IMAGine

Francesco Paolo Castiglione, Davide Iraci, Andrea Montemaggiore

Settembre 2020

Indice

1 Introduzione

IMAGine (IMAge-enGINE) è un linguaggio ideato per l'elaborazione delle immagini a trecentosessanta gradi. Il linguaggio consente le elaborazioni più comuni attraverso un processo immediato ed intuitivo. E' stato pensato e sviluppato tenendo conto di alcuni fattori chiavi quali semplicità d'uso e rapidità di apprendimento, basandosi anche su altri linguaggi già esistenti quali MATLAB, Python o software per la modifica di immagini.
Il resto della relazione è strutturato come segue: la sezione 2 illustra il contesto in cui si colloca il linguaggio proposto, nella sezione 3 vengono analizzati i dettagli implementativi, nella sezione 4 si illustrano le caratteristiche di IMAGine e degli esempi di applicazioni, infine nella sezione 5 vengono riepilogati i punti principali del lavoro svolto.

2 Stato dell'arte

Attualmente l'Image Processing avviene principalmente per mezzo di software già pronti quali Photoshop, Gimp, etc. Questi software sono stati pensati per un utilizzo da parte di amanti delle immagini e per professionisti in campo fotografico. Alcuni di questi, come il già citato Adobe Photoshop vengono proposti in una suite software chiamata "Adobe Creative 2020", la quale, nella sua ultima edizione, comprende altri applicativi per l'Image Processing di livello sempre più alto. Per lo sviluppo del progetto ci siamo principalmente basati sullo stato dell'arte dei linguaggi che permettono operazioni su immagini e sulle operazioni che permettono: da semplici rotazioni, zooming e shrinking alle più complicate operazioni di edge detection, cambio dello spazio di colori, etc;
Come linguaggio da adoperare per produrre il progetto sono stati individuati linguaggi dalla sintassi concisa ed efficace quali C, C++, Java e Python. Ognuno di detti linguaggi ha pro e contro ma tutti richiedono l'utilizzo di librerie esterne per l'implementazione delle modifiche e operazioni su immagini. C e Python sono attualmente i più veloci. C presenta una sintassi più complessa per l'utilizzo delle relative librerie mentre Python possiede numerose librerie con una sintassi di facile utilizzo ma alcune di esse, o meglio quasi tutte, non ottengono un buon risultato performando alcune operazioni più complicate come edge detection. Va sottolineato come il package "skimage" per Python sia uno dei migliori in termini di prestazioni e risultati. Per C e C++ la libreria migliore in termini di prestazioni è "LibVips" adoperata anche in questo progetto, nonostante possieda una sintassi di difficile utilizzo per via dei lunghi nomi dei metodi. L'ultima analisi si basa su Matlab, ampiamente usato, specialmente in ambito accademico. Si tratta del linguaggio per immagini per eccellenza. Alcune operazioni, come media adattiva, mediano adattivo, etc, richiedono che il programmatore abbia una conoscenza approfondita del linguaggio ed ambiente di sviluppo MATLAB, mentre per operazioni più semplici, quale applicazione di filtri di media e mediano, è sufficiente utilizzare funzioni built-in. Dallo stato dell'arte vengono dunque individuati i passaggi chiave per la creazione del linguaggio: sintassi semplici e prestazioni d'alto livello.

3 Descrizione del progetto

In questa sezione vengono introdotti i requisiti individuati dal team e le motivazioni che hanno portato alla scelta della libreria LibVips.

3.1 Scelte progettuali

Dal punto di vista sintattico il team si è ispirato a linguaggi intuitivi quali Javascript, Python e C#. Le variabili vengono dichiarate senza specificare un tipo esplicito. Il valore viene semplicemente assegnato alla variabile in fase di inizializzazione della stessa o successivamente. Le immagini vengono dichiarate fornendo il path, rendendo cosi' più manegevole l'elaborazione delle immagini. Il tipo ricorsivo lista può contenere qualunque elemento, compreso un tipo lista.

3.2 LibVips

3.3 Guida all'uso

4 Caratteristiche del linguaggio

4.1 Grammatica

4.1.1 Tipi

In seguito ad analisi relative ai possibili usi del linguaggio proposto, sono stati individuati i seguenti tipi primitivi:

- `integer`, utilizzato per rappresentare i numeri interi
- `doublePrecision`, utilizzato per rappresentare i numeri in virgola mobile
- `str`, utilizzato per rappresentare le stringhe di caratteri
- `img`, utilizzato per rappresentare le immagini

E' stato inoltre individuato il tipo composto:

- `list`, tipo ricorsivo

Le variabili di tipo *int*, *doublePrecision* e *str* vengono dichiarate senza specificare un tipo esplicito.

Esempio di dichiarazione ed assegnazione:

```
stringVariable="this is a string";
intVariable=5;
doubleVariable=5.5;
img imageVariable="/path/img.png";
list li={1,2,"element"};
```

V1\V2	int	double	string
int	+ - * / or and CMP	+ - * / or and CMP	*
double	+ - * / or and CMP	+ - / * CMP	+
string	+ *	+	+ CMP

Tabella 1: Operazioni consentite dal linguaggio, sono da intendersi come V1 <operatore> V2. Con CMP si indicano le operazioni di comparazione

4.1.2 Operazioni

Il linguaggio consente le seguenti operazioni : somma, sottrazione, moltiplicazione, divisione, and e or logici, e paragoni (==, !=, >, >=, <, <=).

- Il risultato di divisioni, prodotti, somme e differenze fra *integer* e *doublePrecision* è sempre un *doublePrecision*.
- Il risultato di divisioni fra *integer* è un *doublePrecision*
- Il risultato di somme, prodotti e differenze fra tipi uguali mantiene il tipo degli elementi iniziali

In generale, *print(var + string)* stamperà su schermo la concatenazione delle due variabili, indipendentemente dal tipo di *var* (che verrà convertita a stringa)

String1 == String2 valuta il contenuto delle stringhe
String1 != String2 valuta il contenuto delle stringhe
print(3*string) stamperà per tre volte il contenuto di string
print(3+string) stamperà la concatenazione fra la 3 e il contenuto di string
print(string1+string2) stamperà la concatenazione fra le due stringhe

Le combinazioni di tipi consentite per ogni operatore sono riassunte nella tabella 1. I paragoni e gli operatori logici restituiscono sempre degli int, tutti gli altri operatori restituiscono un valore del tipo della colonna, tranne la moltiplicazione `int * string` equivalente a `string * int`. Le operazioni illustrate non sono ammesse per il tipo immagine e per il tipo lista.

4.1.3 Strutture di controllo

Il linguaggio fornisce la possibilità di usare sia comandi condizionali che comandi iterativi. I comandi condizionali sono i seguenti:

- `if ( condition ) then { instructions };`
- `if ( condition ) then { instructions } else { instructions }`

I comandi iterativi sono qui elencati :

- `while ( condition ) do { instructions }`
- `foreach ( tempElement :  list ) { instructions }`

Nell'espressione `while` si itera indefinitivamente finché la condizione non è più verificata. Nle costrutto `foreach` si itera invece per ogni elemento contenuto nella lista. E' possibile effettuare operazioni sull'elemento della lista tempElement, anche nel caso in cui si tratti di un'immagine. Questo rende possibile l'elaborazione di una lista di immagini.

4.1.4 Funzioni

Il linguaggio presenta una ricca selezione di funzioni built-in. All'utente viene inoltre fornita la possibilità di definire la propria funzione custom, con sintassi:

"`def nome_funzione ( parametri_formali ) { istruzioni }`".