

UNIVERSITA DEGLI STUDI DI PISA

Relazione Finale Tirocinio
Scuola di Ingegneria
Dipartimento di Ingegneria dell'Informazione

Influenza del numero di lunghezze d'onda sulla
qualità dell'immagine spettrale.

Tirocinante: Cocomini Davide
Tutor universitario: Pistolesi Francesco, Lazzerini Beatrice
Tutor aziendale: Maccari Antonio, Maccari Francesca

Indice

1	Introduzione	3
1.1	Lo spettro visibile	3
1.2	Percezione del colore	4
1.3	Metamerismo	5
1.4	Scanner spettrofotometrico	6
2	Obiettivo del tirocinio	8
2.1	Confronto tra immagini	8
2.1.1	SSIM	9
2.1.2	Algoritmo	9
2.1.3	Componenti della formula	10
3	Svolgimento	11
3.1	Strumenti	11
3.1.1	Tavola Savoia-Navona-Matte	11
3.2	Rappresentazione delle immagini	13
3.3	Estrazione dell'indice	15
3.4	Esempio di esecuzione	16
4	Conclusioni	17

1 Introduzione

1.1 Lo spettro visibile

Lo “spettro del visibile” si trova nella parte centrale dello spettro ottico, il quale comprende anche i raggi infrarossi e quelli ultravioletti. E’ composto da quella parte di spettro elettromagnetico che include tutte le colorazioni percepite dall’occhio umano, partendo dal rosso fino ad arrivare al viola. La colorimetria utilizza la percentuale della luce incidente che è stata riflessa (nell’intervallo del visibile (400-700 nm) per descrivere il colore dell’oggetto. Applicazioni particolari quali la misurazione della fluorescenza, del bianco ed i colori mimetici prendono in considerazione anche le radiazioni UV (350-400nm) e NIR (700-1300nm). Ciascun oggetto colorato viene pertanto definito da una curva di riflettanza, similmente alle impronte digitali nell’uomo.

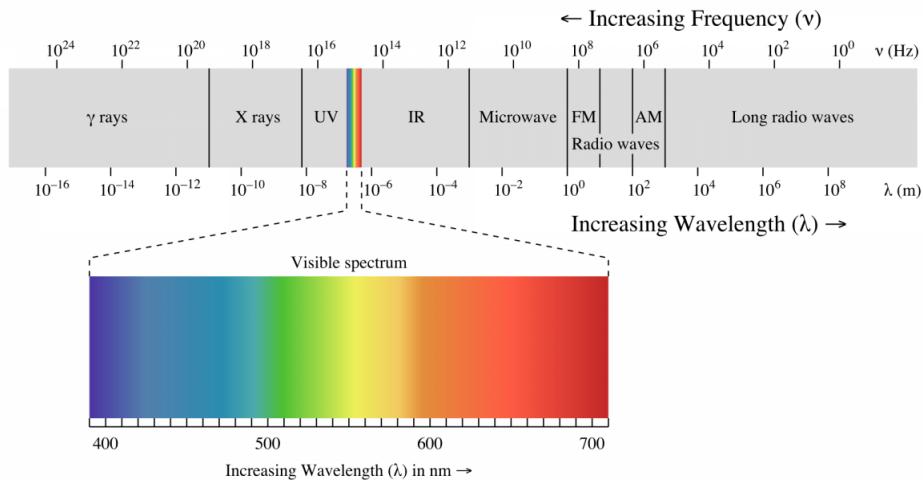


Figura 1: ”Lo spettro visibile”

1.2 Percezione del colore

Il colore nasce dalla luce. La luce che colpisce un oggetto viene parzialmente assorbita a seconda del colore. La parte non assorbita viene riflessa e trasmessa ai recettori cromatici all'interno dell'occhio umano. Questi ultimi trasformano la luce assorbita in impulsi che percorrono le vie nervose fino a raggiungere il cervello, dove vengono interpretati: nasce così un'impressione cromatica. Dal punto di vista prettamente biologico il colore si genera pertanto nell'occhio dell'osservatore e costituisce un'impressione sensoriale.

A proposito di impressione sensoriale: ciascun individuo "percepisce" il colore in modo differente. Tale fenomeno non è riconducibile solamente al fatto che non esistono mai due occhi uguali tra loro. Anche l'interpretazione del colore varia infatti da individuo ad individuo. Perfino la stessa persona può percepire differentemente il colore in momenti diversi ed in base allo stato d'animo. Il colore stesso può pertanto generare sensazioni differenti.

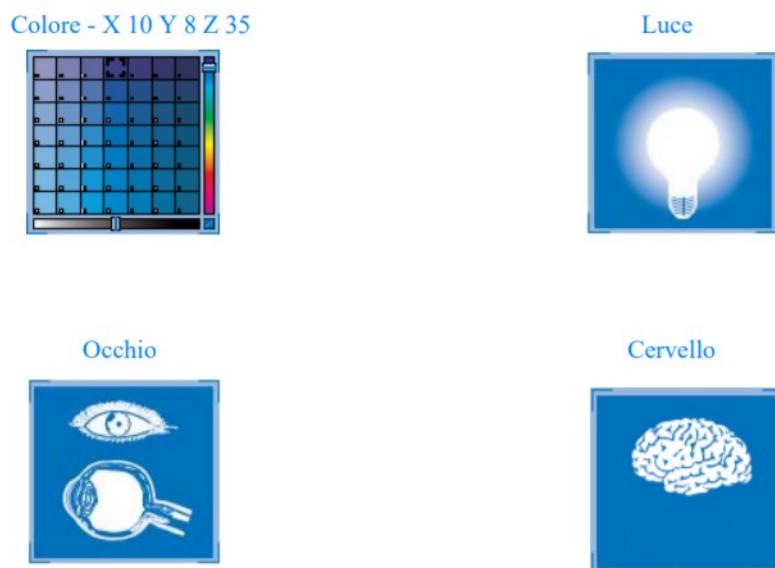


Figura 2: "Percezione del colore"

1.3 Metamerismo

Ciascuno di noi conosce il fenomeno per cui un oggetto colorato osservato sotto una determinata sorgente di luce come la luce diurna, presenta un colore diverso da quello visto sotto un'altra sorgente come ad esempio la luce di una lampadina ad incandescenza. Tale mutamento cromatico, caratteristico di quasi tutti gli oggetti colorati, viene spesso erroneamente definito metamerismo. Ma che cos'è in realtà il metamerismo? Il metamerismo può essere spiegato in termini di variazioni dei valori di tristimolo (X, Y e Z), che quantificano la percezione del colore. Due campioni sono uguali sotto una determinata sorgente di luce quando i relativi valori XYZ sono identici per quel particolare illuminante. Tale condizione è sempre soddisfatta sotto tutti gli illuminanti nel caso di campioni identici caratterizzati da curve di riflettanza uguali. I campioni metamerici presentano invece curve di riflettanza diverse. Ne consegue che tali campioni possono avere valori di tristimolo XYZ uguali sotto un certo illuminante ed apparire uguali, ma al mutare dell' illuminante, valori XYZ diversi ed apparire dunque diversi fra loro.

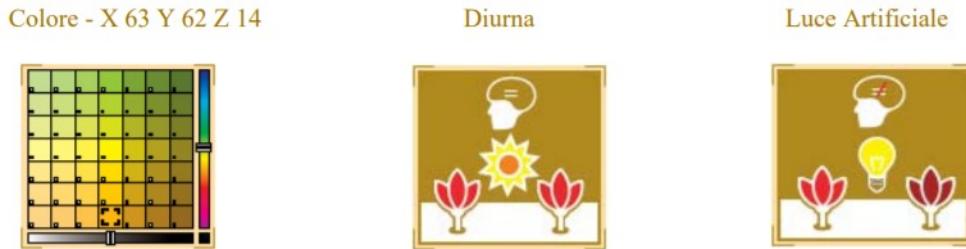


Figura 3: "Metamerismo"

1.4 Scanner spettrofotometrico

Lo scanner spettrofotometrico è nato per riprendere immagini adatte a costituire archivi di dipinti e di altre opere d'arte di superficie piana, utilizzabili per scopi di diagnostica, di restauro e di multimedialità. Le immagini d'archivio devono possedere almeno questi requisiti: riprodurre fedelmente il colore, essere confrontabili con immagini dello stesso oggetto riprese a distanza di tempo, essere stabili nel tempo per fornire una documentazione storica delle opere d'arte. Immagini di questo tipo si ottengono se sono basate sulla misurazione del fattore di riflessione spettrale della loro superficie. Esso è determinato dal rapporto tra il segnale dovuto alla luce riflessa dall'opera d'arte e quella riflessa da un riflettore ideale in identiche condizioni di misurazione ed è quindi indipendente dalle specifiche tecniche della strumentazione usata per la sua misurazione. Inoltre, consente di esprimere il colore nello spazio colorimetrico dell'Osservatore Standard della CIE che è equivalente allo spazio colorimetrico del sistema di visione dell'uomo. Le immagini basate sul fattore di riflessione spettrale costituiscono una nuova categoria di immagini digitali: le immagini spettrali. Le immagini spettrali hanno parecchi vantaggi rispetto alle immagini digitali tricromatiche perché consentono:

1. di sfruttare al meglio il sensore elettronico (numero pixel-immagine uguale numero pixel-sensore);
2. il controllo del metamerismo;
3. la codifica dei colori in ogni spazio colorimetrico;
4. il loro uso anche con i futuri sviluppi della colorimetria;
5. la riproduzione su base spettrale;
6. di scegliere l'illuminante e il tipo di Osservatore Standard CIE per il calcolo del colore.

Lo scanner iperspettrale ottiene informazioni relative alla luce riflessa dall'immagine ottenendo così una serie di immagini su più frequenze che unite compongono l'immagine vera e propria.

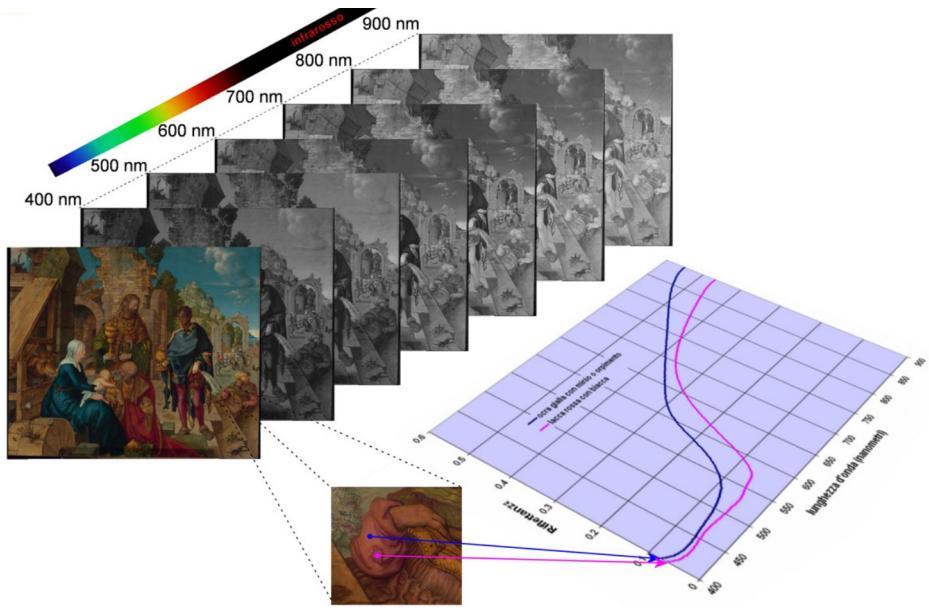


Figura 4: "Risoluzione"

2 Obiettivo del tirocinio

2.1 Confronto tra immagini

Il tirocinio ha l'obiettivo di ricercare una metodologia che misuri la differenza esistente tra due immagini aventi il medesimo soggetto, ma realizzate in tempi o con mezzi differenti affinché si possa capire qual è il minimo numero di bande che uno scanner deve ottenere continuando ad ottenere un risultato soddisfacente. Riducendo le bande è possibile ottimizzare i processi produttivi, abbattendo i tempi di acquisizione e produzione. Nel dettaglio, lo scanner estrae immagini su 31 bande differenti ottenendo come risultato un'immagine estremamente fedele all'originale. Si cercherà di confrontare questa immagine con una ottenuta sullo stesso soggetto ma con sole 21 bande.

2.1.1 SSIM

Risulta quindi necessario individuare un metodo oggettivo per il confronto tra l'immagine originale e quella compressa. Si è deciso di utilizzare lo structural similarity index method (SSIM) che è un metodo ampiamente utilizzato per la predizione della qualità percettiva delle immagini televisive e cinematografiche. L'SSIM considera il degrado dell'immagine come cambiamento percepito nelle informazioni strutturali, mentre incorpora anche importanti fenomeni percettivi, inclusi sia il mascheramento della luminanza che i termini di mascheramento del contrasto.

2.1.2 Algoritmo

L'indice SSIM viene calcolato su varie finestre di un'immagine. La misura tra due finestre x e y di dimensione NxN è:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

1. μ_x la media di x ;
2. μ_y la media di y ;
3. σ_x^2 la varianza di x ;
4. σ_y^2 la varianza di y ;
5. σ_{xy} la covarianza di x e y ;
6. $c_1 = (k_1L)^2$, $c_2 = (k_2L)^2$ due variabili per stabilizzare la divisione con il denominatore inadatto;
7. L la gamma dinamica dei valori dei pixel;
8. $k_1 = 0.01$ e $k_2 = 0.03$ predefiniti.

2.1.3 Componenti della formula

La formula SSIM si basa su tre misurazioni di confronto tra i campioni di x e y : luminanza l , contrasto c e struttura S . Le singole funzioni di confronto sono:

$$1. \ l(x, y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1}$$

$$2. \ c(x, y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2}$$

$$3. \ s(x, y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3}$$

con, oltre alle definizioni di cui sopra:

$$c_3 = c_2/2$$

L'SSIM è quindi una combinazione ponderata di tali misure comparative:

$$SSIM(x, y) = [l(x, y)^\alpha \cdot c(x, y)^\beta \cdot s(x, y)^\gamma]$$

3 Svolgimento

3.1 Strumenti

Per affrontare i problemi relativi alla manipolazione delle immagini si è deciso di utilizzare il linguaggio C++ con il supporto della libreria OpenCV. OpenCV è una nota libreria open-source disponibile in C++, Python e Java e permette di effettuare le operazioni più disparate sulle immagini. Inoltre, essa include a sua volta altre librerie come la libtiff che utilizzeremo per la lettura delle immagini.

3.1.1 Tavola Savoia-Navona-Matte

Come soggetto dell'immagine è stata scelta la tavola Savoia-Navona-Matte, rappresentativa di numerose sfumature di colore ed utilizzata per effettuare test sugli scanner.

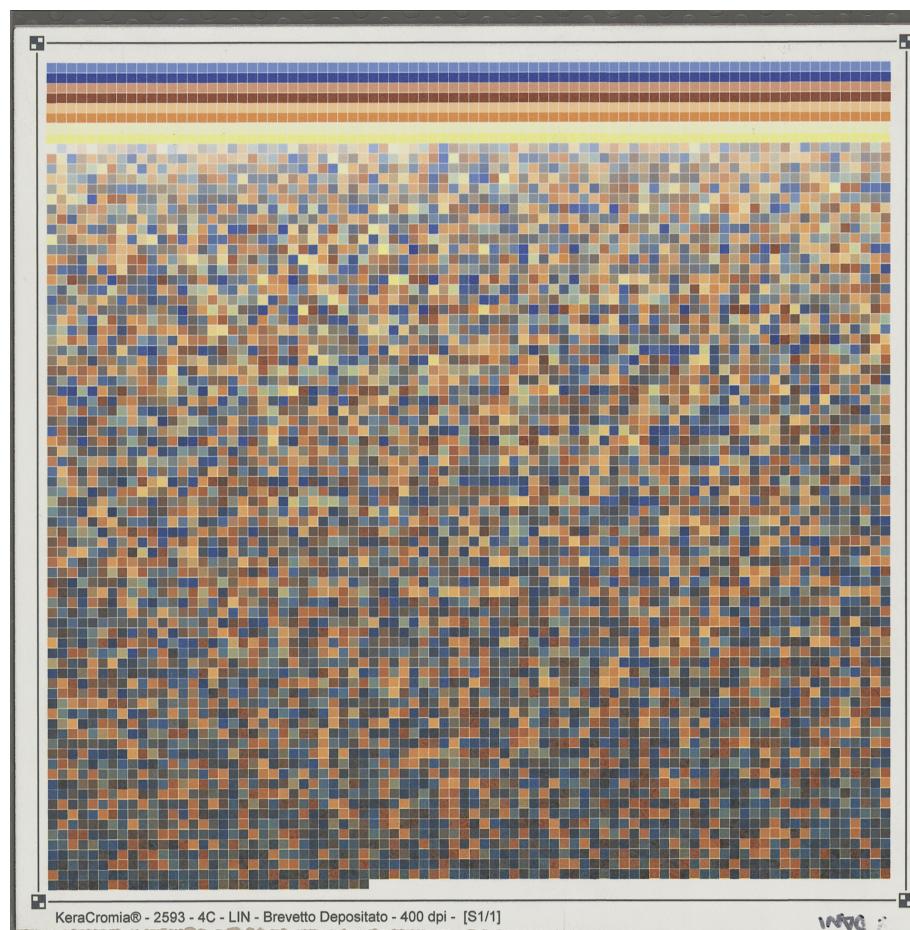


Figura 5: "Tavola Savoia-Navona-Matte"

La tavola è ottenuta come il risultato dell'unione di 31 bande nello spettro visibile da 400nm a 700nm, una ogni 10nm. In Figura 6 sono rappresentati alcuni esempi significativi delle bande sopracitate.

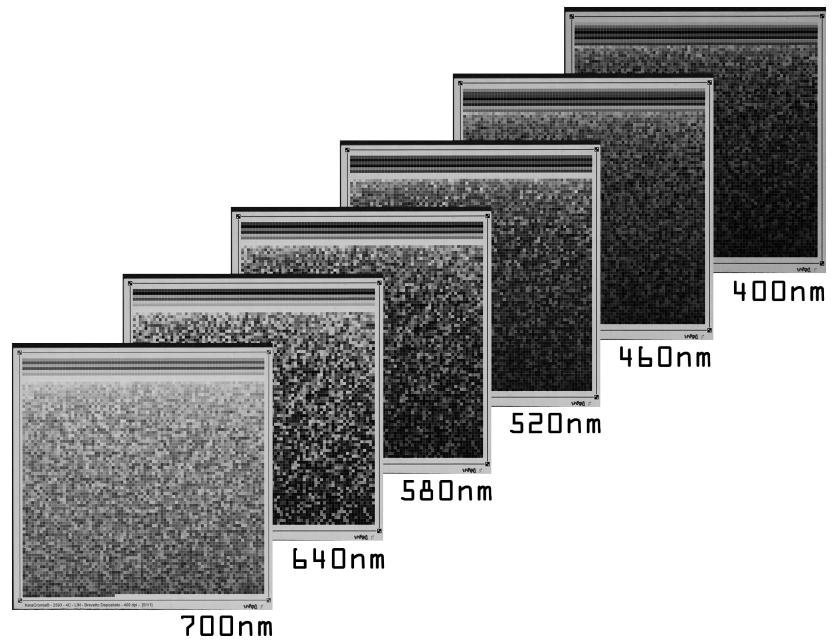


Figura 6: "Suddivisione in bande della tavola Savoia-Navona-Matte"

3.2 Rappresentazione delle immagini

Avendo a disposizione due immagini in formato TIFF, il primo problema da affrontare è riuscire a rappresentare questa immagine in una struttura dati che possa essere poi manipolata per ottenerne informazioni. Per fare ciò sfruttiamo la libreria OpenCV e, in particolare, la libtiff. Tramite le funzioni di questa libreria, si legge l'immagine e la si trasforma in una matrice sfruttando la struttura dati "Mat" messa a disposizione dalla OpenCV:

```
1     bool tifToMat(Mat& image,string imageName){
2         // Read images and create the Mat
3         TIFF* tif = TIFFOpen(imageName.c_str(), "r");
4
5         if (tif) {
6             do {
7                 unsigned int width, height;
8                 uint32* raster;
9
10                // Get the size of the tiff
11                TIFFGetField(tif, TIFFTAG_IMAGEWIDTH, &
12                                width);
13                TIFFGetField(tif, TIFFTAG_IMAGELENGTH,
14                                &height);
15
16                uint npixels = width*height; // Get the total
17                                number of pixels
18
19                raster = (uint32*)_TIFFmalloc(npixels * sizeof(uint32)); // Allocate temp memory (bitmap)
20                if (raster == NULL) // Check the raster's
21                                memory was allocated
22                {
23                    TIFFClose(tif);
24                    cerr << "Raster allocate error" << endl
25                                ;
26                    return false;
27                }
28
29                // Check the tif read to the raster correctly
30                if (!TIFFReadRGBAImage(tif, width, height,
31                                raster, 0))
32                {
33                    TIFFClose(tif);
34                    cerr << "Raster read error" << endl;
35                }
36            }
37        }
```

```

29                     return false;
30     }
31
32     // Create a new matrix of width x height with 8
33     // bits per channel and 4 channels (RGBA)
34     image = Mat(width, height, CV_8UC4);
35
36     // Iterate through all the pixels of the tif
37     for (uint x = 0; x < width; x++)
38         for (uint y = 0; y < height; y++)
39     {
40         uint32& TiffPixel = raster[y*width+x];
41         // Read the current pixel of the TIF
42         Vec4b& pixel = image.at<Vec4b>(Point(
43             y, x)); // Read the current pixel of
44             the matrix
45         pixel[0] = TIFFGetB(TiffPixel); // Set
46             the pixel values as BGRA
47         pixel[1] = TIFFGetG(TiffPixel);
48         pixel[2] = TIFFGetR(TiffPixel);
49         pixel[3] = TIFFGetA(TiffPixel);
50     }
51
52     // Free temp memory
53     _TIFFfree(raster);
54
55     // Rotate the image 90 degrees
56     image = image.t(); // Traspose
57     flip (image, image, 0);
58
59 } while (TIFFReadDirectory(tif)); // Get the next
      tif to go into the channels
60
61     TIFFClose(tif); // Close the tif file
62 }else return false;
63
64 return true;
65 }
```

3.3 Estrazione dell'indice

Adesso che le immagini sono state trasformate in strutture dati, è possibile ricavarne informazioni seguendo le formule precedentemente dette:

```
1      double getSSIM(Mat & img_src, Mat & img_compressed, int
2          block_size, bool show_progress = true)
3      {
4          double ssim = 0;
5
6          if(img_src.cols != img_compressed.cols || img_src.rows !=
7              img_compressed.rows){
8              cout<<"The images got different size"<<endl;
9              return -1;
10         }
11         int nbBlockPerHeight = img_src.rows / block_size;
12         int nbBlockPerWidth = img_src.cols / block_size;
13         // Foreach block in the images
14         for (int k = 0; k < nbBlockPerHeight; k++)
15         {
16             for (int l = 0; l < nbBlockPerWidth; l++)
17             {
18                 int m = k * block_size;
19                 int n = l * block_size ;
20
21                 double avg_o    = mean(img_src(Range(k, k +
22                     block_size), Range(l, l + block_size))) [0];
23                 double avg_c    = mean(img_compressed(Range(k, k
24                     + block_size), Range(l, l + block_size)))[0];
25                 double sigma_o   = variance(img_src, m, n, block_size)
26                     ;
27                 double sigma_c   = variance(img_compressed, m, n,
28                     block_size);
29                 double sigma_co  = covariance(img_src,
30                     img_compressed, m, n, block_size);
31
32                 ssim += ((2 * avg_o * avg_c + C1) * (2 * sigma_co +
33                     C2)) / ((avg_o * avg_o + avg_c * avg_c + C1) * (
34                     sigma_o * sigma_o + sigma_c * sigma_c + C2));
35
36             }
37             // Progress %
38             if (show_progress)
```

```

30             cout << "\r>>SSIM [" << (int) ((( (double)k) /
31                                         nbBlockPerHeight) * 100) << "%]";
32         }
33         ssim /= nbBlockPerHeight * nbBlockPerWidth;
34     if (show_progress)
35     {
36         cout << "\r>>SSIM [100%]" << endl;
37         cout << "SSIM : " << ssim << endl;
38     }
39     return ssim;
40 }
```

La funzione scorre come una finestra mobile (di dimensione block size) contemporaneamente l’immagine originale e quella compressa e ne calcola le varianze, le medie e la covarianza. Ognuno di questi confronti contribuirà al risultato della funzione che sarà sempre un numero compreso tra -1 ed 1. Se le due immagini sono identiche, allora il risultato sarà 1 altrimenti sarà un numero sempre minore fino al -1 che rappresenta due immagini completamente diverse.

3.4 Esempio di esecuzione

Una volta compilato il programma tramite il Makefile, l’utente deve preoccuparsi esclusivamente di fornire le immagini da confrontare ed eseguire il programma.

```

+++ Welcome to SSIM calculator by Davide Cocomini ***
You can use this program to calculate how much similar are two TIFF images with
the same subject
Do you want to use the default settings? Y/N
Y
Starting computation ...
>>SSIM [100%]
SSIM : 0.995251
```

Figura 7: ”Suddivisione in bande della tavola Savoia-Navona-Matte”

4 Conclusioni

L'idea di utilizzare 21 bande anziché 31 ha generato un'immagine che, confrontata con l'originale, ha dato un indice SSIM di: +-X,XXXX Il risultato ci dice quindi che le immagini sono XXXXXXXX, segue quindi che XXXXX.