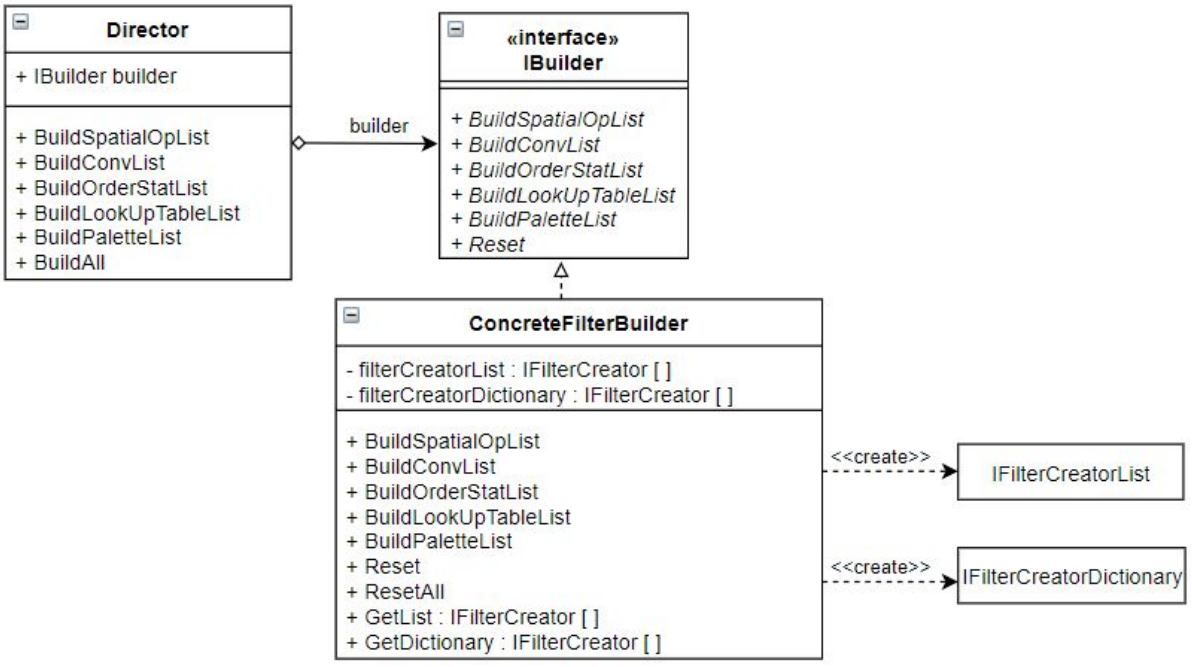

Diagramma 8

4) Gestire parametri in input dall'utente

Le operazioni indicate con * in Tabella 1 richiedono l'input di parametri da parte dell'utente. Si individuano le seguenti classi di parametri e i rispettivi valori attesi:

- Valore di intensità : intero compreso tra 0 e 255
- Range di intensità : 2 interi compresi tra 0 e 255, di cui il primo è minore del secondo
- Dimensione del Kernel : intero dispari compreso tra 3 e 15
- Costante reale : valore reale compreso tra 0 e 20
- Lista di punti : lista contenente almeno due elementi

Si sviluppano classi parser ad hoc per ciascun tipo di parametro. Tutte le classi realizzano l'interfaccia IParse e implementano i metodi Parse, che restituisce un valore boleano relativo al successo del parsing, e GetError, che restituisce il messaggio di errore corrispondente.

La classe astratta IParseDataTable gestisce il parsing di liste di valori (utilizzati per creare filtri personalizzati) e la loro visualizzazione in una DataGridView, senza specificare il tipo di dato contenuto nella lista. La classi derivate implementano il parsing di liste di tipi specifici:

- ParseIntensityPoints esegue il parsing di liste di valori di tipo Point (X,Y), dove X e Y sono i valori di intensità in input e output, a partire dalle quali si generano look-up table per trasformazioni di intensità personalizzate
- ParseColourPoints esegue il parsing di liste di valori di tipo colour (R,G,B) da cui si ricavano palette personalizzate

La classe ParseManager gestisce il parsing di tutti i parametri, fungendo da centro unico per l'acquisizione di parametri da altre funzioni.

La classe Visitor implementa una intefaccia IVisitor, è member object della classe ParseManager e collabora con esso alla gestione del parsing. Ha un attributo che indica se il parsing dell'ultimo parametro è andato a buon fine.
Ogni qual volta un parametro è richiesto dalla funzione FactoryMethod del filter creator selezioato, il Visit richiama il metodo Parse nel parser target. Se il parsing ha successo si può procedere con la creazione del filtro, altrimenti si mostra all'utente il relativo messaggio di errore.

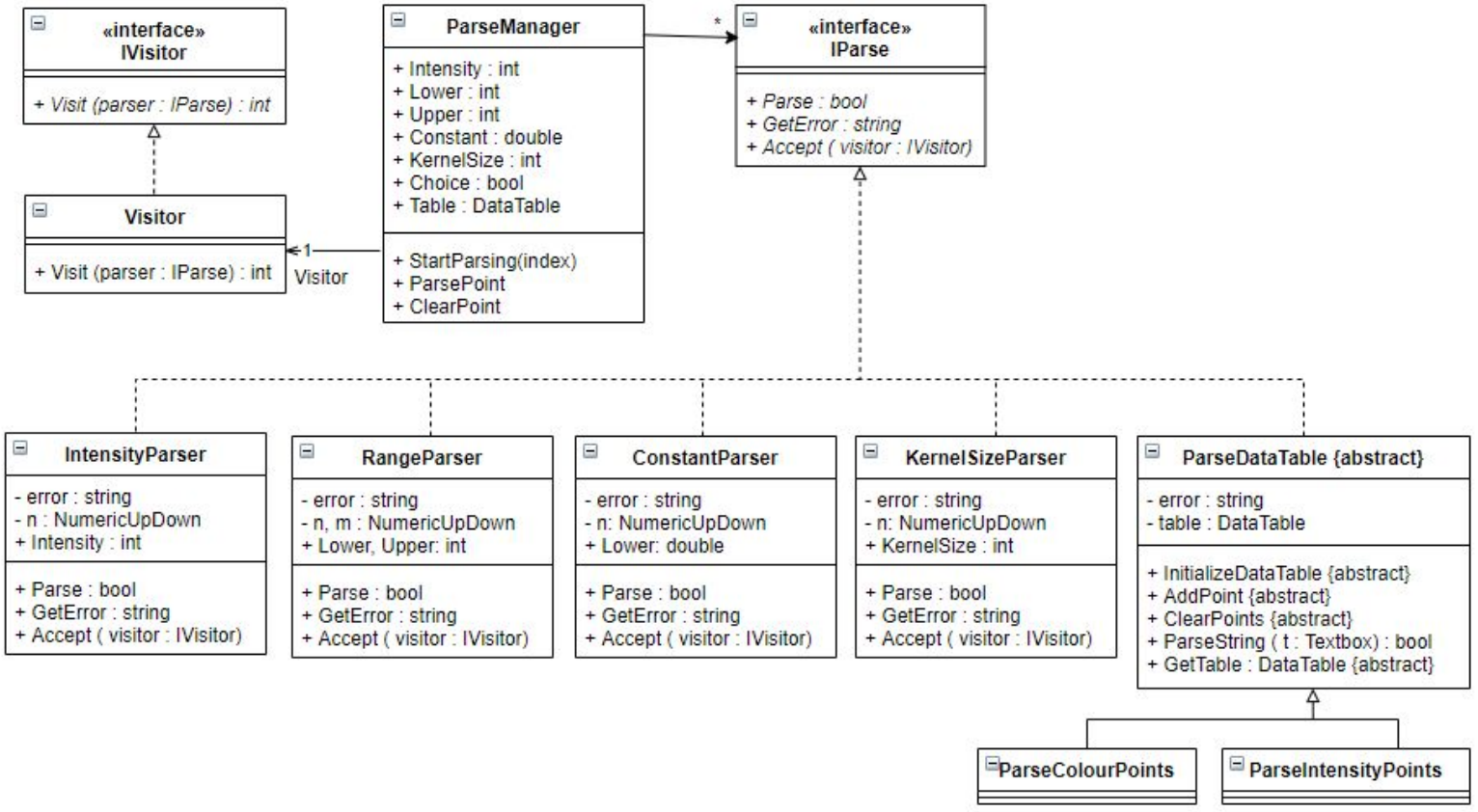

Diagramma 9

Dati in formato DataTable vengono convertiti in oggetti di tipo LookupTable e RemapColours tramite il pattern Adaptor per replicare l'interfaccia degli altri filtri

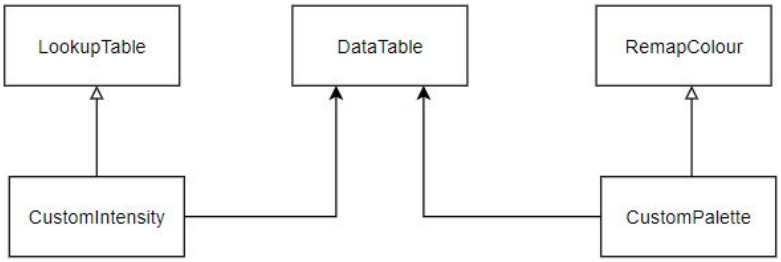

Diagramma 10

5) Menu personalizzato

Si è scelto di creare un menu personalizzato invece di adoperare Menu e Toolbars già presenti nel ToolBox di WindowsForm per avere maggior flessibilità anche da un punto di vista estetico.

I menu principali (verticale) e secondari (orizzontale) sono pannelli di bottoni, la cui visibilità è controllata da un timer.

La classe astratta MenuManager contiene il metodo astratto Open che viene concretizzato dalle due classi derivate PrimaryMenu e SecondaryMenu.

Tali classi possono essere utilizzate in ciascuna applicazione Windows Form per realizzare un menu personalizzato.

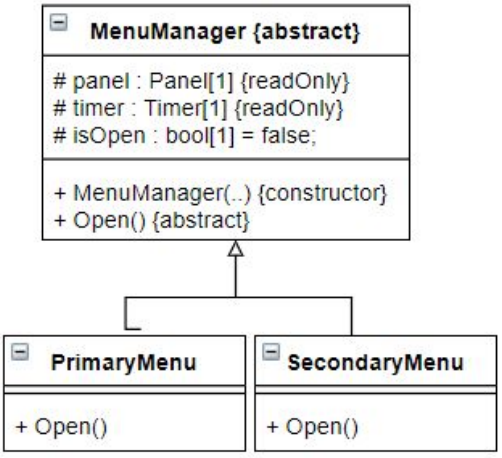

Diagramma 11

3. Scelte architetturali.

1 Architettura software

Il Form contiene quattro elementi principali che gestiscono tutti gli aspetti dei programma:

- MenuManager - gestione dei menù di scelta dell'utente
- DisplayManager - visualizzazione di immagini, grafici e statistiche
- ParseManager - parsing dei parametri dall'utente
- Director & ConcreteBuilder - creazione del dizionario dei FilterCreators

La scelta dell'utente determina l'aggiornamento degli indici, che individuano il corretto creator nel dizionario.
Il filtro corrispondente viene creato a run-time e utilizzato per generare l'immagine di output.

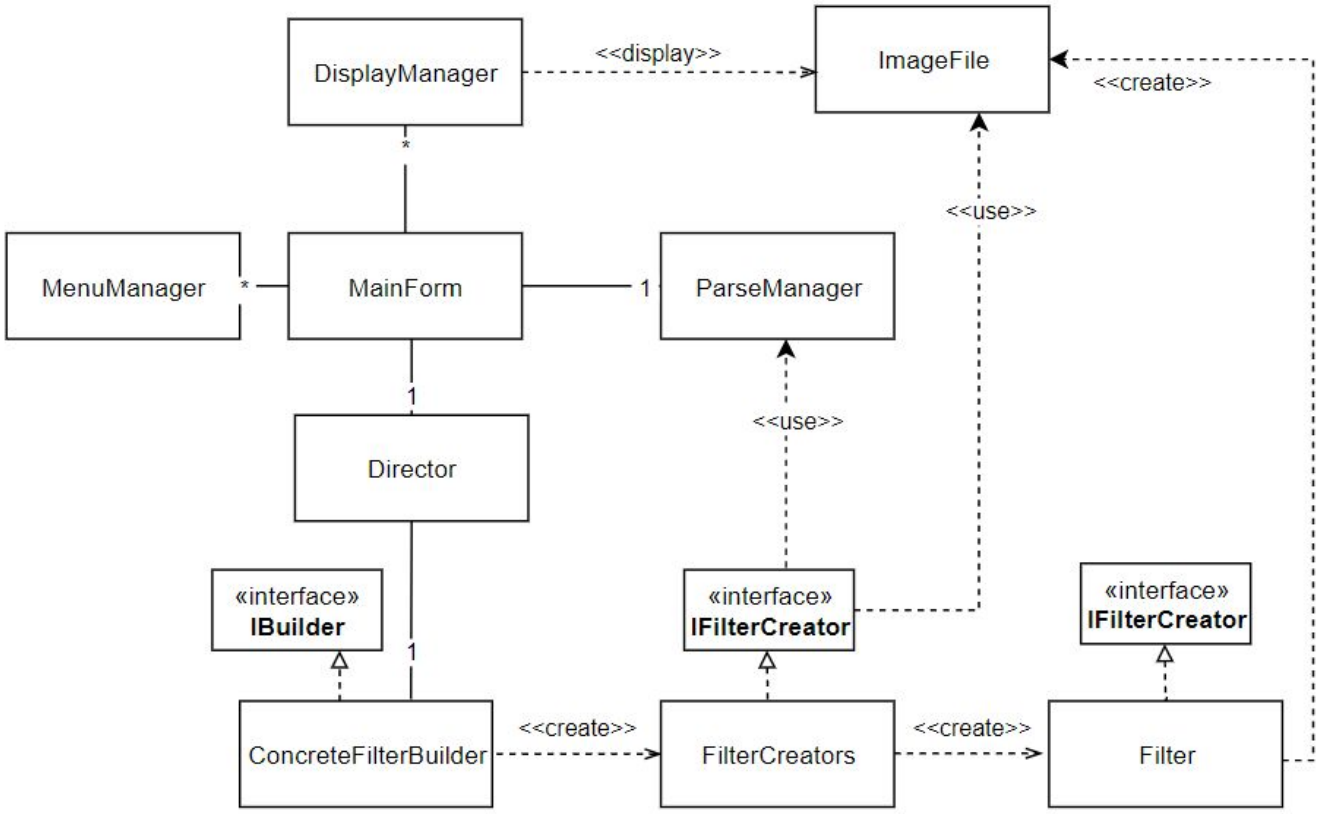

Diagramma 12

2 Design Pattern

a) Creational Patterns

L'utilizzo dei pattern Factory Method e Builder è descritto nella sezione 2.3 con i relativi diagrammi delle classi (Diagramma 7 e Diagramma 8).

- **Factory Method** consente di delegare l'istanziazione di un oggetto di tipo specifico ad una classe preposta a questo scopo e permette di scrivere codice altamente polimorfico.
- **Builder** consente di separare la costruzione di una struttura altamente complessa, come il dizionario di liste di creators, dal suo utilizzo e visualizzazione.

b) Structural Methods

L'utilizzo del pattern Composite è descritto nella sezione 2.1 a) con relativo schema delle classi (Diagramma 4).

- **Composite** consente di creare filtri singoli o composti, formati a loro volta da filtri semplici o composti, ma con la quale ci si possa interfacciare indifferentemente dal loro tipo.

L'utilizzo del pattern Adaptor è descritto nella sezione 2.4 con relativo schema delle classi (Diagramma 10).

- **Adaptor** ci consente di convertire il risultato del parsing di una lista di valori memorizzati in Data Table in altri oggetti che implementano l'interfaccia desiderata, come LookupTable e Remap Colour.

c) Behavioural Patterns

L'utilizzo del pattern Visitor è descritto nella sezione 2.4 con relativo schema delle classi in (Diagramma 9).

- **Visitor** consente di aggiungere funzionalità alla struttura che esso visita, come ad esempio la verifica delle condizioni di parsing e la stampa dei corrispondenti avvisi di errore.

4. Use Cases

Use Case : Acquisizione Immagine RGB
Id : UC1
Actor : Utente
Preconditions: il programma è stato avviato
Basic course of events: 1. L'utente clicca il pulsante Upload Image 2. L'utente seleziona un'immagine da una cartella del proprio device 3. Il programma acquisisce l'immagine e identifica il tipo di immagine come RGB 4. Un messaggio di attenzione chiede all'utente se vuole convertire l'immagine da RGB a grayscale 5. L'utente seleziona OK 6. Il programma converte l'immagine in grayscale 7. Il programma mostra l'immagine, le statistiche e il grafico dell'istogramma
Postconditions: Le immagini di input e output con relative statistiche e grafici sono mostrate all'utente
Alternative paths: Se in 5 l'utente seleziona Cancel, si torna alla schermata precedente e l'utente può selezionare un'altra immagine

Use Case : Trasformazioni semplici [senza parametri]
Id : UC2
Actor : Utente
Preconditions: Immagine in grayscale acquisita con successo
Basic course of events: 1. Il programma mostra l'immagine, le statistiche e il grafico dell'istogramma 2. Il menù principale diventa visibile 3. L'utente seleziona il tipo di trasformazione da eseguire 4. Il menù secondario per la trasformazione scelta diventa visibile 5. L'utente seleziona il filtro 6. Il programma genera il filtro e lo applica all'immagine. Se ha successo, mostra l'immagine, le statistiche e il grafico dell'istogramma
Postconditions: Le immagini di input e output con relative statistiche e grafici sono mostrate all'utente
Alternative paths: Se 6 fallisce, un messaggio di errore compare e si torna al menu principale

Use Case : Trasformazioni con parametri
Id : UC3
Actor : Utente
Preconditions: L'utente ha caricato l'immagine e selezionato il filtro
Basic course of events: 1. Il pannello di scelta parametri diventa visibile 2. L'utente seleziona i parametri richiesti o accetta i valori di default 3. L'utente clicca il tasto Apply 4. Il pannello di scelta scompare 5. Il programma genera il filtro e lo applica all'immagine. Se ha successo, mostra l'immagine, le statistiche e il grafico dell'istogramma 6. Il pannello con le opzioni Save e Set as Original compare 7. L'utente clicca il tasto Save 8. L'utente digita il nome, seleziona il formato desiderato e clicca OK 9. Il programma tenta di salvare l'immagine. Se ha successo, genera l'immagine nella cartella prestabilita
Postconditions: L'immagine trasformata è stata salvata nel dispositivo
Alternative paths: Se 9 fallisce, un messaggio di errore compare e all'utente è richiesto di ritentare o tornare all'applicazione

Use Case : Trasformazioni personalizzate
Id : UC4
Actor : Utente
Preconditions: L'utente ha caricato l'immagine e selezionato il filtro personalizzato
Basic course of events: 1. Il pannello di scelta parametri diventa visibile 2. L'utente digita i valori di input e clicca il tasto Add 3. I valori sono aggiunti a una tabella visibile all'utente 4. L'utente clicca il tasto Clear 5. Tutte le righe intabella vengono cancellate 6. L'utente aggiunge nuovi valori alla tabella 7. L'utente clicca il tasto Apply 8. Il programma genera il filtro e lo applica all'immagine. Se ha successo, mostra l'immagine, le statistiche e il grafico dell'istogramma
Postconditions: L'output con relative statistiche e grafici è mostrato all'utente
Alternative paths: 9. Se i valori digitati non sono nel formato atteso, un messaggio di attenzione compare, informando l'utente sul formato da utilizzare 10. Se i valori digitati fuoriescono dal range, vengono modificati automaticamente per rientrare nel range 11. Se l'utente clicca il tasto Apply avendo inserito meno di 2 valori, compare un messaggio di attenzione che lo invita ad aggiungere altri valori.