

GIACOMO DIAZ

# APPUNTI DI INFORMATICA

## con applicazioni di image processing e image analysis

Premessa: il corso prevede che lo studente abbia già confidenza con l'uso del computer e con l'ambiente operativo Windows. In particolare lo studente deve essere in grado di:

- accendere e spegnere il computer in modo corretto
- aprire e chiudere correttamente una applicazione o programma
- utilizzare il menù di sistema associato alla finestra dell'applicazione
- trovare, cancellare, rinominare, copiare, spostare un file e modificarne gli attributi
- fare le stesse operazioni su più file utilizzando i caratteri jolly (\* ?) e la tecnica 'drag & drop'
- applicare i comandi copia e incolla a cartelle, files, porzioni di testo e di immagini

### 1. ANATOMIA DI UN NUMERO

#### Il sistema decimale

I nostri numeri si dicono decimali. Infatti i nostri numeri utilizzano 10 cifre: da 0 a 9. 10 è la cosiddetta 'base' del sistema di numerazione. Ogni qualvolta si supera il valore massimo esprimibile con 1, 2 o tre cifre (9, o 99 o 999) si sostituiscono i 9 con zero e si aggiunge un 1 a sinistra (10, 100, o 1000). Quindi la posizione della cifra all'interno di un numero decimale indica quante unità, decine, migliaia, ecc.

Ricordiamo che, in base 10,

1 equivale a  $10^0$

10 equivale a  $10^1$

100 equivale a  $10^2$

1000 equivale a  $10^3$

ecc.

Ad es. analizziamo il numero 13654

1	3	6	5	4
$1 \times 10^4 +$	$3 \times 10^3 +$	$6 \times 10^2 +$	$5 \times 10^1 +$	$4 \times 10^0$
10000 +	3000 +	600 +	50 +	4

#### Il sistema esadecimale

Se come base prendiamo 16, avremmo a disposizione solo **sedici** cifre:

0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

In base 16,

$16^0$  equivale ad 1

$16^1$  equivale a 16

$16^2$  equivale a 256

$16^3$  equivale a 4096

$16^4$  equivale a 65536

$16^5$  equivale a 1048576

ecc.

Ad es. il numero esadecimale A86E0F

A	8	6	E	0	F
$10 \times 16^5 +$	$8 \times 16^4 +$	$6 \times 16^3 +$	$14 \times 16^2 +$	$0 \times 16^1 +$	$15 \times 16^0 +$
10485760 +	524288 +	24576 +	3584 +	0 +	15

corrisponde al valore decimale 11038223

### Il sistema binario

Se come base prendiamo 2, avremmo a disposizione solo **due** cifre: 0 e 1.

Ricordiamo che, in base 2,

$2^0$  equivale ad 1

$2^1$  equivale a 2

$2^2$  equivale a 4

$2^3$  equivale a 8

$2^4$  equivale a 16

ecc.

Ad es. il numero binario 100110

1	0	0	1	1	0
$1 \times 2^5 +$	$0 \times 2^4 +$	$0 \times 2^3 +$	$1 \times 2^2 +$	$1 \times 2^1 +$	$0 \times 2^0 +$
32 +	0 +	0 +	4 +	2 +	0

corrisponde al valor decimale 38

### In generale

il numero di interi, zero compreso, rappresentabile da **n** cifre in un determinato sistema con base **b** ( $b = 2, 8, 10, 16, \dots$  = binario, ottale, decimale, esadecimale, ...) è dato da

$$b^n$$

Ad es.

con 3 cifre ( $n=3$ ) del sistema decimale ( $b=10$ ) si possono rappresentare:  $10^3 = 1000$  dati

con 4 cifre ( $n=4$ ) del sistema esadecimale ( $b=16$ ) si possono rappresentare:  $16^4 = 65,536$  dati

con 8 cifre ( $n=8$ ) del sistema binario ( $b=2$ ) si possono rappresentare:  $2^8 = 256$  dati

Ovviamente, partendo da zero, il valore massimo rappresentato sarà

$$b^n - 1$$

Ad es.

con 3 cifre ( $n=3$ ) del sistema decimale ( $b=10$ ) si può ottenere al massimo il valore:  $10^3 - 1 = 999$

con 2 cifre ( $n=4$ ) del sistema esadecimale ( $b=16$ ) si può ottenere al massimo il valore:  $16^4 - 1 = 65535$

con 8 cifre ( $n=8$ ) del sistema binario ( $b=2$ ) si può ottenere al massimo il valore:  $2^8 - 1 = 255$

### Esercizio

Trasformare in binario ed esadecimale i valori decimali

decimale	binario	esadecimale
12		
16		
255		
256		
4096		
0		

## 2. BITS e BYTES

I computer, per necessità, utilizzano il sistema binario in quanto la rappresentazione dei valori 0 e 1 si adatta bene alla condizione più o meno polarizzata di un magnete o ai livelli basso e alto di un'onda elettrica. Il computer quindi utilizza solo le cifre 0 o 1. Queste cifre sono dette bit (pronuncia: bit). I bit comunque sono sempre riuniti in gruppi di 8. 8 bit formano un byte (pronuncia: bait). Quindi un byte corrisponde ad un numero binario di 8 cifre con cui si può rappresentare un valore compreso tra 0 e 255 (inclusi).

Un byte basta e avanza per indicare tutte le lettere dell'alfabeto (21 in italiano, qualcosa di più in altre lingue). Invece non basta per indicare le pagine di un libro, o i giorni dell'anno, o comunque numeri maggiori di 255. In tutti questi casi i bytes vengono a loro volta raggruppati per ottenere valori di 16, 32, 64 o più bit. Comunque sempre multipli di 8 bit.

Schematicamente:

1 byte (8 bit) possono rappresentare  $2^8 = 256$  valori (da 0 a 255)

2 bytes (16 bit) possono rappresentare  $2^{16} = 65,536$  valori (da 0 a 65,535)

4 bytes (32 bit) possono rappresentare  $2^{32} = 4,294,967,296$  valori (da 0 a 4,294,967,295)

ecc.

## 3. OPERAZIONI ED OPERATORI

Per fare una operazione occorre un operatore ed uno o più operandi: 1 solo operando se l'operatore è unario (es. radice quadrata di 4), 2 operandi se l'operatore è binario (es. 5 moltiplicato 3), più operandi se l'operatore è multiplo (es. la media di 4, 6, 8, 9, 10).

In informatica esistono operatori di tipo :

logico, che operano a livello di bit, in cui 1 = vero e 0 = falso

numerico, che operano a livello di valori di byte e multipli di byte

### Operatori logici

Questi operatori esigono operandi di natura dualistica (es un fatto vero o falso; un oggetto presente o assente; ecc.). In campo informatico l'operando dualistico è il bit in quanto il bit può avere come valore 0 o 1. Non è però escluso che si possano utilizzare anche bytes stabilendo il dualismo tra valori inferiori o uguali a ... e valori superiori a .... Occupiamo però ora solo di operazioni logiche tra bit (bit-wise logic operations).

NOT (unario)

questo operatore restituisce il complementare: 1 se il bit è 0 e viceversa. In altre parole, l'operatore NOT inverte i bit. Questo produce l'effetto 'negativo'.

Es.

1	0	1	1	0	1	0	0
---	---	---	---	---	---	---	---

NOT

=

0	1	0	0	1	0	1	1
---	---	---	---	---	---	---	---

AND (binario)

questo operatore restituisce 1 quando i bit sono entrambi 1; altrimenti restituisce 0.

Es.

1	0	1	1	0	1	0	0
---	---	---	---	---	---	---	---

AND

0	1	1	1	0	1	1	0
---	---	---	---	---	---	---	---

=

0	0	1	1	0	1	0	0
---	---	---	---	---	---	---	---

**OR (binario)**

questo operatore restituisce 1 quando almeno un bit è 1; altrimenti restituisce 0.

Es.

1	0	1	1	0	1	0	0
0	1	1	1	0	1	1	0
1	1	1	1	0	1	1	0

OR

risultato

**XOR (binario)**

questo operatore restituisce 1 quando i bit sono diversi. Se i bit sono uguali restituisce 0.

Es.

1	0	1	1	0	1	0	0
0	1	1	1	0	1	1	0
1	1	0	0	0	0	1	0

XOR

risultato

Riapplicando il 2° operando al risultato dell'operazione si riottiene il primo operando.

1	1	0	0	0	0	1	0
0	1	1	1	0	1	1	0
1	0	1	1	0	1	0	0

(risultato precedente)

XOR

(2° operando precedente)

risultato

Due operazioni XOR in serie consentono di disegnare e poi di cancellare una figura da una immagine. Questo metodo si può utilizzare per produrre animazioni, ecc. Il movimento della freccetta del mouse sul monitor è realizzato in questo modo.

**NAND (binario)**

questo operatore restituisce il complemento (negativo) di AND: 0 quando i bit sono entrambi 1; 1 negli altri casi.

Es.

1	0	1	1	0	1	0	0
0	1	1	1	0	1	1	0
1	1	0	0	1	0	1	1

NAND

risultato

**NOR (binario)**

questo operatore restituisce il il complemento (negativo) di OR: 0 quando almeno un bit è 1; altrimenti 1.

Es.

1	0	1	1	0	1	0	0
0	1	1	1	0	1	1	0
0	0	0	0	1	0	0	1

NOR

risultato

**Operatori numerici**

Questi sono i normali operatori che eseguono la somma, la moltiplicazione, ecc. tra bytes. Tuttavia, se sin dalle scuole elementari abbiamo imparato a fare le addizioni, moltiplicazioni e divisioni - tra numeri decimali - , è abbastanza complicato moltiplicare o dividere bytes. Inoltre, se i simboli di queste operazioni sono ormai standardizzati (+, -, ×, /, √, ecc), le sigle degli operatori numerici non sono standard (mentre lo sono quelle come AND, OR, XOR, ecc. degli operatori logici). A seconda del numero di operandi possiamo classificare gli operatori numerici in

**operatori unari (un solo operando)**

RAD(x) restituisce la radice quadrata di x

LOG(x) restituisce il logaritmo decimale di x

ABS(x) restituisce valore assoluto di x

INT(x) restituisce il valore intero di x

INV(x) restituisce il reciproco di x (1/x)

ecc.

**operatori binari (due operandi)**

SUB(x,y) restituisce la differenza con segno tra x e y

DIF(x,y) restituisce la differenza assoluta tra x e y

MOL(x,y) restituisce il prodotto tra x e y

DIV(x,y) restituisce il quoziente tra x e y

ecc.

**operatori multipli (più operandi)**

SUM(x,y,z,...) restituisce la somma di x,y,z,...

MEN(x,y,z,...) restituisce la media di x,y,z,...

MED(x,y,z,...) restituisce la mediana di x,y,z,...

STD(x,y,z,...) restituisce la deviazione standard di x,y,z,...

MIN(x,y,z,...) restituisce il minore tra x,y,z,...

MAX(x,y,z,...) restituisce il maggiore tra x,y,z,...

ecc.

=====

**4. LE IMMAGINI DIGITALI**

Le immagini digitali sono costituite da un mosaico di elementi quadrati (pixels) ognuno dei quali possiede un valore corrispondente ad un colore (immagini a colori) o ad un tono di grigio (immagini in grigio) o al bianco o nero (immagini in bianco-nero). Notate che il nostro modo di dire *in bianco e nero*, come dei film non a colori, non è appropriato nel campo dell'informatica. Occorrerebbe dire film *in grigio*. Nel nostro caso, le immagini in bianco-nero sono proprio in bianco e nero e basta.

**Immagini a colori**

Distinguiamo innanzitutto le immagini a colori reali, che interessano a noi, da quelle a palette (tavolozza di colori) e quelle a pseudocolori.

Le immagini a **palette**, per economizzare informazione, utilizzano di volta in volta gamme diverse di colori adatte alle necessità (sarebbe come il pittore che si porta appresso solo i colori che servono al suo quadro). Per quanto una rappresentazione a palette possa apparire cromaticamente perfetta, la diversa codificazione dei colori di ciascuna immagine (ogni immagine ha una diversa palette) non ne consente un uso scientifico.

Le immagini a **pseudocolori** sono essenzialmente immagini monocromatiche, solitamente a toni di grigio, che applicano dei colori diversi (e falsi, da cui il nome) ai diversi livelli di densità.

Es.

bianco	=	bianco
dal bianco al grigio chiaro	=	rosso
dal grigio chiaro al grigio medio	=	blu
dal grigio medio al grigio scuro	=	giallo
dal grigio scuro al nero	=	verde
nero totale	=	nero

La rappresentazione a pseudocolori ha lo scopo di facilitare la distinzione dei diversi toni di grigio. Tipiche immagini a pseudocolori sono le immagini satellitari. Dal punto di vista dell'informazione, le immagini a pseudocolori equivalgono alle immagini a toni di grigio.

Le immagini a **colori reali** (true-color) sono quelle che esprimono in modo omogeneo tutte le tonalità dei colori visibili.

Solitamente un pixel di una immagine a colori contiene tre bytes che rappresentano l'intensità dei colori fondamentali: rosso, verde e blu. Una immagine definita in questo modo è detta immagine RGB (dai nomi dei colori in inglese: red, green e blue). Ad esempio, il pixels con i tre bytes: 234 123 078 avrà una intensità 234 di rosso, 123 di verde e 078 di blu.

Es:

1°	2°	3°	byte	
rosso	verde	blu		<i>colori fondamentali dell'immagine nel sistema RGB</i>
255	255	255	=	bianco
0	0	0	=	nero
255	0	0	=	rosso
0	255	0	=	verde
0	0	255	=	blu
30	30	30	=	grigio scuro
200	200	200	=	grigio più scuro
ecc.				

Quindi una immagine a colori di 100 x 100 pixels conterrà in tutto 100 x 100 x 3 bytes = 30,000 bytes. Poiché 3 bytes contengono in tutto 3 x 8 = 24 bits, queste immagini sono dette anche a 24 bit. Quante colori diversi possiamo ottenere in questo modo?  $255 \times 255 \times 255 = 16,581,375!$  Se non ci bastano dobbiamo ricorrere ad un numero maggiore di bytes (es. aggiungendo un 4° byte di grigio-nero che aumenta la gamma del chiaroscuro, così come certe stampe a colori sono fatte in quadricromia per avere il massimo della qualità). Quasi tutti i software di disegno, ritocco e di analisi consentono di visualizzare i valori dei pixels. Alcuni consentono anche di impostare un colore in forma numerica.

### Immagini a toni di grigio

I pixels delle immagini in grigio contengono un solo byte. Poiché con un byte possiamo avere valori che vanno da 0 a 255, i grigi in tutto saranno 256.

Es.

unico byte

255	=	bianco
0	=	nero
205	=	grigio chiaro
23	=	grigio scuro
ecc.		

Quindi una immagine a toni di grigio di 100 x 100 pixels conterrà in tutto 100 x 100 x 1 bytes = 10,000 bytes.

Poiché 1 byte contiene 8 bits, queste immagini sono dette anche a 8 bit.

Se non ci bastano 256 toni di grigio, soprattutto quando dobbiamo fare delle misure accurate, possiamo utilizzare pixels definiti da 10, 12 o 16 bits, ed allora avremmo a disposizione

$2^{10} = 1024$  toni di grigio

$2^{12} = 4096$  toni di grigio

$2^{16} = 65,536$  toni di grigio

Naturalmente, gli strumenti di imaging a 12 e 16 bit sono anche molto più costosi di quelli a 8 bit.

### Immagini in bianco e nero

Queste immagini appaiono poco frequentemente, e sono prevalentemente utilizzate per creare mappe e maschere, importantissime nella fase di elaborazione delle immagini. Queste immagini utilizzano pochissima informazione. Per ogni pixel abbiamo solo bianco = 1, nero = 0. Quindi un byte è sufficiente per codificare 8 pixels, e una immagine in bianco e nero di 100 x 100 pixels conterrà in tutto 100 x 100 x 1 bytes = 10,000 bytes.

## 5. IMAGE PROCESSING

**Image Processing** comprende tutti quei metodi che attuano una manipolazione dell'immagine e che solitamente hanno come risultato finale un'altra immagine (migliorata o con alcuni oggetti messi più in evidenza). **Image Analysis** comprende tutti quei metodi che estraggono informazioni quantitative dall'immagine e che solitamente hanno come risultato finale una serie di dati.

I metodi di **Image Processing** consistono essenzialmente in:

- ⇒ operazioni che controllano la **LUT** o look-up table: lineari (contrasto e luminosità) e non lineari (parametro gamma)
- ⇒ operazioni che attenuano o esaltano i **dettagli**
- ⇒ operazioni che attenuano o esaltano i **contorni**
- ⇒ operazioni che attenuano o esaltano le **strutture interne**
- ⇒ operazioni che isolano degli oggetti all'interno di una immagine (mediante **estrazione dei contorni** o mediante **soglie [thresholding]** di densità, di colore o di caratteristiche geometriche o texturali)
- ⇒ operazioni logiche (AND, OR, XOR, NOT, NOR, ecc.) che confrontano l'immagine che ci interessa con una immagine binaria o 'maschera'. Il fine è sempre quello di isolare degli oggetti all'interno di una immagine.
- ⇒ operazioni che rimuovono il disturbo o **noise**. Esistono principalmente tre tipi di noise:
  - noise **posizionale** (causato, ad es., dalla sporcizia di una lente; le macchie sono le stesse e nella stessa posizione in tutte le immagini) - il noise posizionale si rimuove mediante la sottrazione del background, cioè sottraendo all'immagine che ci interessa una immagine vuota, presa nelle stesse condizioni (stessa ottica, stesso tempo di esposizione, ecc.)
  - noise **stazionario** (causato, ad es., da una interferenza elettrica; l'immagine mostra delle righe generalmente parallele, su tutta l'area, più o meno oblique e più o meno distanziate) - il noise stazionario si rimuove mediante l'individuazione dell'interferenza (in forma di frequenza anomala) e la sua sottrazione mediante l'analisi (e sintesi) di Fourier.
  - noise **casuale** (causato per lo più quando il rapporto segnale/rumore è basso; questo noise appare nelle immagini come effetto-neve ed è facilmente osservabile nelle riprese con telecamera in condizioni di scarsa illuminazione oppure in TV quando il segnale di antenna non è buono) - il noise casuale si può rimuovere con diversi metodi (paragonabili ai metodi di averaging): filtro medio, mediano, gaussiano, ecc. attuati in modo incondizionato o condizionato al livello di variazione locale. Modo condizionato significa che il filtro è applicato ad un certo pixel solo e se quel pixel ha un valore alterato (in pratica, se il valore di quel pixel è molto diverso dal valore dei pixel adiacenti). In questo modo si evita di applicare il filtro quando non è necessario, conservando la risoluzione originaria dell'immagine. Viceversa, i filtri attuati in modo incondizionato degradano inevitabilmente la risoluzione dell'immagine.

In generale, consideriamo che:

1. ogni operazione va ripetuta per ogni pixel dell'immagine. Quindi se l'immagine ha 1024 righe e 1024 colonne di pixels, in totale 1,048,576 pixels, occorrerà fare 1,048,576 operazioni. L'efficienza del software e la velocità di calcolo dei computer attuali consentono di fare ciò in un attimo.
2. dal punto di vista dei pixels coinvolti, le operazioni possono essere
  - a. **puntuali**: il risultato dell'operazione dipende solo dal valore del pixel sotto esame. Queste sono ad es. tutte le operazioni che modificano la LUT (luminosità, contrasto e parametro gamma). Ad es. il contrasto si può aumentare moltiplicando i valori dei pixels per 2:

2×

...	...	...	...	...
...	20	22	24	...
...	34	36	30	...
...	35	38	45	...
...	...	...	...	...

risultato:

...	...	...	...	...
...	40	44	48	...
...	68	72	60	...
...	70	76	90	...
...	...	...	...	...

- b. **contestuali**: il risultato dell'operazione dipende non solo dal valore del pixel destinatario, ma anche da quello dei pixel adiacenti. Quasi tutti i cosiddetti 'filtri' si basano su operazioni di questo tipo. As es. un filtro mediano 3×3 assegna ad ogni pixels il valore della mediana calcolata sui nove pixels compresi nell'area 3×3 che sta attorno al pixel da ri-valutare.

...	...	...	...	...		...	...	...	...	...
...	20	22	24	...		...	20	22	24	...
...	34	36	30	...	risultato	...	34	34	30	...
...	35	38	45	...	filtro mediano	...	35	38	45	...
...	...	...	...	...		...	...	...	...	...

3. dal punto di vista dell'applicazione, le operazioni possono essere  
a. **globali o non-condizionate**: applicate a tutti i pixels.

- b. **locali o condizionate**: applicate solo a determinati pixels a seconda di una certa condizione 'locale'. Generalmente questa condizione riguarda il valore dei pixels circostanti. Ad es., in caso di noise casuale ('sale e pepe') di forte intensità ma di bassa frequenza (= pochi pixel fortemente alterati) potremmo voler applicare il filtro mediano solo ai pixels alterati. Come individuare i pixels alterati? Calcolando la varianza dei pixels compresi in un'area quadrata (3×3 o 5×5), con il pixels sotto esame al centro. Se la varianza supera una certa soglia assumiamo che il pixel centrale è alterato dal noise. A questo pixel e solo a questo applichiamo il filtro mediano. Questo metodo è il migliore in quanto evita di applicare il filtro mediano a tutti i pixels. Come già detto, il filtro mediano riduce il noise, ma anche degrada la risoluzione effettiva dell'immagine.

Supponiamo di stabilire  $\text{varianza} = 1000$  come soglia per l'applicazione del filtro mediano.

...	...	...	...	...	<b>varianza = 68</b>
...	20	22	24	...	filtro mediano
...	34	36	30	...	non applicato
...	35	38	45	...	
...	...	...	...	...	

...	...	...	...	...	<b>varianza = 2532</b>	...	...	...	...	...
...	20	22	24	...	filtro mediano	...	20	22	24	...
...	34	180	30	...	applicato	...	34	34	30	...
...	35	38	45	...	risultato:	...	35	38	45	...
...	...	...	...	...		...	...	...	...	...

Il pixel centrale aveva un valore (180) troppo 'diverso' per essere genuino, reale. Il filtro mediano lo ha aggiustato.



---

## 5. IMAGE ANALYSIS

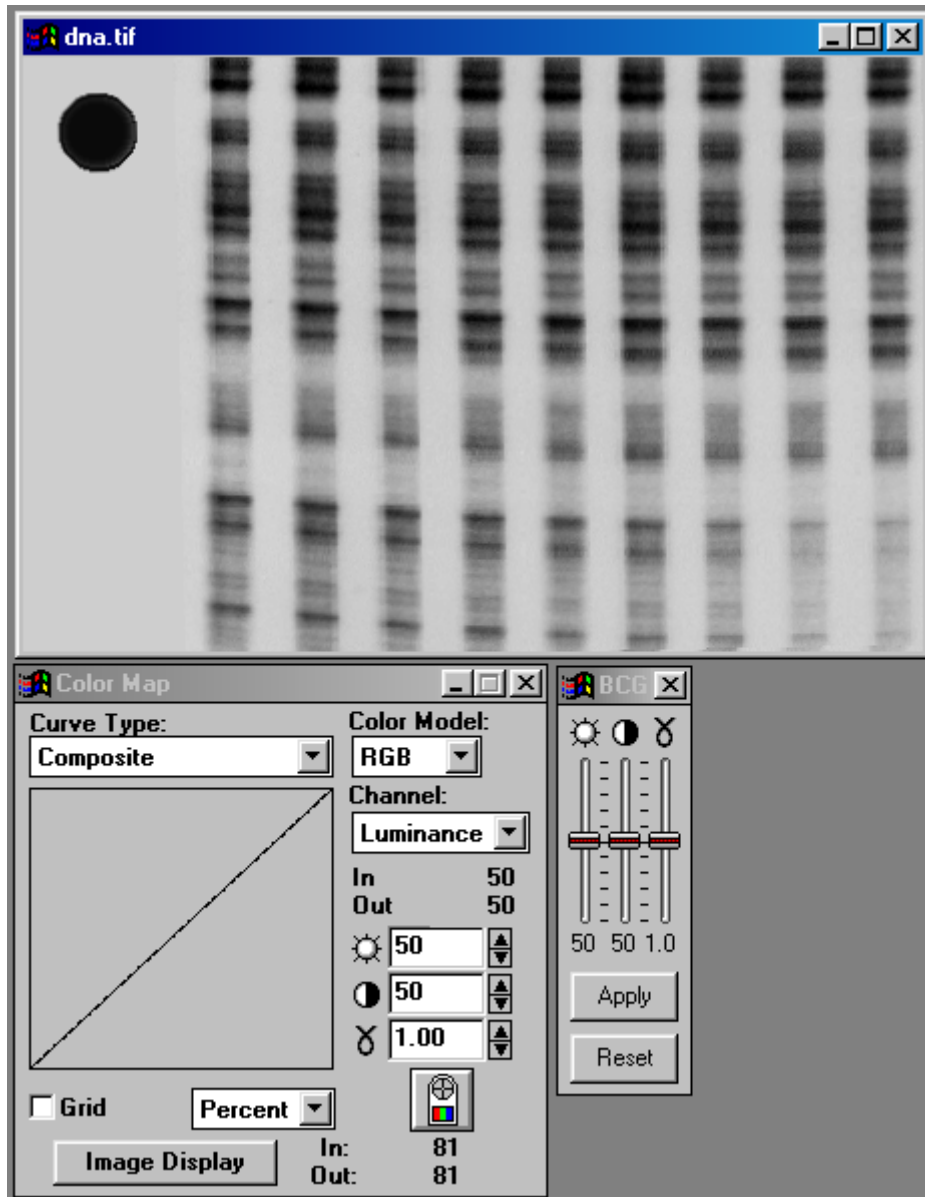
I metodi di *Image Analysis* sono essenzialmente:

- ⇒ metodi che valutano la **grandezza** (area, lunghezza, spessore, perimetro, ecc.)
- ⇒ metodi che valutano la **forma** (ellitticità, rapporto frattale perimetro/area, curvatura media, convessità, ecc.)
- ⇒ metodi che valutano le **densità** (intensità dei valori di grigio)
- ⇒ metodi che valutano la **texture** (variazione locale dei valori di grigio)

Per comprendere meglio il significato di questi metodi, consideriamo una scacchiera formata da caselle bianche e nere:

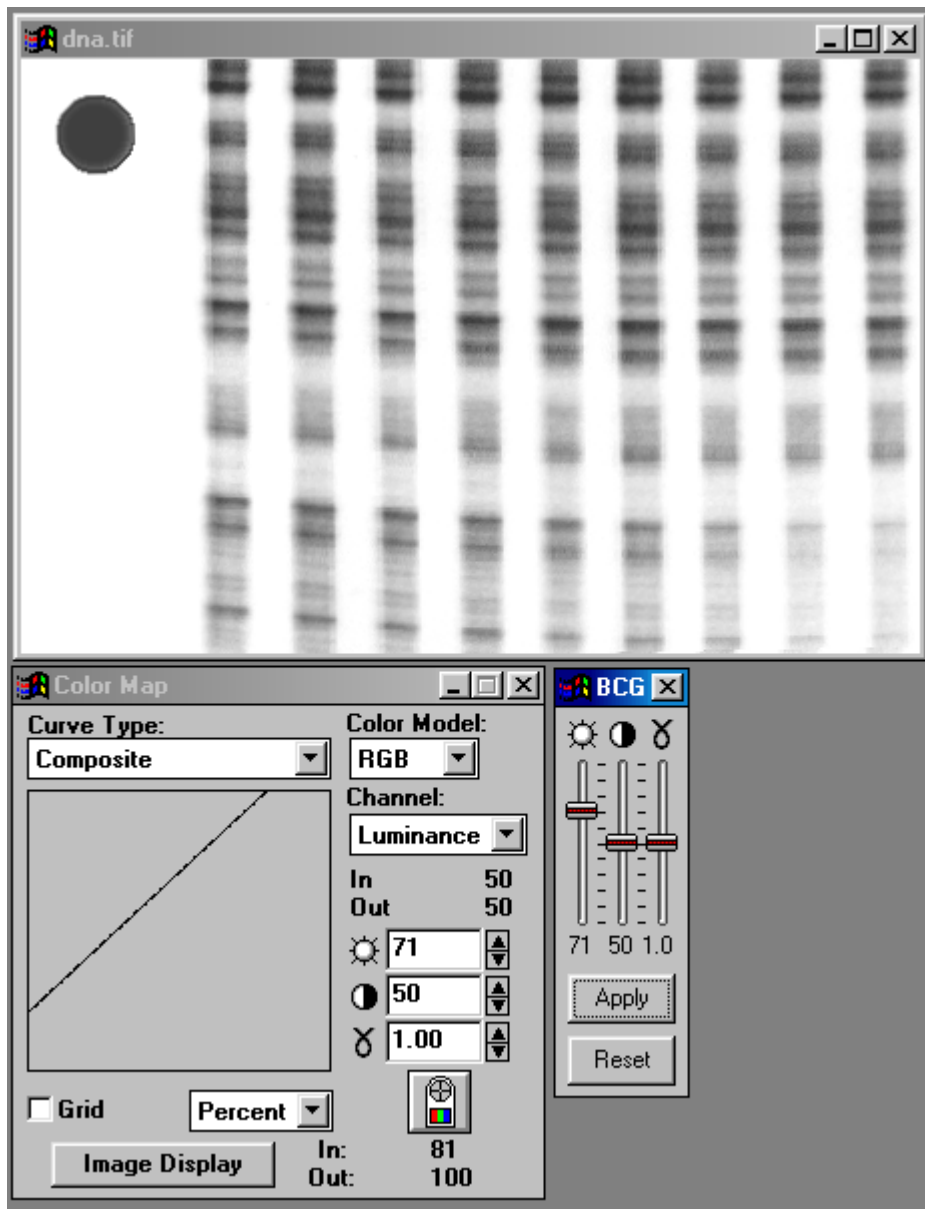
- l'esame della **grandezza** riguarda la misura del lato dei quadretti
- l'esame della **forma** riguarda l'aspetto quadrato delle caselle
- l'esame della **densità** riguarda il fatto che, mediamente, la scacchiera è grigia (127.5), con una alta deviazione standard in quanto i valori variano tra il bianco (255) ed il nero (0). Tuttavia è da notare che la stessa media e deviazione standard si potrebbe avere da una scacchiera con 3x3 grandi caselle o anche 1000x1000 piccolissime caselle. Quindi questo dato della densità (media e deviazione standard globale) è un dato ambiguo. Il problema è risolto solo con l'esame della texture
- l'esame della **texture** generalmente - nell'approccio più elementare - suddivide l'immagine in sottoaree e valuta come i parametri di media e deviazione standard cambiano localmente rispetto alla media e deviazione standard dell'immagine totale. Suddividendo la scacchiera in aree sempre più piccole possiamo osservare che, fino a che queste sottoaree sono assai più grandi della singola casella, media e deviazione standard corrisponderanno alla media e deviazione standard dell'intera scacchiera. Ma non appena l'area di scansione diventa più piccola di una singola casella, ecco che la media da omogenea (127.5) comincerà ad oscillare tra 0 (se casella nera) e 255 (se casella bianca) mentre la deviazione standard, che prima era assai alta, diventerà molto bassa sino a zero (tutto nero entro la casella nera come tutto bianco entro la casella bianca). Quindi, al di là dei parametri statistici globali, solo i parametri statistici locali ci informano di come i dati sono accostati, intessuti o 'regionalizzati'.

## 6. APPLICAZIONI

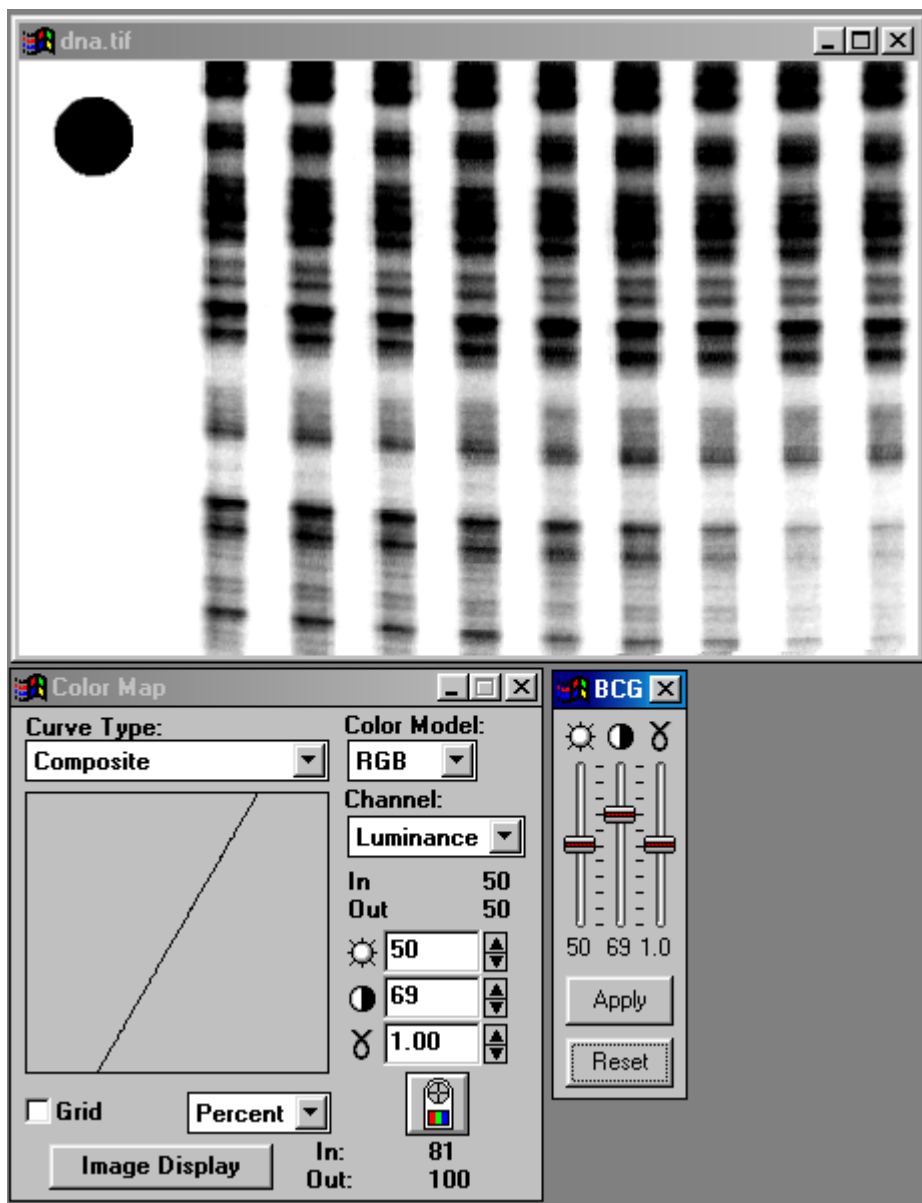


Controllo della LUT (look-up table) con regolazione della luminosità, del contrasto e del parametro gamma.

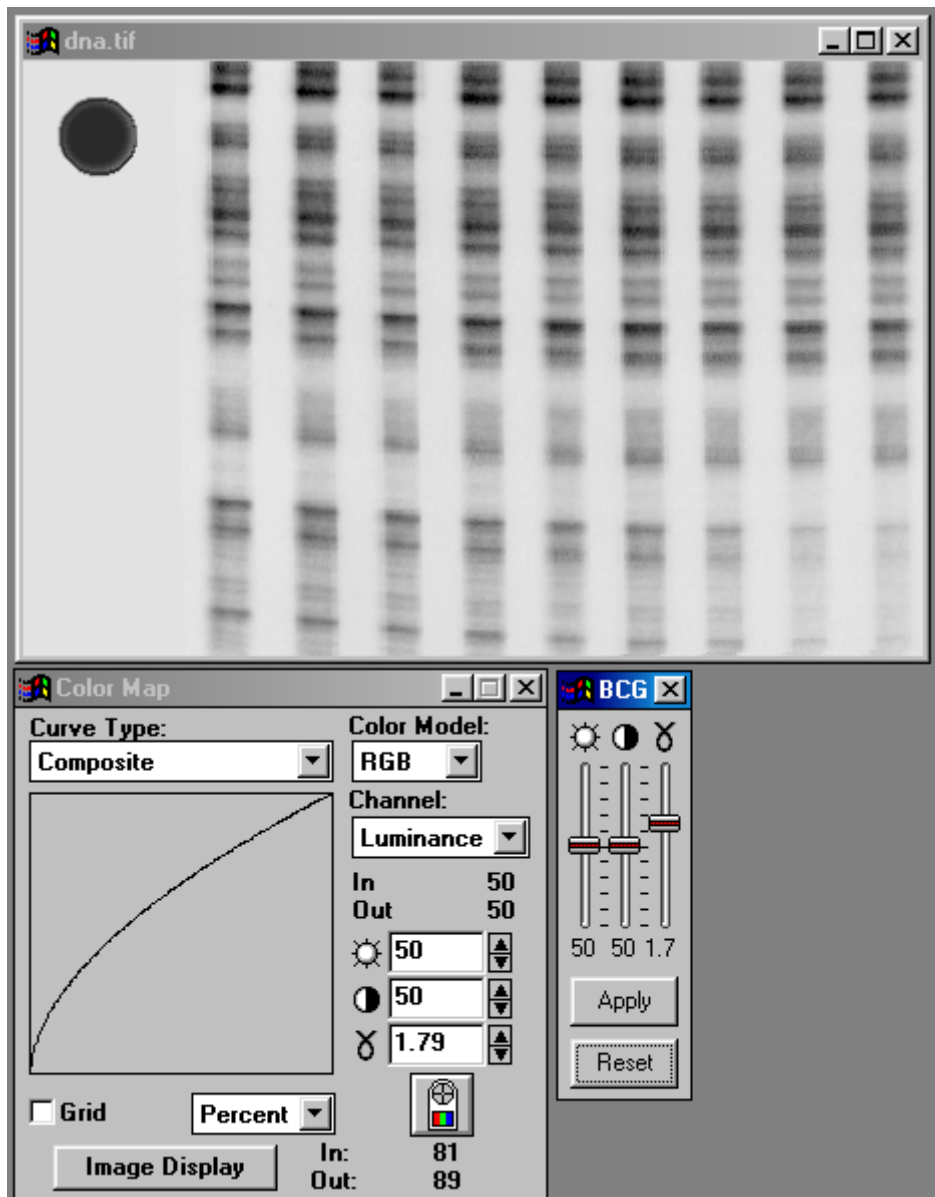
Il grafico che correla i valori di grigio dell'immagine (X) ai toni di grigio visualizzati (Y) mostra una retta a 45°. Quindi i valori di grigio da 0 a 255 dell'immagine corrispondono agli stessi toni di grigio da 0 a 255 visualizzati.



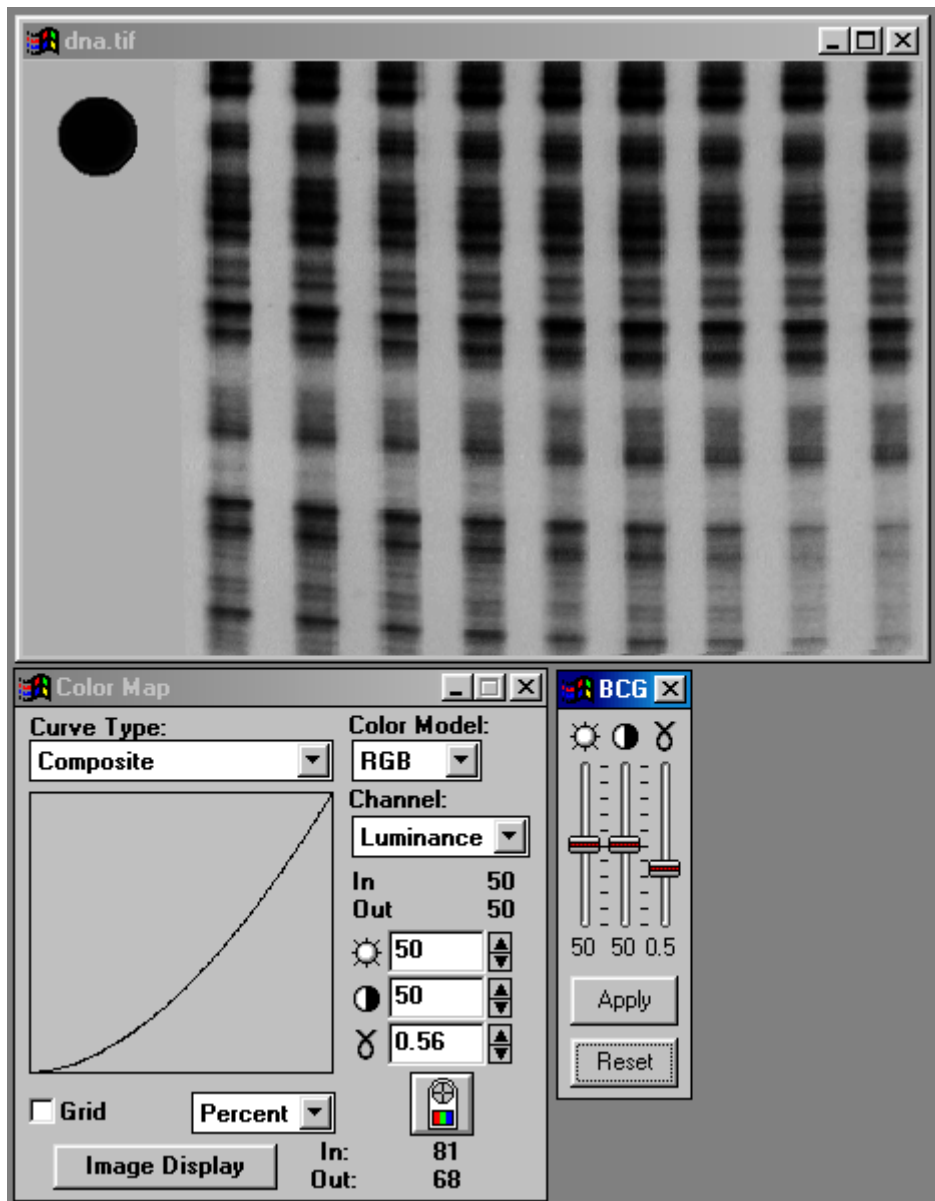
E' stata aumentata la luminosità (vedi 1° cursore spostato verso l'alto). La retta del grafico ha la stessa inclinazione ma è spostata in alto di circa 20 unità. Il che significa che i valori di grigio da 0 a 255 dell'immagine corrispondono a toni di grigio da 20 a 275 visualizzati. Ma poiché il sistema di rappresentazione è a 8 bit (0-255) tutti i valori superiori a 255 sono troncati a 255. Questo è un classico fenomeno di saturazione verso l'alto prodotto dall'aumento della luminosità.



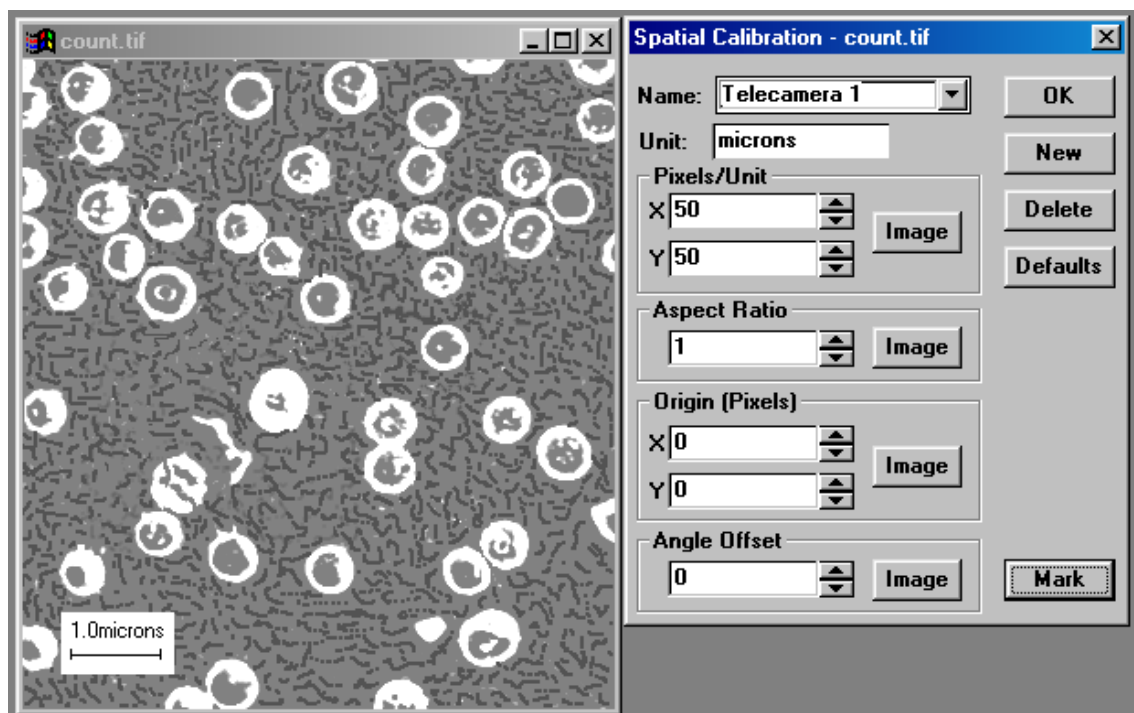
E' stato aumentato il contrasto (vedi 2° cursore spostato verso l'alto). La retta del grafico passa sempre per il centro del grafico, ma è più inclinata. Il che significa che i valori di grigio da 0 a 255 dell'immagine corrispondono ad un intervallo di toni di grigio più esteso, ad es. da -40 a +295. Ma poiché il sistema di rappresentazione è a 8 bit (0-255) tutti i valori superiori a 255 sono troncati a 255, mentre quelli inferiori a 0 sono troncati a 0 (saturazione verso l'alto e verso il basso).



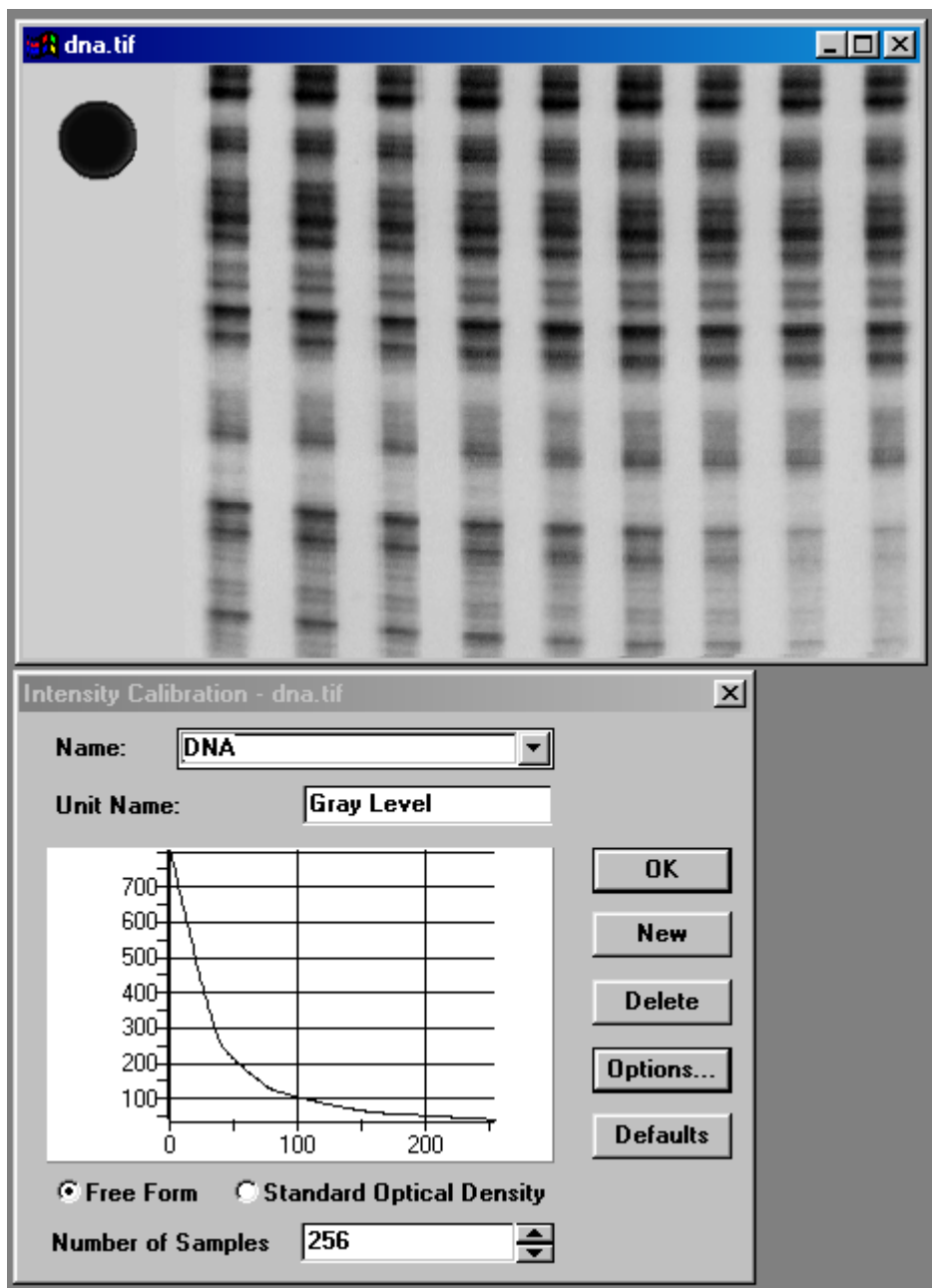
E' stato aumentato il parametro gamma (vedi 3° cursore spostato verso l'alto). La retta del grafico è divenuta una curva molto ripida all'inizio e poco ripida alla fine. Il che significa che i toni di grigio bassi (gli scuri) sono più contrastati (schiariti) mentre non sono granché modificati i toni di grigio alti (i chiari).



E' stato diminuito il parametro gamma (vedi 3° cursore spostato verso il basso). La retta del grafico è divenuta una curva molto ripida solo alla fine. Il che significa che i toni di grigio chiari sono più contrastati (scuriti) mentre non sono granché modificati i toni di grigio bassi (gli scuri).

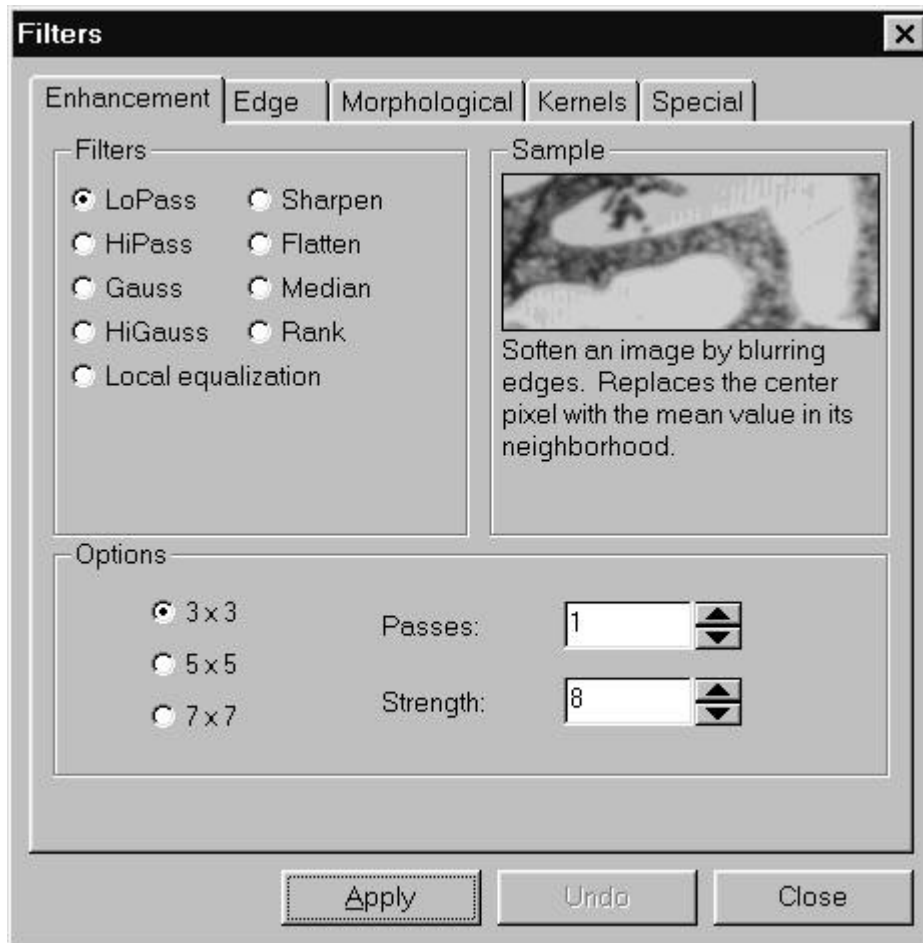


Calibrazione spaziale (50 pixels = 1 micron)



Calibrazione di intensità. La curva riporta le coppie di valori impostate per  
X = valore di grigio  
Y = valore di intensità





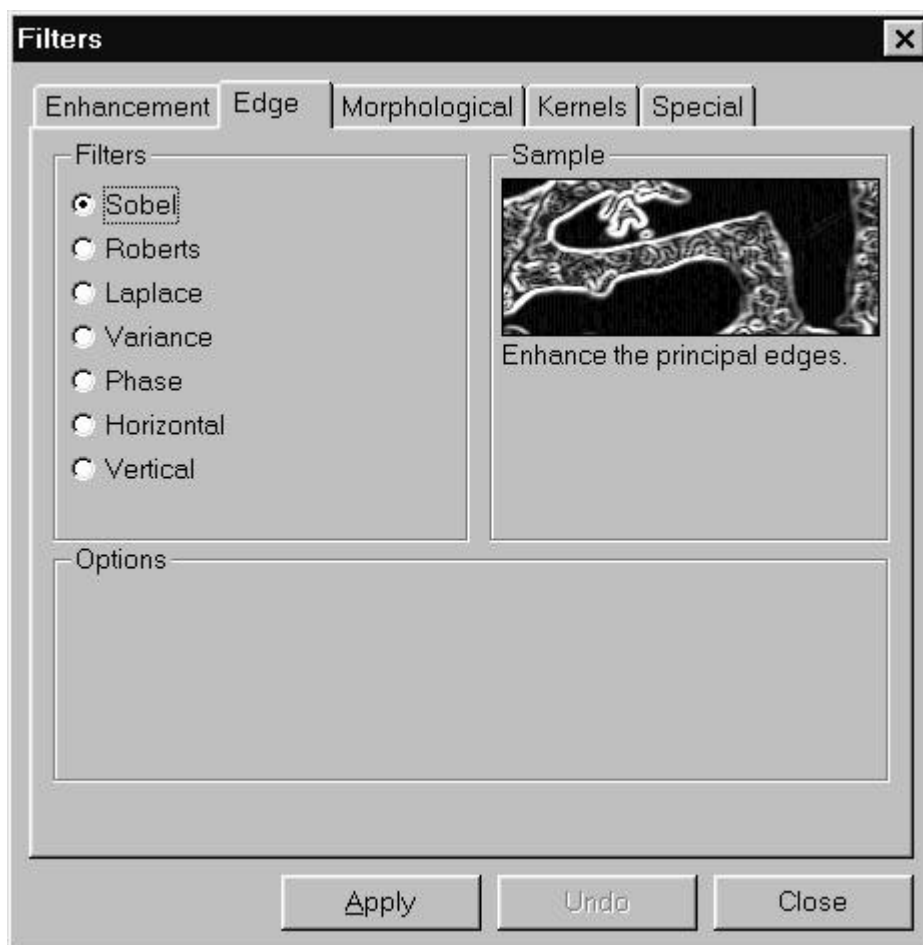
Filtri di **convoluzione** che migliorano tutta l'immagine (enhancement filters, enhancement = miglioramento)

I filtri di **convoluzione** processano una area di pixels (3x3 o 5x5 o 7x7) attorno al pixel centrale, moltiplicando i valori dei pixels con una matrice di coefficienti (valori interi). Questa matrice è detta 'kernel'. I risultati di tali moltiplicazioni sono sommati e poi la somma è divisa per la somma dei coefficienti (procedimento detto media pesata o weighted average). Il risultato è assegnato al pixel centrale. Il filtro low-pass produce la semplice media aritmetica

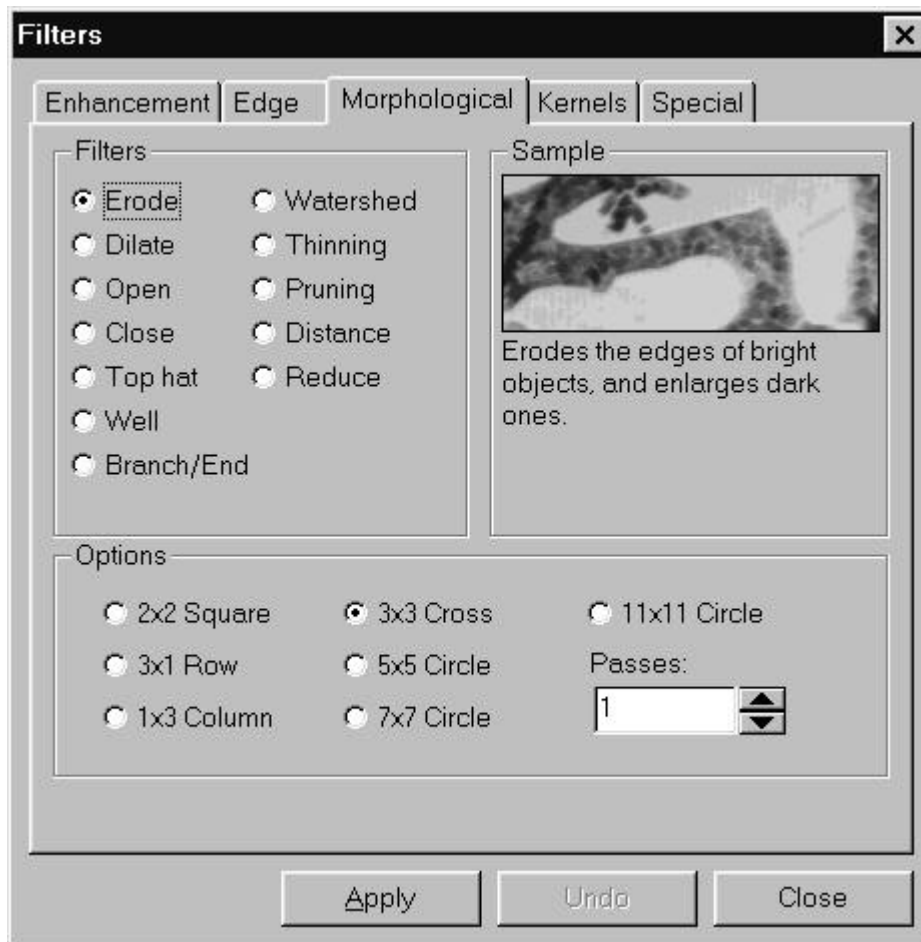
Nota che il processo di convoluzione utilizza sempre i valori originali dei pixels, anche se le righe sopra e le colonne a sinistra il pixel centrale sotto esame sono state già processate. I valori originali sono sempre conservati in memoria anche perché con il comando UNDO (ripristina) il programma deve sempre essere in grado di eliminare l'immagine filtrata e ripristinare l'immagine originaria.

**Passes:** numero di volte che il filtro viene applicato in stretta successione

**Strength:** parametro che proporziona l'effetto del filtro (1 = minimo, 10 = massimo)

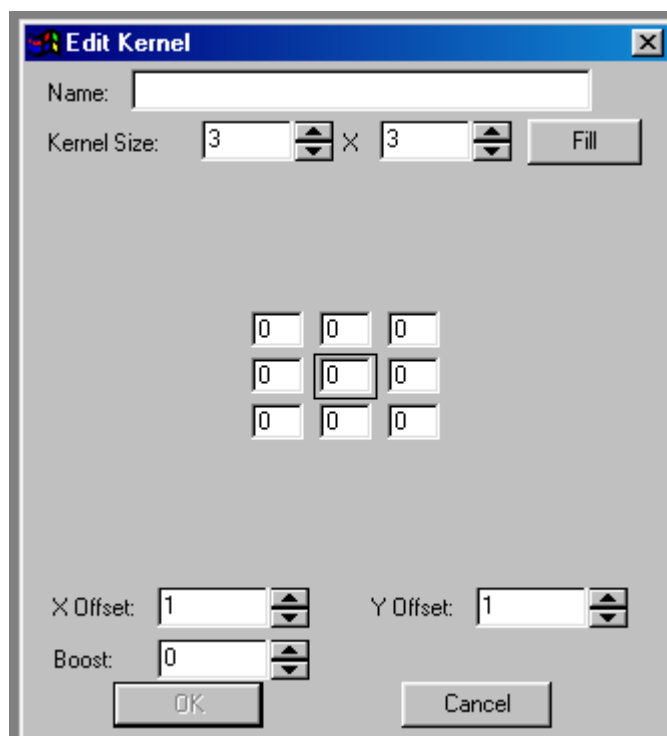
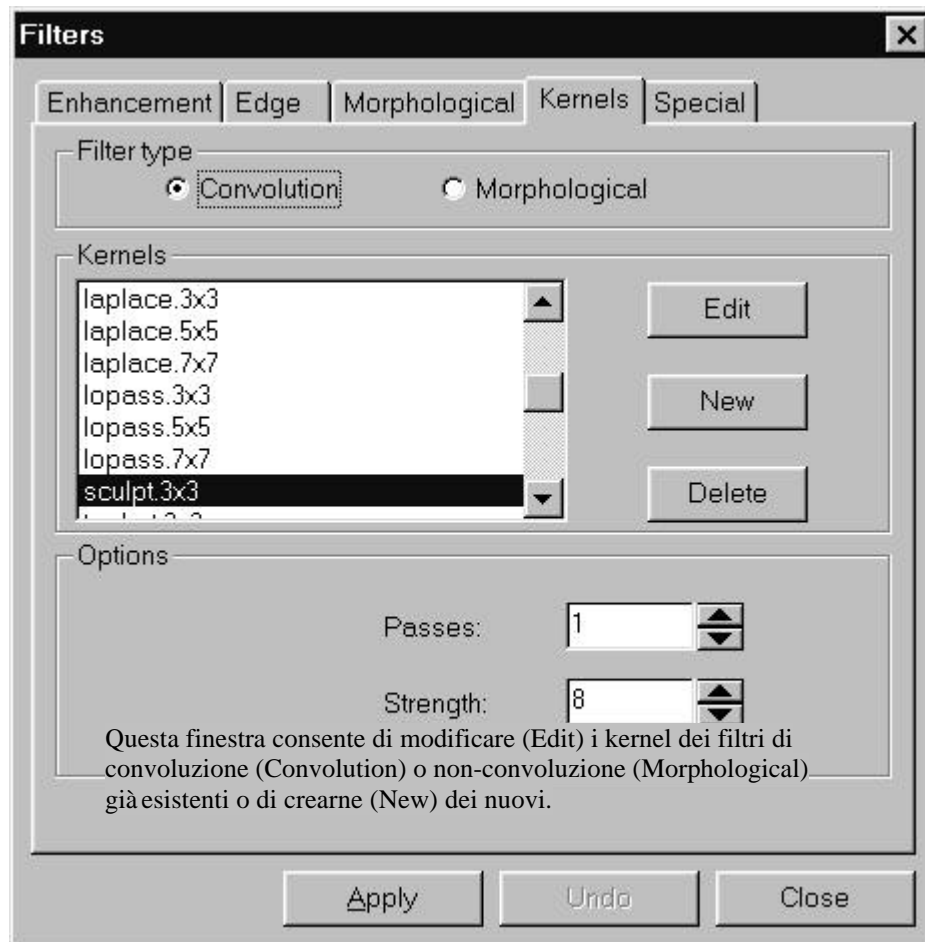


Filtri di convoluzione che migliorano i bordi o contorni e le variazioni texturali (edge filters)



Filtri di **non-convoluzione** che modificano la morfologia (morphological filters)

Anche i filtri di non-convoluzione operano con aree di pixels (2x2 o 3x1 o 1x3 o 3x3 o 5x5 o 7x7) attorno ad un pixel centrale. Comunque, a differenza dei filtri di convoluzione, non vi sono coefficienti che moltiplicano i pixel. I valori dei pixels sono argomento di complesse funzioni statistiche o matematiche. Il risultato poi è assegnato al pixel centrale.



**Edit Kernel**

Name:

Kernel Size:  ×

0	-1	-1	-1	0
-1	-1	-1	-1	-1
-1	-1	21	-1	-1
-1	-1	-1	-1	-1
0	-1	-1	-1	0

X Offset:  Y Offset:

Boost:

**Edit Kernel**

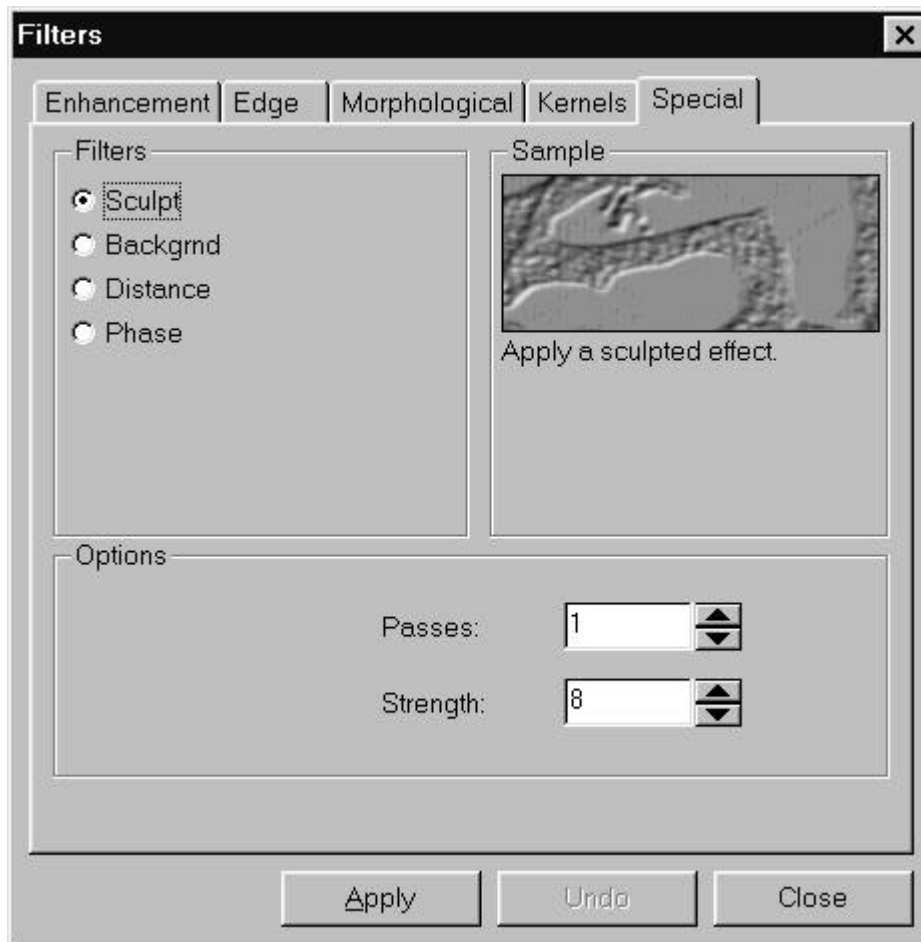
Name:

Kernel Size:  ×

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

X Offset:  Y Offset:

Boost:



Questi sono altri filtri di non convoluzione per operazioni particolari.

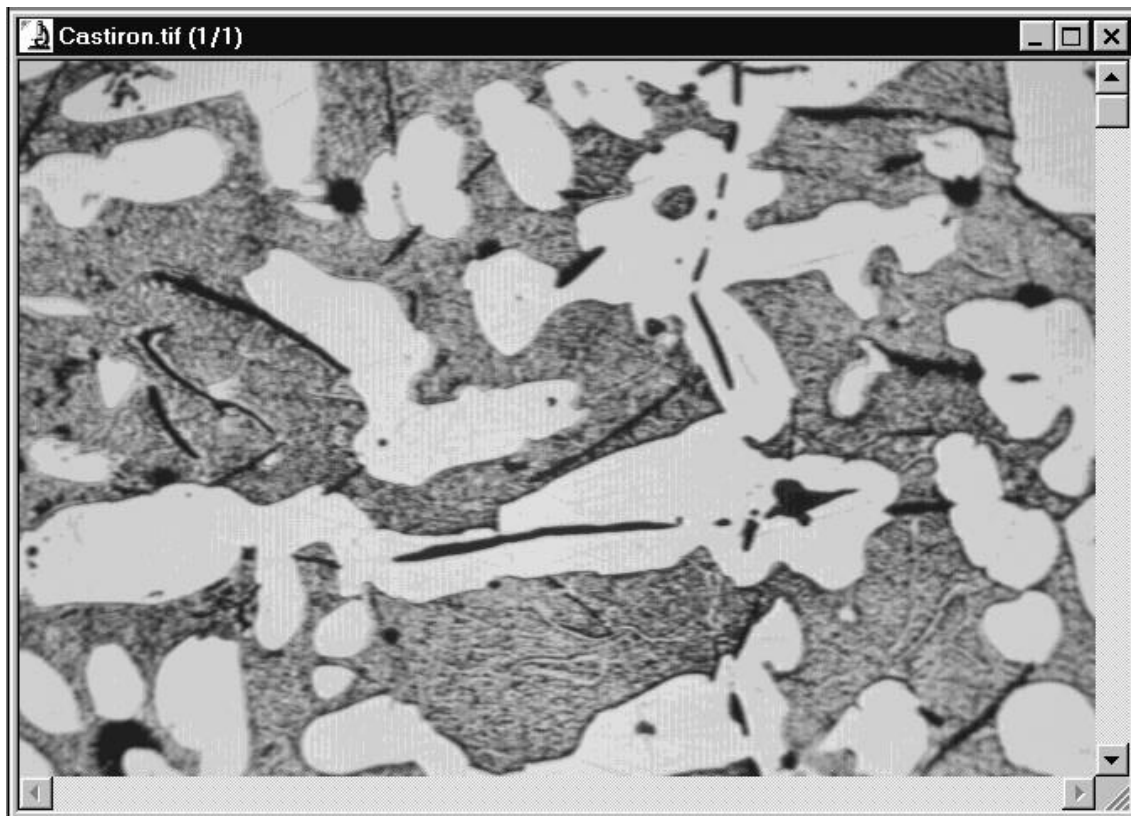
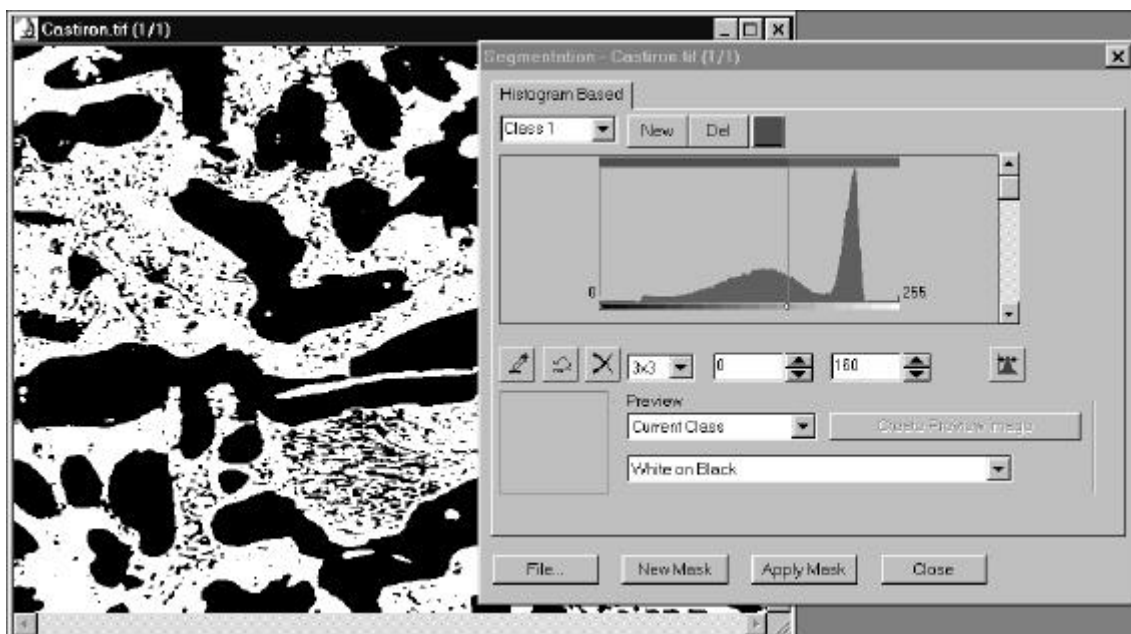
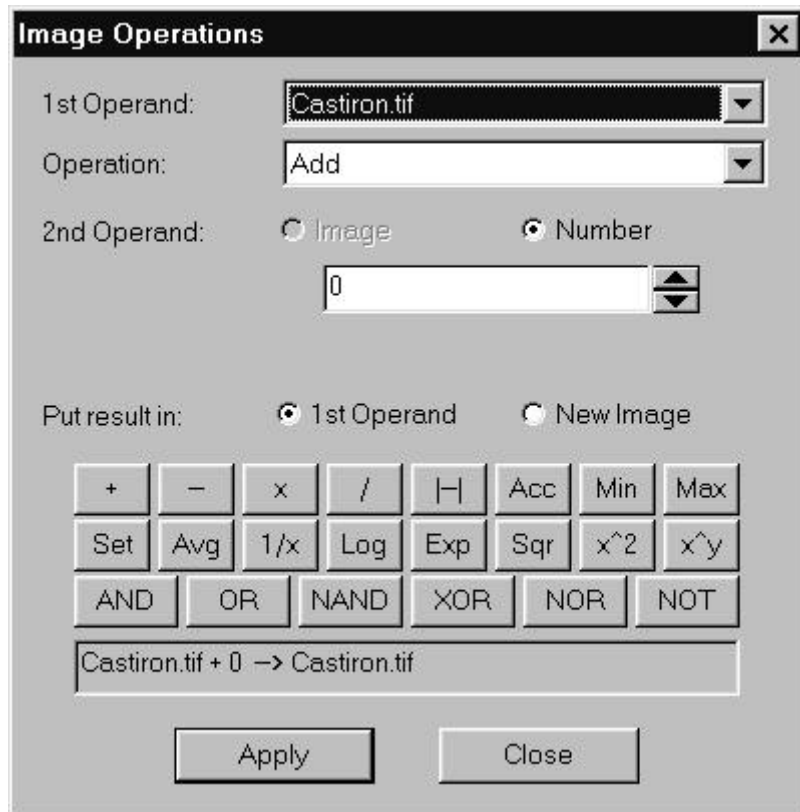


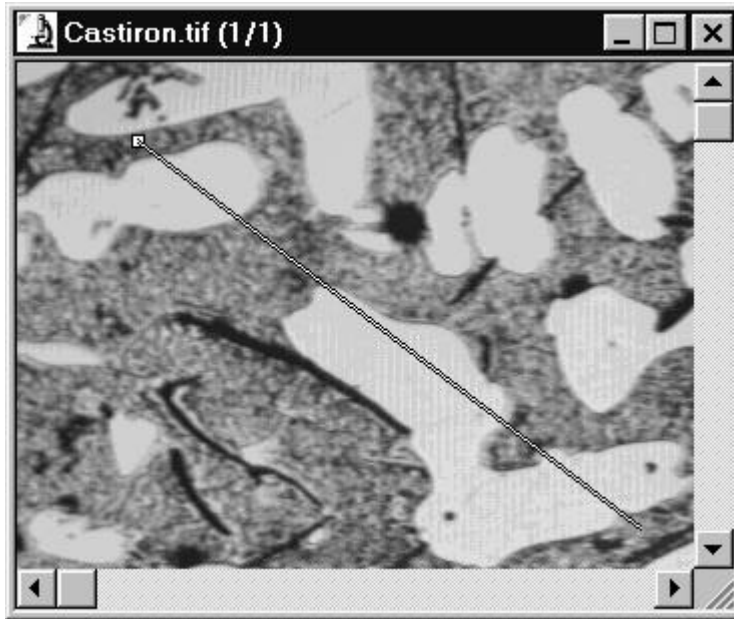
Immagine originale a 256 toni di grigio e, sotto, immagine binaria ottenuta con **soglia** (thresholding). La suddivisione dell'immagine in grigio in tanti oggetti binarizzati (0 = neri e 1 = bianchi) sulla base della soglia applicata è detta **segmentazione**. Una analoga segmentazione si può ottenere sulla base del colore da immagini true color.



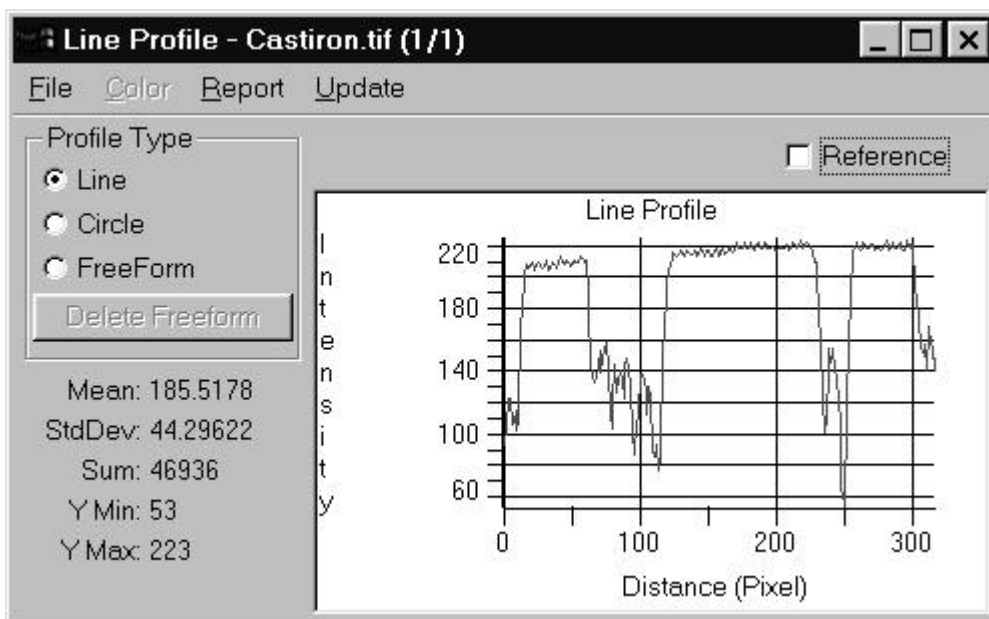


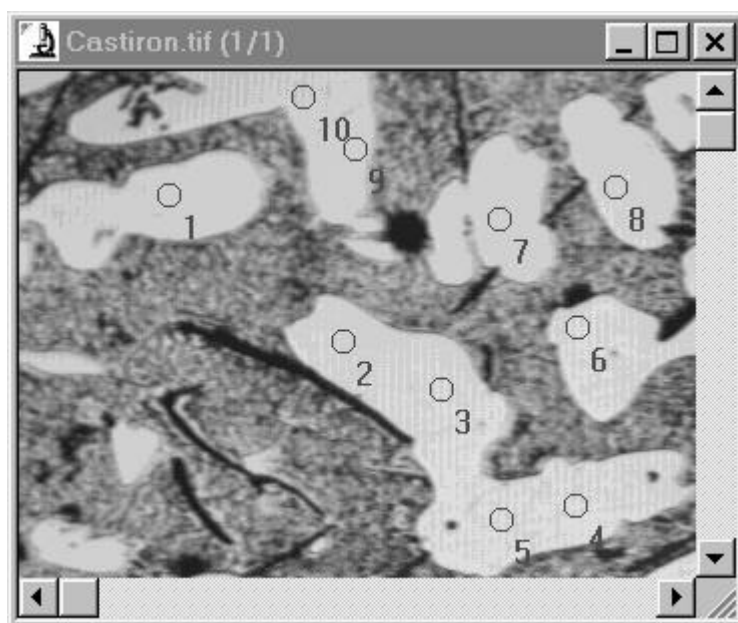
Lista di possibili **operazioni aritmetiche e logiche** (puntuali) che si possono applicare tra una immagine ed un numero (applicato a tutti i pixels) oppure tra due immagini





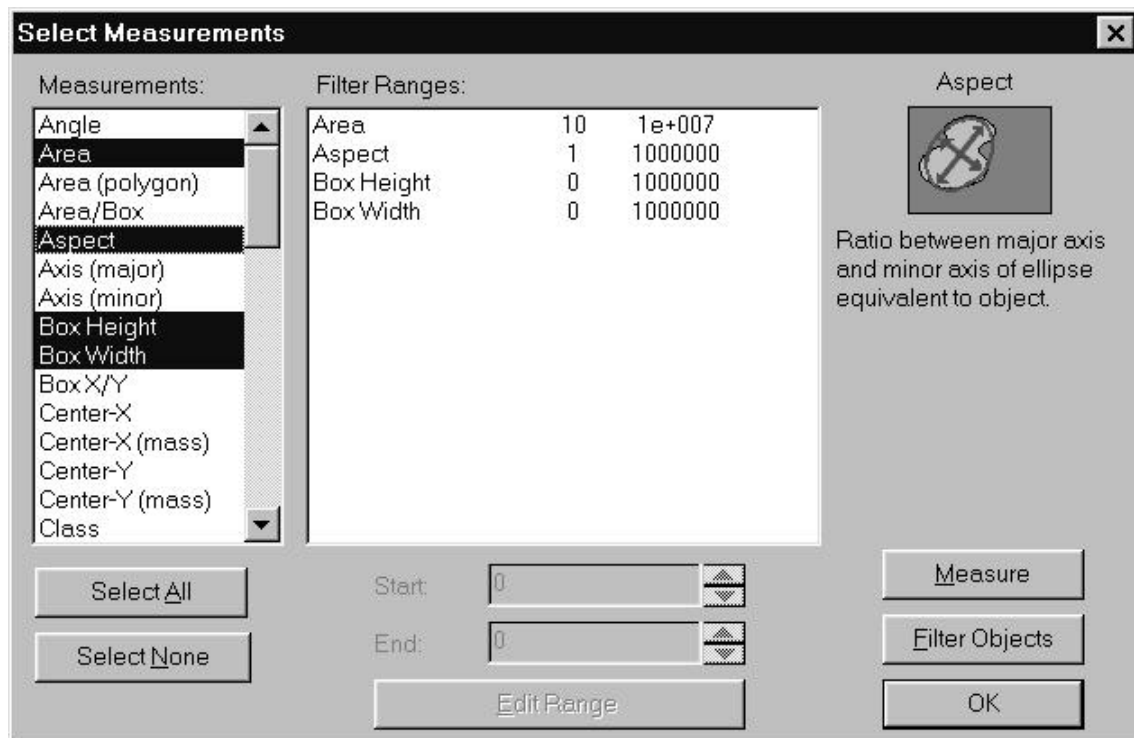
Profilo di intensità dei pixels intercettati da una linea tracciata sull'immagine.  
Nota come le tre parti più alte del grafico (valori tra 200 e 230) corrispondano alle tre zone più chiare dell'immagine intercettate dalla linea.



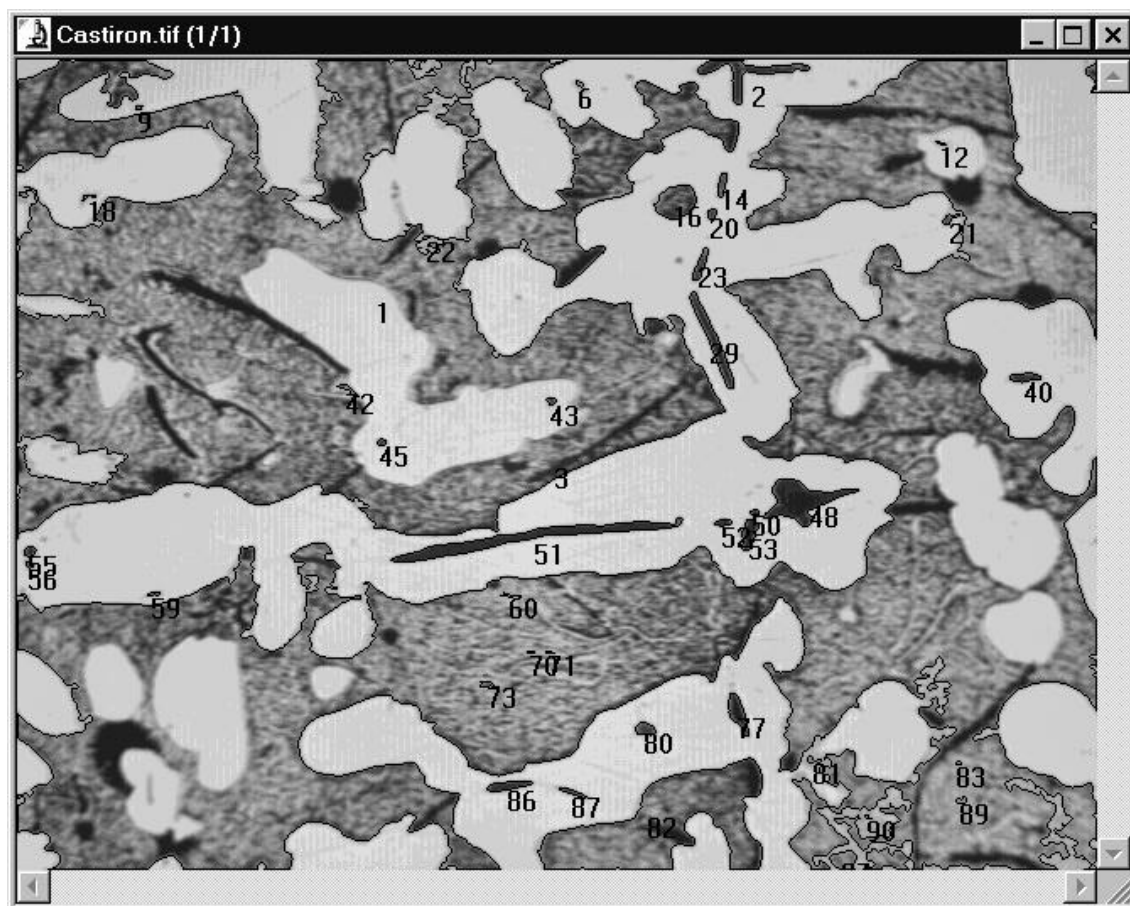


Valori di grigio medi di piccole aree circolari (tag point analysis).

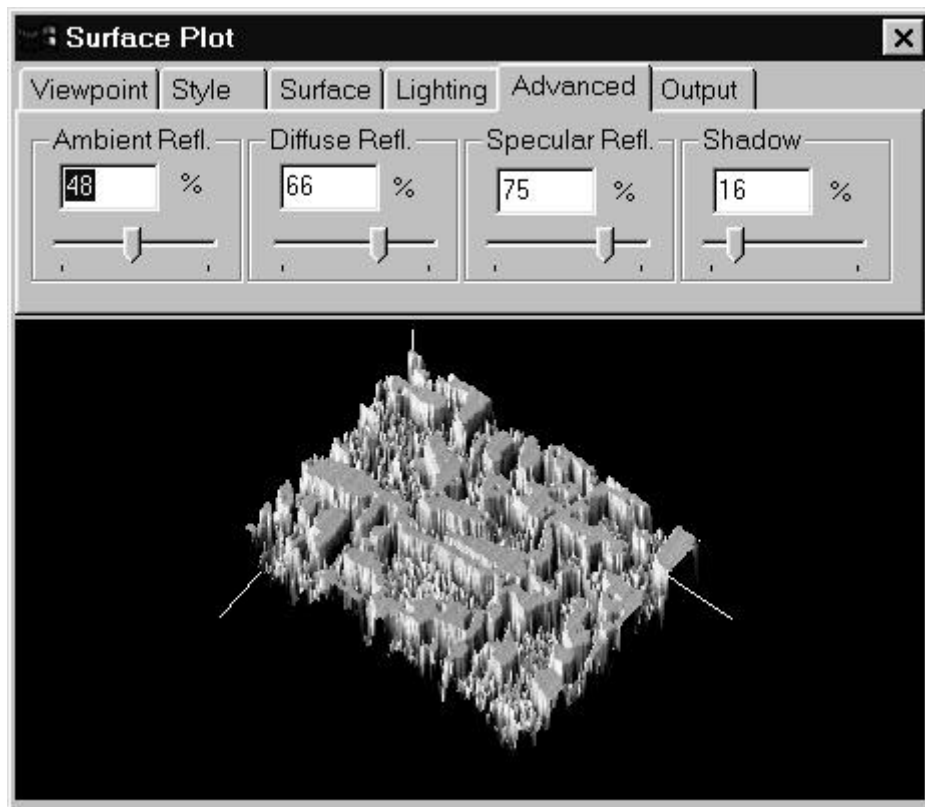
Manual Point Count - Castiron.tif (1/1)		
File View Options Update		
Tag Points	Point	Intensity
Delete Points	○ 1	207.74774
Delete All	○ 2	216.23424
Classes	○ 3	219.38739
Total Count:	○ 4	220.31532
10	○ 5	219.81081
	○ 6	219.96396
	○ 7	215.23424
	○ 8	216.98198
	○ 9	210.75676
	○ 10	205.03604



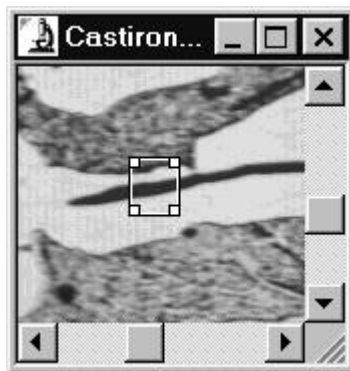
Selezione dei parametri di grandezza e di forma (a sinistra) e definizione dei relativi ranges di accettazione (al centro). La definizione di questi parametri deve precedere l'analisi quantitativa degli oggetti o **blobs** presenti in una immagine.



I blobs (oggetti) misurati sono identificati nell'immagine con etichette numeriche.

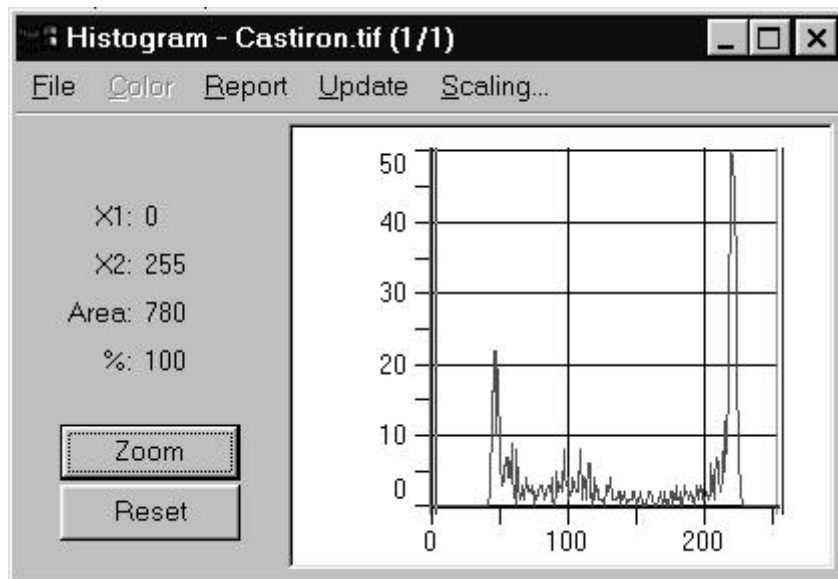


L'immagine precedente visualizzata tridimensionalmente.  
L'altezza (asse z) è costituita dal valore di grigio di ciascun pixels.

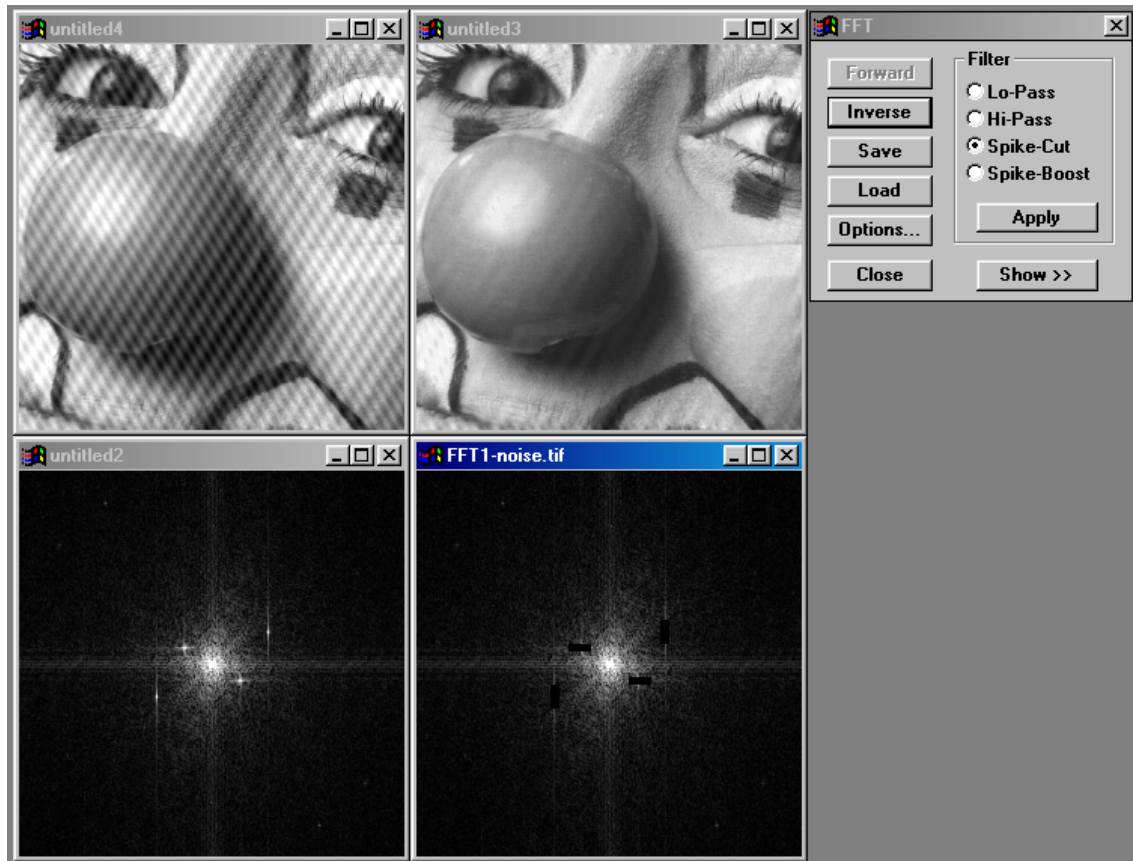


I valori dei pixels presenti in una porzione di immagine o AOI (Area Of Interest) possono essere visualizzati in tabella. Numericamente, l'oggetto sottile e scuro che attraversa l'AOI ha valori bassi (tra 50 e 70) mentre lo sfondo chiaro ha valori alti (tra 180 e 200).

Bitmap Analysis									
File Options Update									
Pixel	261	262	263	264	265	266	267	268	
280	63	97	136	177	206	221	223	220	
281	168	185	198	209	216	223	223	220	
282	211	217	217	219	220	223	222	219	
283	219	222	220	219	219	222	219	215	
284	216	220	217	215	213	211	205	195	
285	181	185	179	170	157	144	125	106	
286	85	88	82	78	73	70	63	58	
287	54	56	50	49	48	50	49	48	
288	46	49	45	46	46	48	46	45	
289	44	47	44	44	44	46	46	46	
290	44	48	45	46	46	48	48	48	
291	46	49	47	48	51	59	71	88	
292	65	80	94	115	137	160	175	187	
293	168	180	186	194	202	210	213	214	
294	209	214	213	214	216	220	220	219	

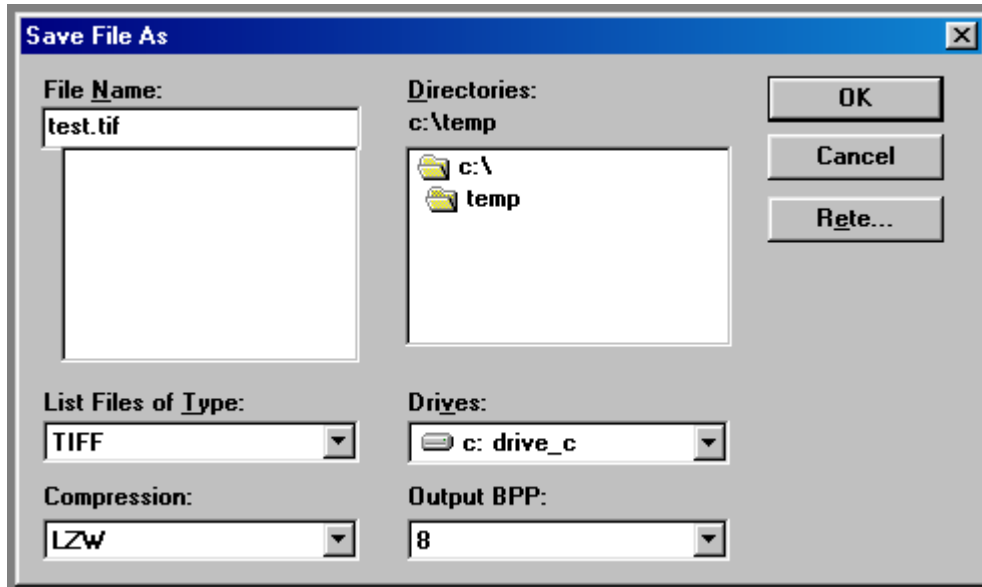


L'istogramma della stessa porzione di immagine. Il picco a sinistra rappresenta l'oggetto sottile scuro mentre il picco a destra rappresenta lo sfondo.



Il noise stazionario (o periodico) può essere visualizzato e poi rimosso analizzando lo spettro bidimensionale delle frequenze. L'immagine in base a sinistra è la cosiddetta **trasformata di Fourier** dell'immagine originale in alto a sinistra: si tratta della stessa immagine, con tutti i suoi dettagli, ma descritta non più nel cosiddetto **dominio dello spazio** bensì nel **dominio delle frequenze**. I quattro puntini chiari separati dalla massa centrale chiara rappresentano le frequenze anomale che provocano le fastidiose striscie. Dopo aver eliminato tali frequenze (metodo spike-cut: immagine in basso a destra), l'immagine può essere risintetizzata (in alto a destra) abbastanza pulita.





Finestra di dialogo che consente di salvare una immagine in diversi formati grafici.

I formati grafici (TIF, BMP, PCX, GIF, TGA, JPG, ecc.ecc.) differiscono essenzialmente per

- A. La ricchezza di informazioni che accompagna l'immagine. Ad es., se l'immagine è multipagina, come i fax, i valori della LUT associata, il nome dell'autore dell'immagine, un possibile commento sul contenuto, ecc. Uno dei formati più 'ricchi' in tal senso è il formato TIF.
- B. Il tipo di compressione. Spesso c'è necessità di comprimere le immagini per poterle più facilmente archiviare e trasportare. La compressione può essere con o senza perdita di dati (**lossy** or **no-lossy compression**). Ovviamente la compressione senza perdita non potrà mai ottenere riduzioni molto piccole. Di solito, con la compressione no-lossy si ottiene una riduzione della dimensione dei files attorno al 50%. Attenzione tuttavia che in certe immagini particolarmente complesse tale compressione produce addirittura un aumento delle dimensioni del file! Un protocollo di compressione no-lossy per immagini, molto utilizzato, è il LZW (Lempel-Zif encoding) applicato al formato TIF. Un altro è il RLE (Run Length Encoding) applicato al formato BMP. Altri protocolli no-lossy, più generici, utilizzati per qualsiasi tipo di file, sono ZIP, ARJ e RAR.
- C. In caso di lossy-compression, sono importanti l'efficienza e la flessibilità di compressione. Il formato JPG è in tal senso molto apprezzato, in quanto consente di dosare il grado di compressione in relazione alla destinazione dell'immagine (molto compressa per uso Internet; meno compressa per uso CD; pochissimo compressa = alta qualità per uso stampa). Un altro importante formato di compressione con perdita è il CCITT (compressione fax Group 3 e Group 4) applicabile però solo a immagini TIF binarie (non in scala di grigi). E' quindi applicabile a tutti i testi ed ai grafici. Vi sono poi vari altri protocolli di compressione che vantano diverse proprietà. In applicazioni di acquisizione continua in real time (tipo video digitale), una importante caratteristica è non solo il livello qualitativo ma anche la velocità con cui avviene il processo di compressione. Nel caso del video digitale esistono tecniche di compressione molto veloci che valutano le sole differenze tra immagini successive (es. si ha un gran risparmio se si considera solo un oggetto che si sposta davanti ad una scena fissa). Viceversa, nel campo delle immagini statiche vi sono metodi di compressione molto efficienti (es. compressione frattale) in grado di ridurre enormemente le dimensioni dei

files, ma che comportano tempi di elaborazione assai lunghi. Comunque tutto il campo dei metodi di compressione è in costante evoluzione.

A titolo di esempio, queste sono le dimensioni dei files ottenuti salvando sempre la stessa immagine in diversi formati e con diversi livelli di compressione.

DNA (8bit gray - TIF not compressed).TIF	131 KB
DNA (8bit gray - TIF LZW-compressed).TIF	68 KB
DNA (8bit gray - BMP not compressed).B...	133 KB
DNA (8bit gray - PCX).PCX	105 KB
DNA (8bit gray - GIF).GIF	83 KB
DNA (8bit gray - TGA).TGA	131 KB
DNA (8bit gray - JPG alta qualità).JPG	51 KB
DNA (8bit gray - JPG media qualità).JPG	14 KB
DNA (8bit gray - JPG bassa qualità).JPG	7 KB