Elaborazione di Segnali e Immagini (ESI)
LABORATORIO

Lezione 2

Manuele Bicego

Corso di Laurea in Informatica

Dipartimento di Informatica - Università di Verona

Matlab: concetti avanzati

Debug

- MATLAB mette a disposizione un ottimo debugger
- La modalità di debug si attiva dopo aver salvato il file .m e inserendo un breakpoint nell'editor a sinistra del codice (bullet rossa).
- Mandando in esecuzione il codice l'esecuzione si fermerà al primo break point, entrando in modalità di debug prima di eseguire quella riga.
- Da lì, si può proseguire una riga alla volta, oppure fino al prossimo break point, oppure fino alla fine del codice.

Funzioni utili per il debug

- **`dbstop if error`**; entra in modalità di debug nel momento in cui riscontra un errore nel codice, in corrispondenza della riga che ha prodotto l'errore.
  - Utile per ispezionare il workspace alla ricerca del motivo dell'errore;
- **`dbstop if warning`**; stessa cosa di sopra, con la differenza che è uno warning ad attivare la modalità di debug
- **`dbquit`**; esce dalla modalità di debug
- **`dbclear all`**; rimuove tutti i breakpoint

Funzioni utili per il debug

- **CTRL+r** commenta la riga corrente o la porzione di codice selezionata
- **CTRL+t** leva il primo commento sulla sinistra (se presente) o della porzione di codice selezionata
- **CTRL+i** indenta la riga di codice o la porzione di codice selezionata guardando l'intero scope dello script o funzione in cui ci si trova

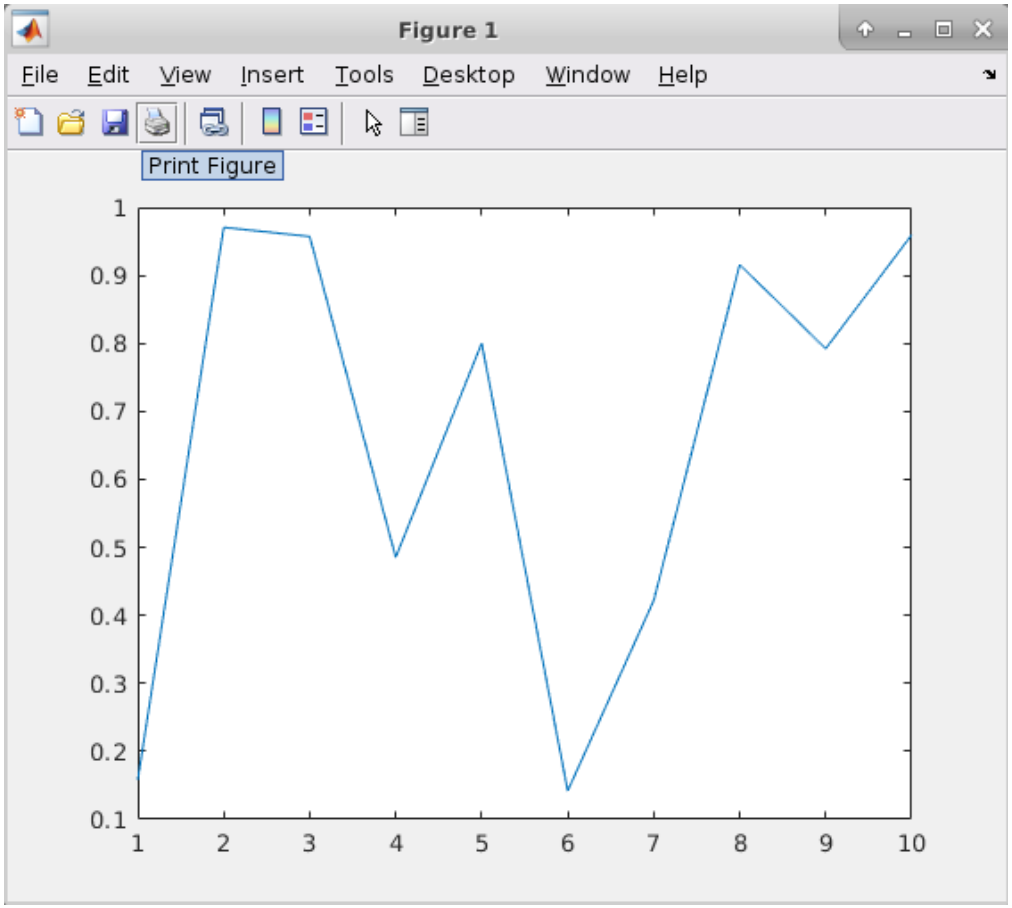
Visualizzazione grafica: Segnali

- MATLAB ha diversi strumenti per la gestione delle parti grafiche
  - è molto adatto per produrre figure varie, inclusi grafici 2D e 3D.
  - Può esportare il risultato (.fig) in diversi formati, come EPS, PDF e JPG.
- Le figure possono essere inizializzate con il comando **`figure(number)`**.

Visualizzazione grafica: Segnali

- line plot: tipo di grafico più frequentemente utilizzato per i segnali
  - richiede due vettori della stessa dimensione
  - X: coordinate orizzontali di ogni punto
  - Y: valori corrispondente nell'asse delle ordinate y.

```
>> A = [1:10];
>> B = rand(1,10);
>> plot(A,B)
```



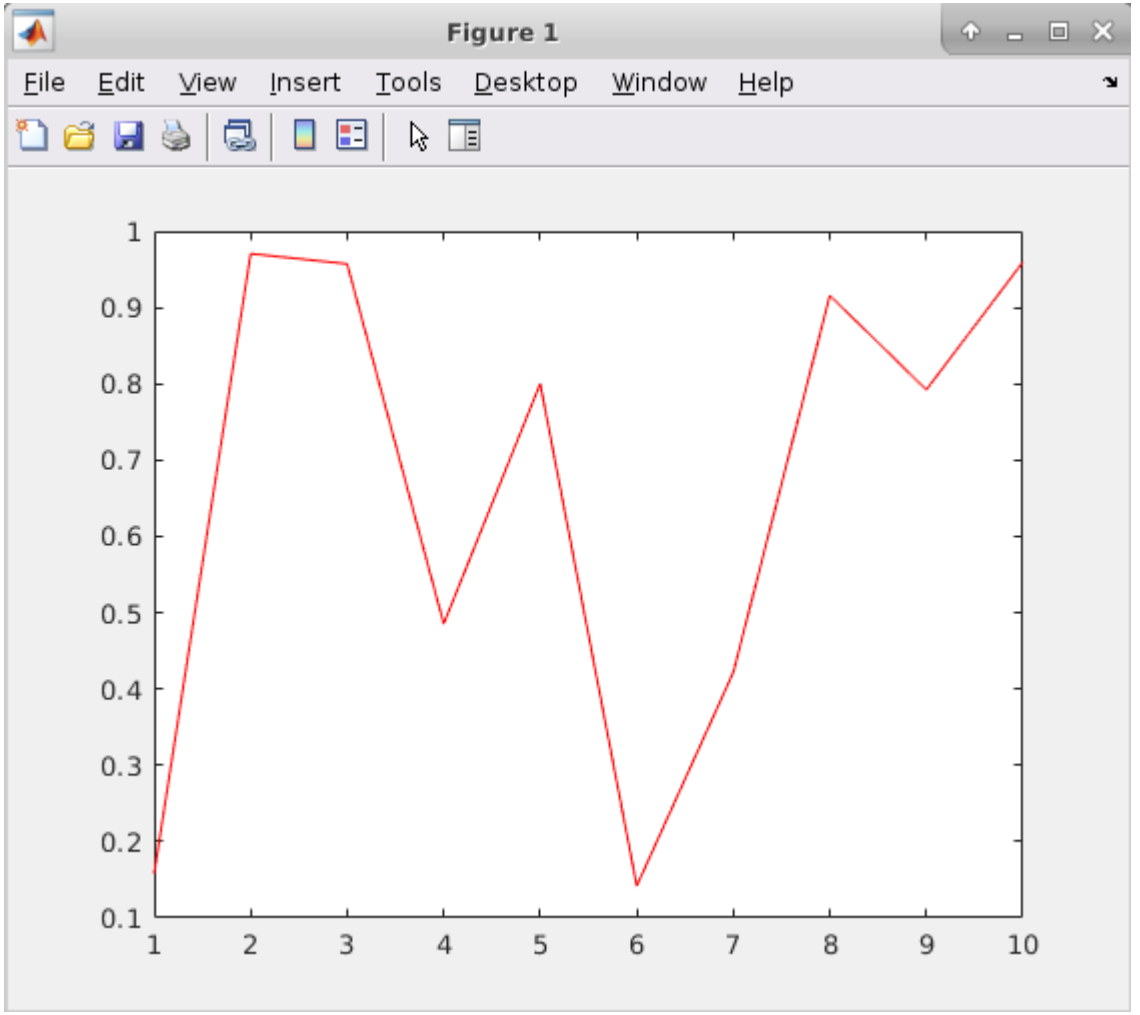
Visualizzazione grafica: Segnali

- MATLAB di default usa la linea continua blu, senza marker, e senza labels agli assi.
- Per migliorare la visualizzazione di un grafico creato con plot, ci sono vari elementi che possono essere aggiunti, ad esempio:
  - Linee con uno stile specifico (in termini di dimensioni, marker, colore etc), nomi agli assi e definizione dei loro limiti, titolo, legenda, griglia,...

Visualizzazione grafica: Segnali

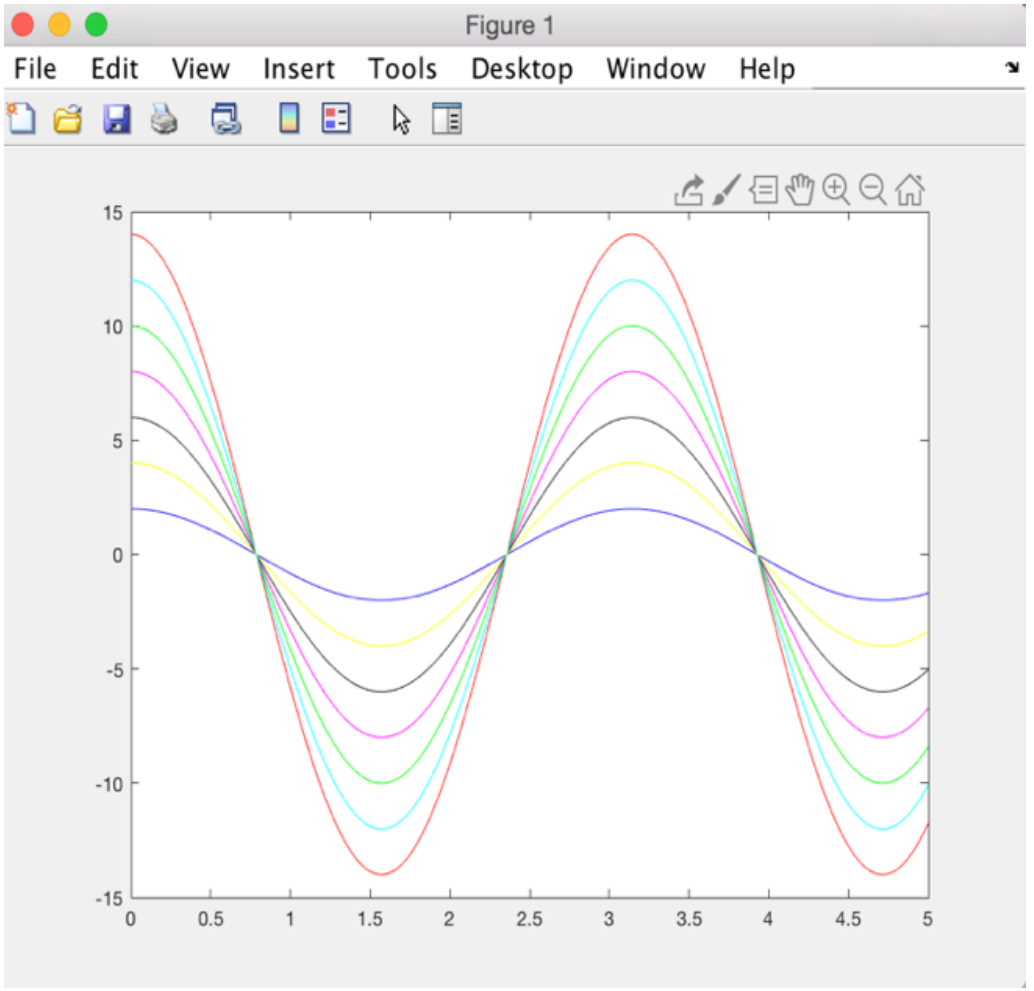
- Colore della linea: posso specificare il colore della linea

```
>> plot(A,B,'r')
```



Visualizzazione grafica: Segnali

- Colore della linea: MATLAB ha 8 colori predefiniti

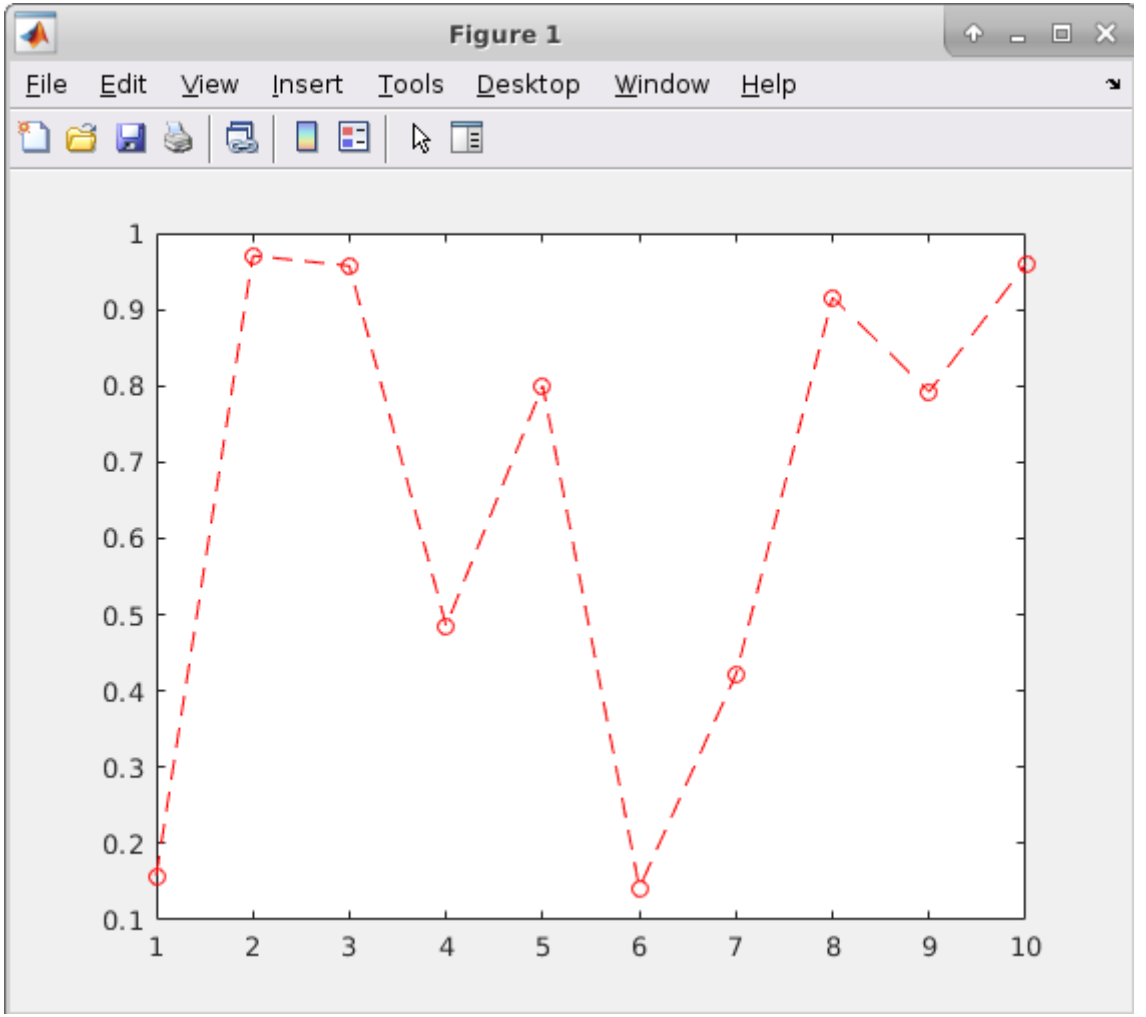


b	blue
g	green
r	red
c	cyan
m	magenta
y	yellow
k	black
w	white

Visualizzazione grafica: Segnali

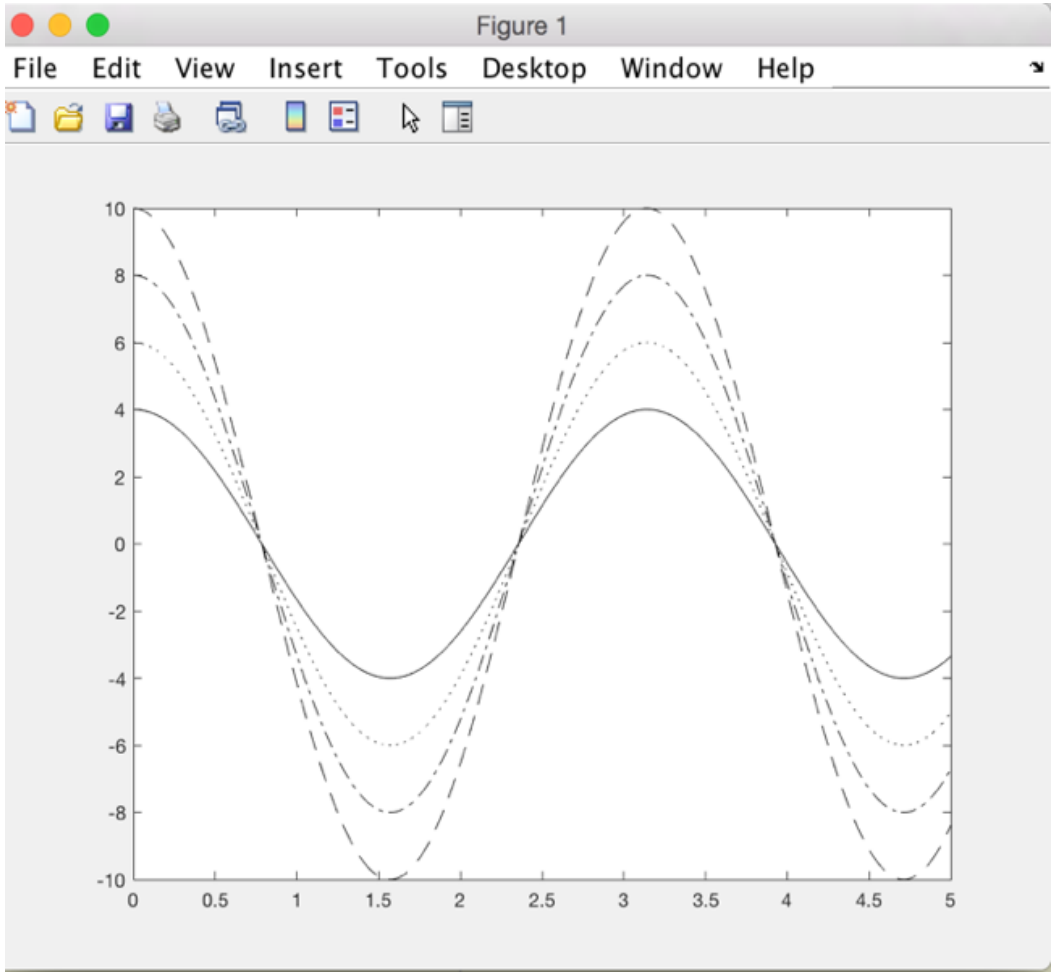
- Tipologia della linea e marker: posso specificare la tipologia (continua, tratteggiata) e il tipo di marker

```
>> plot(A,B,'--or')
```



Visualizzazione grafica: Segnali

- Tipologia della linea e marker:



.	point	-	solid
o	circle	:	dotted
x	x-mark	-.	dashdot
+	plus	--	dashed
*	star	(none)	no line
s	square		
d	diamond		
v	triangle (down)		
^	triangle (up)		
<	triangle (left)		
>	triangle (right)		
p	pentagram		
h	hexagram		

Visualizzazione grafica: Segnali

Altri comandi utili (si veda l'help per il funzionamento)

- **`title('titolo')`**: per settare il titolo
- **`xlabel('name')`**: per settare il nome dell'asse x
- **`ylabel('name')`**: per settare il nome dell'asse y
- **`axis([xmin xmax ymin ymax])`**: per settare minimo/massimo per asse
- **`xlim([]), ylim([])`**: per settare minimo/massimo per un asse
- **`legend('text')`**: per visualizzare una legenda a lato
- **`grid on/off`**: per visualizzare (o no) la griglia nel grafico

Visualizzazione grafica: Segnali

- **`hold on`**: per visualizzare piu' grafici sovrapposti
- **`clf`**: per pulire il contenuto di una figura
- **`stem`** : per visualizzare la sequenza dei dati Y come steli che si estendono per tutta la lunghezza X
- **`subplot`**: per creare piu' immagini all'interno di una singola figura

*[ESEMPIO1, ESEMPIO2 nel file **`Esempi.m`**]*

Visualizzazione grafica: Segnali

- Ultime note:
  - In alcuni casi è utile interagire con i comandi attivabili direttamente dall'interfaccia grafica MATLAB.
  - Le proprietà di un'immagine si possono modificare attraverso i comandi **`get`** (osserva particolari proprietà delle immagini) e **`set`** (cambia tali proprietà su valori decisi da utente).

Suoni

- Per leggere un file contenente un suono

```
[Y, FS] = audioread (FILENAME)
```

- Questo comando legge un file audio specificato dalla stringa FILENAME, e restituisce i dati campionati in Y e la frequenza di campionamento FS, in Hertz.

Suoni

Esempio (probabilmente in Delta non funziona):

- Caricare il file

```
[y,Fs] = audioread('400SineWave.mp3');
```

- Ascoltarlo

```
sound(y(1:Fs*0.5,:),Fs)
```

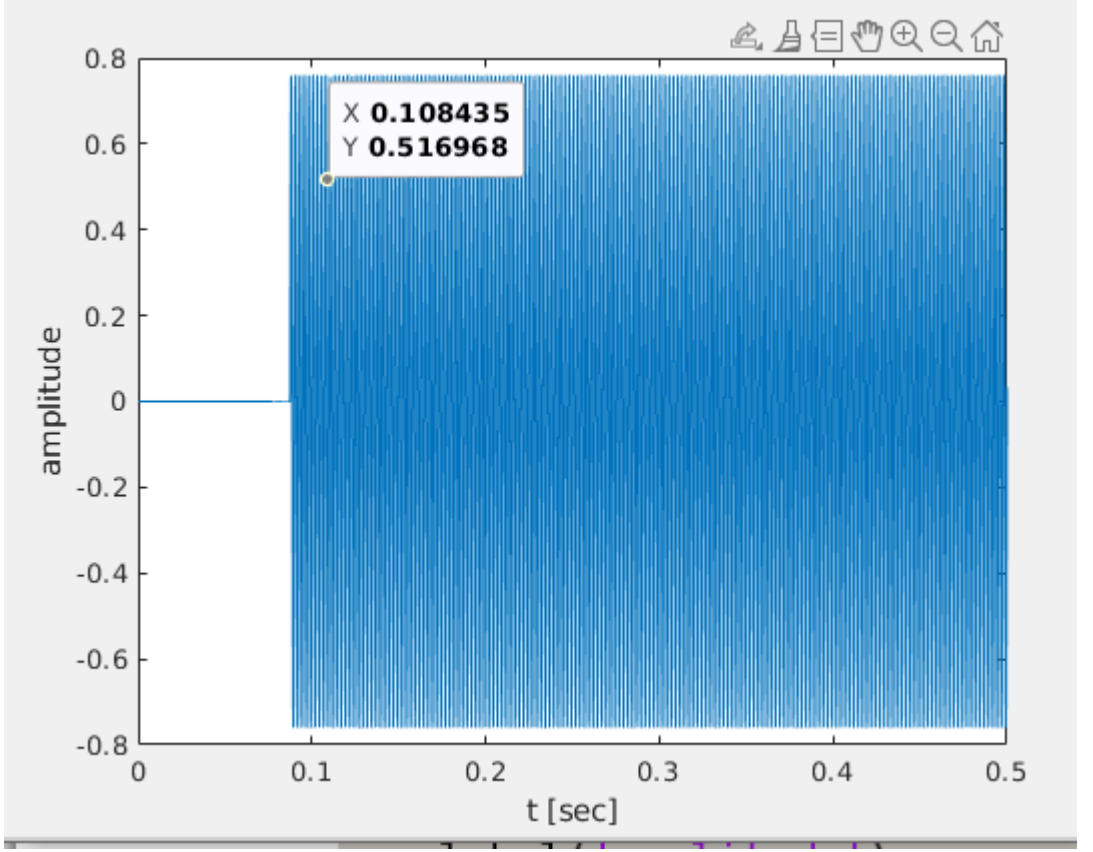
(*Si veda l'help delle due funzioni*)

Suoni

Esempio (probabilmente in Delta non funziona):

- Visualizzarlo (i suoni sono segnali che si possono visualizzare)

```
>> t = 1:size(y(1:Fs/2,1),1);
>> t = t./Fs;
>> figure; plot(t,y(1:Fs/2,1))
>> xlabel('t [sec]')
>> ylabel('amplitude')
>>
```



Suoni

- MATLAB (superiore alla versione 2015) mette a disposizione anche un audio recorder, istanziabile attraverso la funzione **`audiorecorder`**

*[ESEMPIO3, ESEMPIO4 nel file **`Esempi.m`**]*

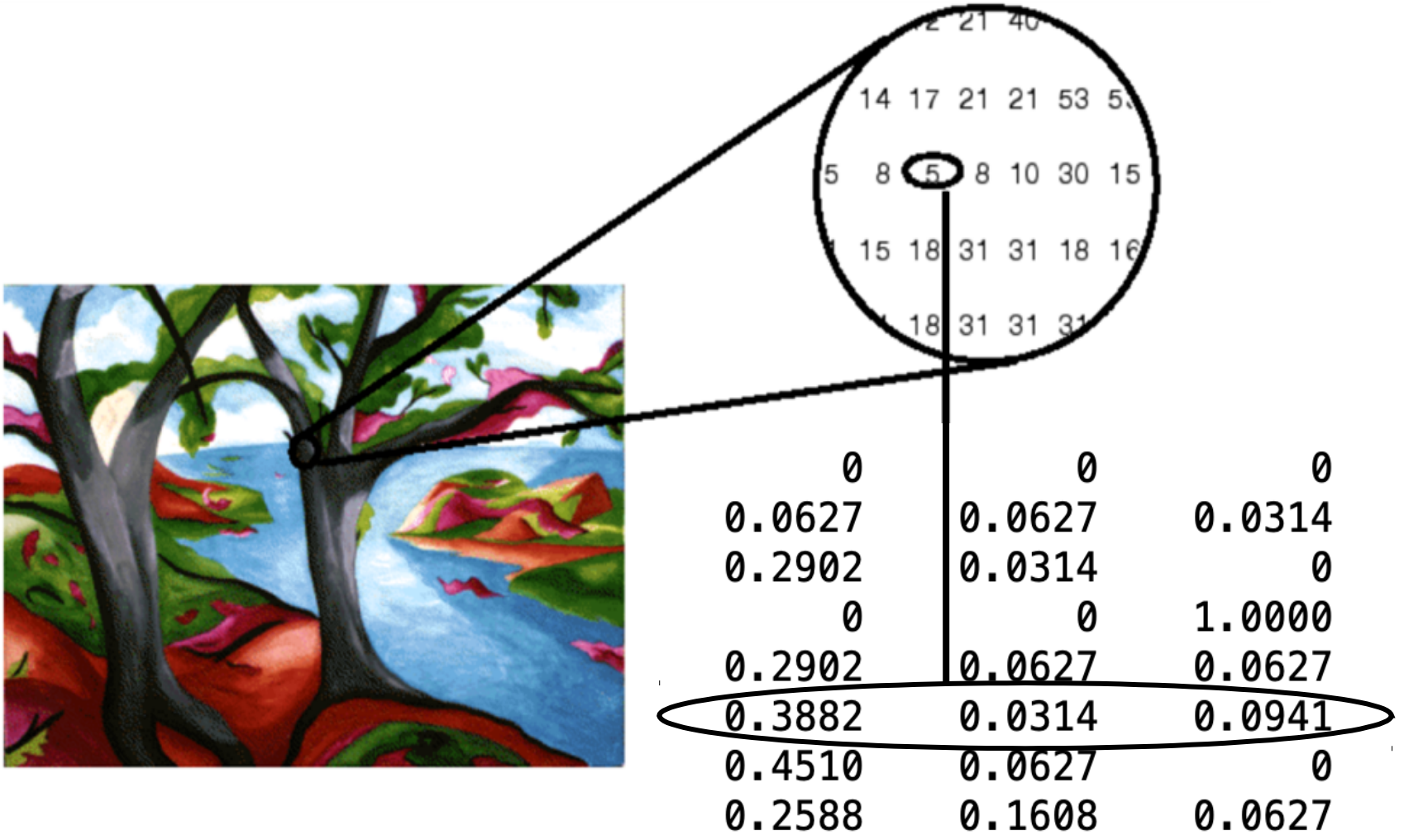
Visualizzazione grafica: Immagini

- In matlab ci sono due tipologie di immagini:
  - Immagine **indicizzata**: matrice di dati i cui valori rappresentano un “puntatore” al colore vero, contenuto in una mappa di colore
  - Immagine **di intensità**: matrice di dati i cui valori rappresentano già i colori (in particolare rappresentano intensità all'interno di un intervallo).

Visualizzazione grafica: Immagini

- Un' **immagine indicizzata** è composta da una matrice di dati, **`X`** e una matrice di colori, **`map`**.
- **`map`** è un array m-by-3 di double contenente valori a virgola mobile nell'intervallo [0, 1];
  - ogni riga specifica i componenti rosso, verde e blu di un singolo colore.
- Un'immagine indicizzata utilizza la "mappatura diretta" dei valori dei pixel ai valori della mappa di colori:
  - il colore di ciascun pixel dell'immagine viene determinato mappando il valore di X al corrispondente colore nella mappa di colori (I valori di X quindi devono essere numeri interi)

Esempio: immagine trees.tif in Matlab



Visualizzazione grafica: Immagini

- **Immagine di intensità**: matrice di dati, I, i cui valori rappresentano intensità all'interno di un intervallo:
  - MxN (singolo canale): il valore di ogni pixel indica il suo livello di grigio
  - MxNxK (3 canali): i tre valori di ogni pixel indicano il colore (tipicamente secondo la codifica RGB)

Esempio: immagine peppers.png

49	55	56	57	52	53
58	60	60	58	55	57
58	58	54	53	55	56
83	78	72	69	68	69
88	91	91	84	83	82
69	76	83	78	76	75
61	69	73	78	76	76

Red

64	76	82	79	78	78
93	93	91	91	86	86
88	82	88	90	88	89
125	119	113	108	111	110
137	136	132	128	126	120
105	108	114	114	118	113
96	103	112	108	111	107

Green

66	80	77	80	87	77
81	93	96	99	86	85
83	83	91	94	92	88
135	128	126	112	107	106
141	129	129	117	115	101
95	99	109	108	112	109
84	93	107	101	105	102

Blue

Visualizzazione grafica: Immagini

- Per leggere un file contenente una immagine in scala di grigi o a colori, MATLAB mette a disposizione il comando **`imread`**

```
>> I = imread (filename,fmt);
>> [I,map] = imread (...) ;
>> [I,map] = imread (filename);
>> [I,map] = imread (URL ,...) ;
```

- Legge un'immagine di formato "fmt" e di nome "filename", oppure caricata dal web dall'indirizzo specificato da "URL"

Visualizzazione grafica: Immagini

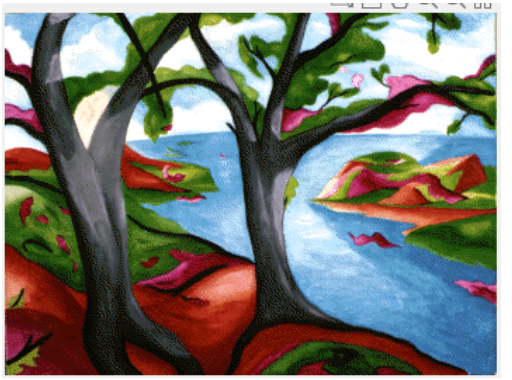
- Immagine **indicizzata**: in questo caso **`imread`** ritorna l'immagine (nella variabile **`I`**) e la mappa di colore (salvata in **`map`**)

- Immagine **di intensità**: in questo caso **`imread`** ritorna l'immagine (nella variabile **`I`**) e una **`map`** nulla

Visualizzazione grafica: Immagini

- Più nel dettaglio: la matrice `I` ha le seguenti caratteristiche:
  - di dimensioni M × N se immagine **indicizzata**, a **scala di grigio** oppure **binaria**, M × N sono i pixels delle M righe e N colonne;
  - di dimensioni M × N × 3 se **immagine di intensità a colori** rappresentati con il modello RGB o HSV;
  - di dimensioni M × N × 4 se **immagine di intensità** a **colori** rappresentati con il modello CMYK (tipico dei file *.tiff).

Esempio: **immagine indicizzata** (trees.tif in Matlab)



```
>> [Itrees,map] = imread('trees.tif');
>> whos Itrees
  Name           Size                Bytes  Class

  Itrees       258x350               90300  uint8

>> whos map
  Name          Size                Bytes  Class

  map         256x3                  6144  double
```

Matrice MxN, immagine indicizzata a colori

Se l'elemento (13,15) della matrice contiene il valore 5, esso verrà visualizzato con il colore indicato nella sesta riga della matrice map (la prima riga corrisponde a 0, la seconda a 1 etc)

Esempio: **immagine di intensità** a toni di grigio (cameraman.tif in Matlab)

```
>> [Icam, map] = imread('cameraman.tif');
>> whos Icam
  Name         Size                Bytes  Class

  Icam       256x256               65536  uint8

>> whos map
  Name       Size                Bytes  Class

  map        0x0                     0  double
```

Matrice MxN, immagine a toni di grigio

Se l'elemento (13,15) della matrice contiene il valore 5, esso verrà visualizzato con livello di grigio 5

Esempio: **immagine di intensità** a colori (immagine peppers.png)

```
>> [Ipep, map] = imread('peppers.png');
>> whos Ipep
  Name         Size                    Bytes  Class

  Ipep       384x512x3                589824  uint8

>> whos map
  Name       Size                Bytes  Class     Att

  map        0x0                     0  double
```

Matrice MxNx3, immagine a colori

Se l'elemento (13,15) della matrice contiene i tre valori di RGB per visuallizare il colore

*[ESEMPIO5 nel file **Esempi.m**]*

Visualizzazione grafica: Immagini

- Per visualizzare una immagine in scala di grigi o a colori contenuta in una matrice I, MATLAB mette a disposizione una serie di comandi.
- Il comando principale è **`imshow`**

```
>> imshow(I)
```

visualizza l'immagine in scala di grigi o a colori contenuta in una matrice I

Visualizzazione grafica: Immagini

Altre varianti:

- **`>> imshow (I,map) % Per immagine indicizzata`**
  - visualizza una immagine indicizzata con la relativa mappa di colore contenuta nella variabile map
- **`>> imshow (I,[low high]);  % Per scala di grigi`**
  - Si visualizzano solo i pixels con valori all'interno dell'intervallo [low high], gli altri valori saranno sostituiti con il colore nero, se il loro valore è minore di low, altrimenti, se maggiore, con il colore bianco.

Visualizzazione grafica: Immagini

- **`>> imshow (I,[]);   % Per scala di grigi`**
  - Come la precedente con

    `low = min(I(:))`

    `high = max(I(:))`

- Per fare più plot nella stessa figura:
  `subplot(nrighe,ncolonne,posizione_griglia)`

```
>> I = imread ('cameraman.tif');
>> figure (1)
>> subplot (1,2,1), imshow (I)
>> subplot (1,2,2), imshow (I,[0 80])
```

*[ESEMPIO6 nel file **Esempi.m**]*

Visualizzazione grafica: Immagini

- Un ulteriore modo alternativo per visualizzare una immagine in scala di grigi o a colori contenuta in una matrice A è quello di utilizzare il comando **`imagesc(A)`**
- L'immagine ottenuta è visualizzata utilizzando tutto il range di colori compreso nella mappa di colore (colormap).
  - Se un'immagine ha valore minimo m e massimo M, e supponendo una colormap gray, m verrà mappato a nero, M a bianco, e tutti i valori intermedi saranno interpolati a 255 valori.
- **`imagesc`** nasce per visualizzare una generica informazione bidimensionale, massimizzando l'utilizzo della mappa cromatica.

Ulteriori comandi

- **`B = imresize(A,scale)`**
  - Permette di riscalare un’immagine (RGB o in scala di grigi), rimpicciolendola oppure ingrandendola a seconda del valore del parametro scale.
  - Se scale è tra 0-1, B<A, al contrario se è maggiore di 1, B>A come dimensioni.

Ulteriori comandi

- **`J = imrotate(I,angle)`**
  - Permette di ruotare un'immagine, in senso antiorario, oppure in senso orario se il valore di angle è negativo.
  - Angle è espresso in gradi;

- **`J = imcrop(I)`**
  - Visualizza l'immagine e apre un tool iterativo per selezionarne una porzione.

Ulteriori comandi

- **`I = rgb2gray(RGB)`**
  - converte una immagine da RGB a scala di grigi

- **`imwrite (I, ‘filename’, ‘fmt’),`**
  `imwrite (I, map, ‘filename’, ‘fmt’);`
  - Per scrivere un file contenete una immagine in scala di grigi o a colori

Esercizi principali

Esercizio 1

- Prendete la foto di Paperino oppure fatevi una foto al volto.
- Copiate questa foto nella directory di lavoro, e caricatela attraverso MATLAB.
- Attraverso opportune indicizzazioni della matrice in cui è contenuta la foto, sostituite ai pixel che rappresentano gli occhi dei pixel neri, facendo comparire una sorta di occhiali da sole.
  - NOTA: Ricordo che il valore nero si ottiene con una terna RGB = [0,0,0].

Esercizio 1

- Ruotate l'immagine originale di 45 gradi verso sinistra
- Estraete la parte dell'immagine contenente gli occhi (comando **`imcrop`**)
- Create una figura con quattro plot (comando **`subplot`**) per visualizzare:
  - L'immagine originale
  - L'immagine modificata
  - L'immagine ruotata
  - L'immagine con il dettaglio degli occhi

Esercizio 1

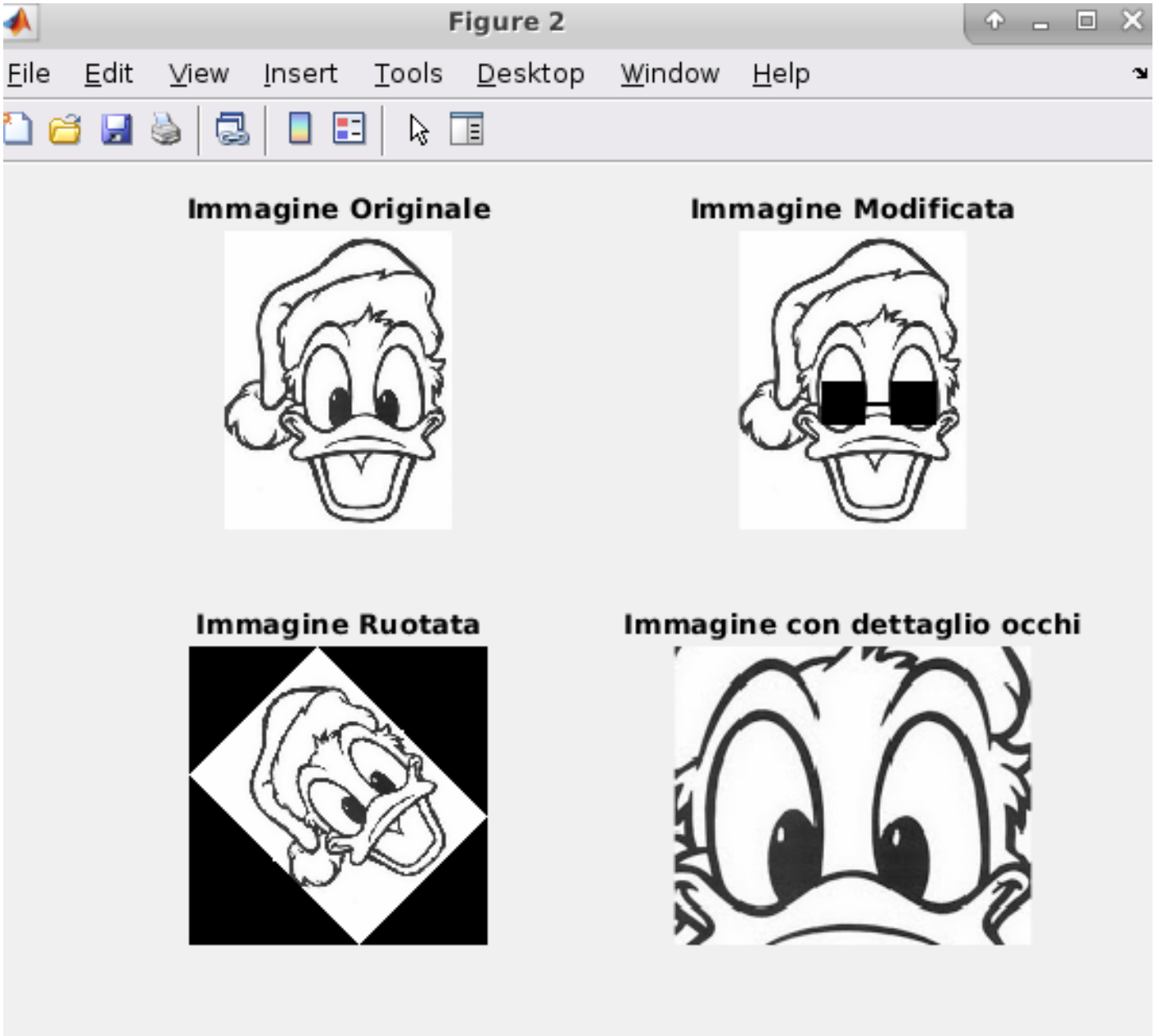
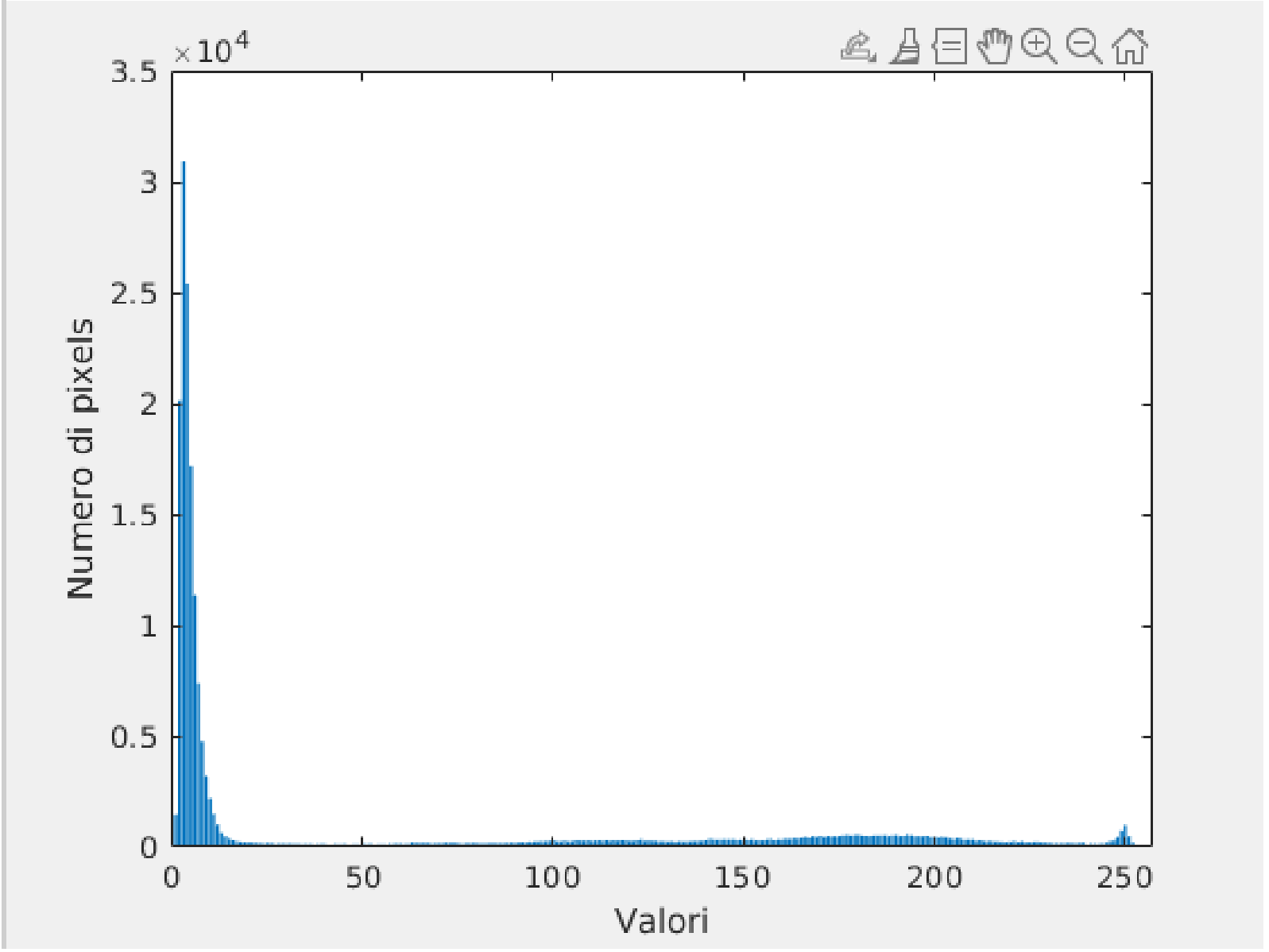


Figure 2
File Edit View Insert Tools Desktop Window Help
Immagine Originale
Immagine Modificata
Immagine Ruotata
Immagine con dettaglio occhi

Esercizio 2

- Realizzare una funzione che, data l'immagine a livelli di grigio **`moon.tif`**, conti quanti pixel (= entries i,j all'interno della matrice) assumono un particolare valore di grigio, per tutti i valori di grigio compresi tra 0 e 255.
- Il risultato sarà un vettore di naturali di dimensionalità (256,1) (chiamato istogramma).
- Provare a visualizzare questo vettore usando il comando bar.
  - Nota: non utilizzare comando **`imhist`** o **`histogram`**

Esercizio 2


×10^4
3.5
3
2.5
2
1.5
1
0.5
0
Numero di pixels
0
50
100
150
200
250
Valori

Esercizi extra

Esercizio 3

- Caricare nel workspace l'immagine "**`seattle.png`**" e assegnarla alla variabile **`I`**
- Costruire una nuova matrice **`Ih`** in cui ad ogni elemento di `I` viene sottratto il suo precedente sulle colonne (in valore assoluto).
  - In altre parole, l'elemento (i,j) della nuova matrice Ih deve essere uguale a **`abs(I(i,j)-I(i-1,j))`** (attenzione agli indici di inizio e fine del for)
- Ripetere l'esercizio costruendo l'immagine **`Iv`**, ottenuta sottraendo ad ogni elemento quello adiacente sulle righe: **`Iv(i,j)=abs(I(i,j)-I(i,j-1))`**
- Cosa rappresentano **`Ih`** e **`Iv`**?

Esercizio 3

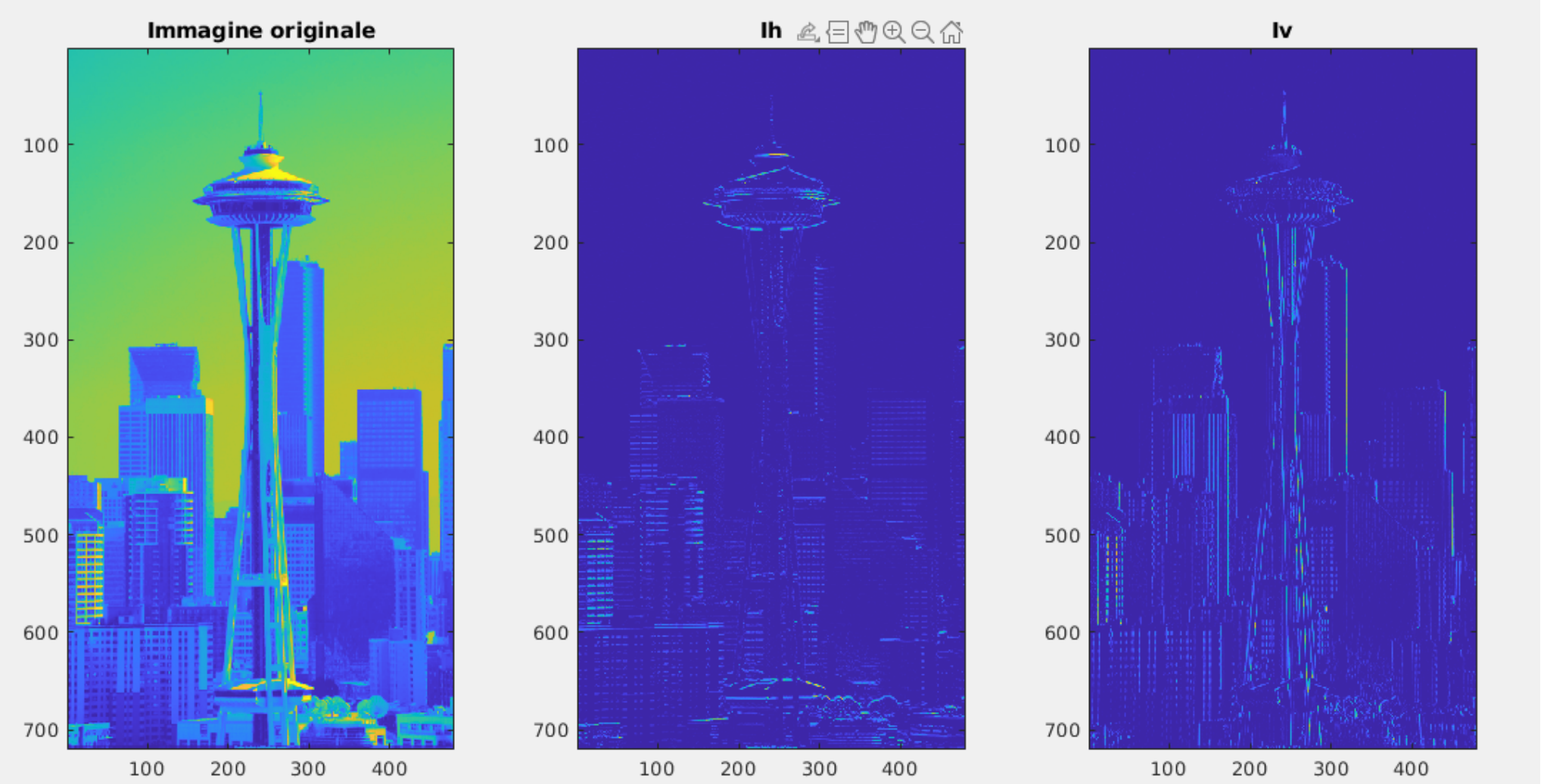


Immagine originale
Ih
Iv
100
200
300
400
500
600
700
100
200
300
400

Esercizio 4

- Scrivere una funzione, **`MYflip`**, che dato un vettore
  - ne crei una copia “riflessa” (per esempio, da [1 2 3] ottengo [3,2,1])
  - La concateni a sinistra al vettore originale (ossia [3,2,1,1,2,3])

Questa funziona tornerà utile nell’analisi frequenziale di segnali.

```
>> vettore_A = [1 2 3]

vettore_A =

     1     2     3

>> res = MYflip(vettore_A)

res =

     3     2     1     1     2     3
```

Esercizio 5

- Caricare l'immagine "**`cells.png`**", che contiene una visualizzazione di cellule U2OS (Human Bone Osteosarcoma)
- Provare ad evidenziare i nuclei.
  - Suggerimento: i nuclei sono caratterizzati da valori sopra una certa soglia (provare con diverse soglie)
- Costruire inoltre una figura con due subplot dove l'immagine originale viene affiancata all'immagine in cui sono evidenziati i nuclei.
  - Nota: non usare la funzione **`imbinarize`**

Esercizio 5

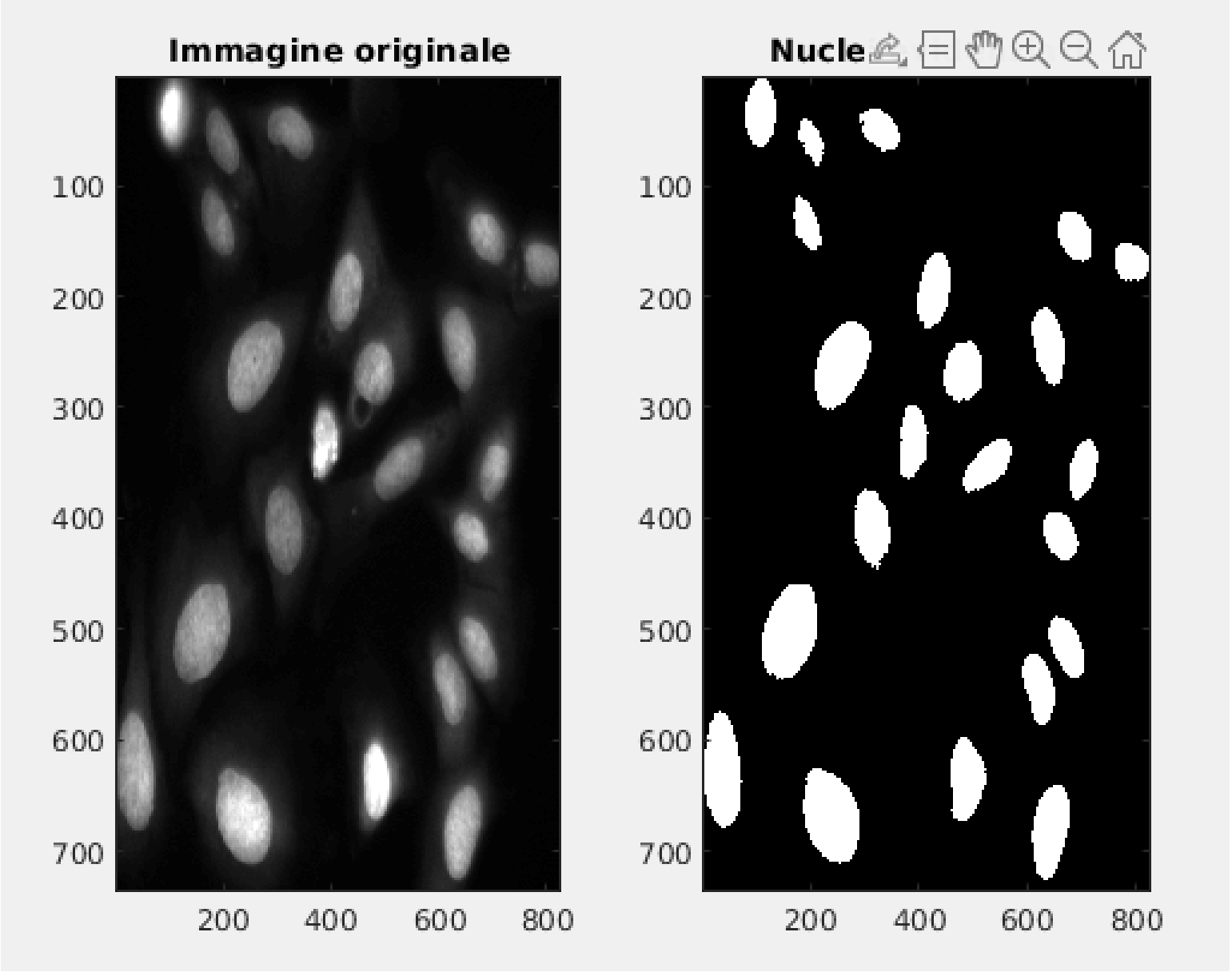
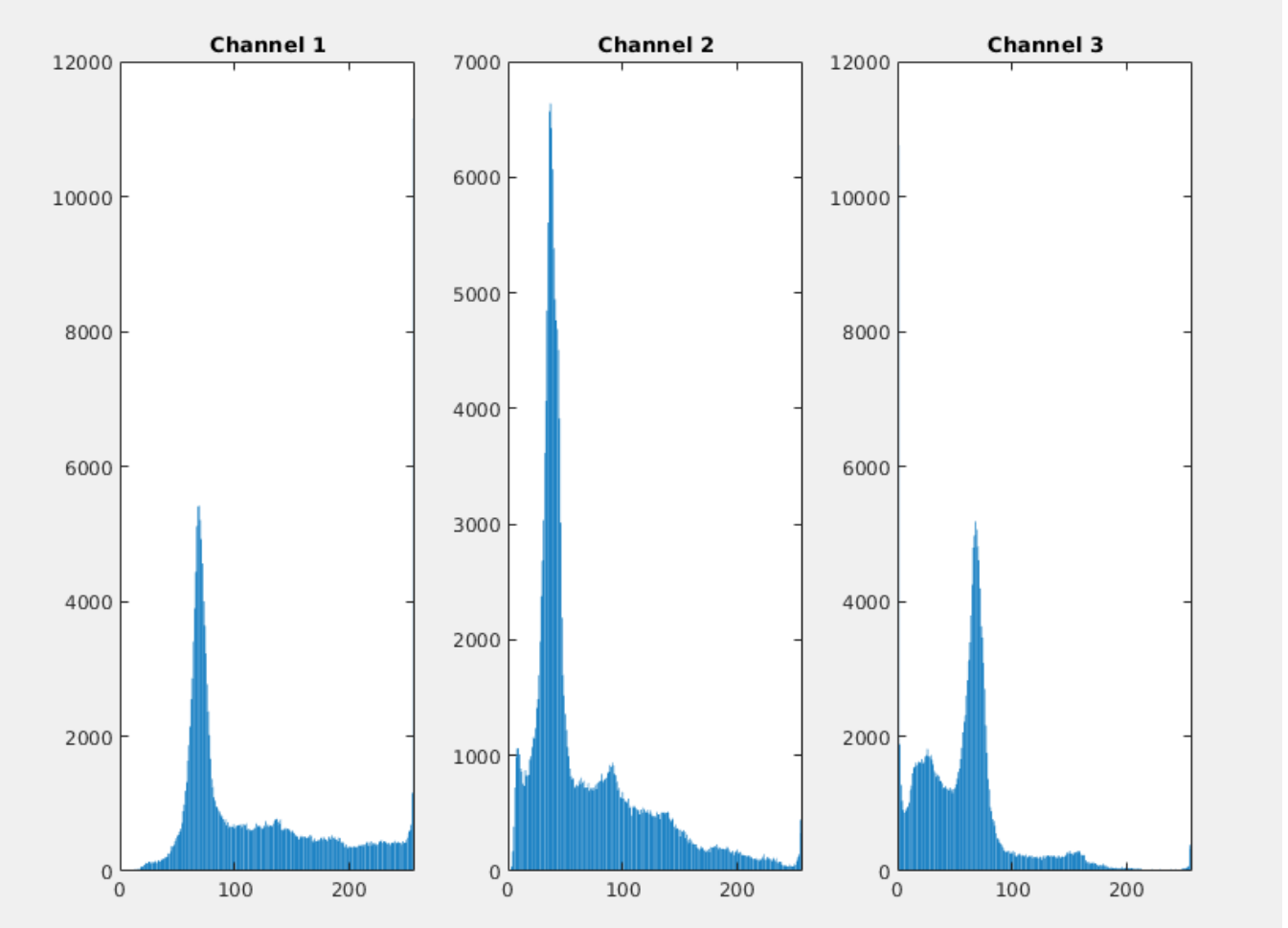


Immagine originale
Nucle
100
200
300
400
500
600
700
200
400
600
800
100
200
300
400
500
600
700
200
400
600
800

Esercizio 6

- Estendere l’esercizio 2 al caso di immagini a colori, in particolare utilizzando l’immagine “**`peppers.png`**”
  - Calcolare l’istogramma per ogni canale di colore.
  - Domanda: qual’è il canale con più valori diversi? Suggerimento: contare i valori dell'istogramma diversi da zero

Esercizio 6


Channel 1
12000
10000
8000
6000
4000
2000
0
0
100
200
Channel 2
7000
6000
5000
4000
3000
2000
1000
0
0
100
200
Channel 3
12000
10000
8000
6000
4000
2000
0
0
100
200