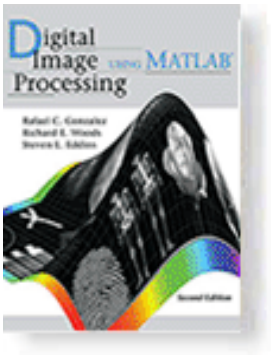
DIGITALNA OBRADA I ANALIZA SLIKE

(NASTAVNI MATERIJAL predavanja/vježbe)

Akademska godina 2016/2017

dr.sc. Barbara Džaja

Napomena: Ovaj nastavni materijal obuhvaća predavanja i laboratorijske vježbe iz kolegija ***Digitalna obrada i analiza slike***. U njoj **nisu** i svi zadaci koji su rješavani na laboratorijima, kao ni svi studentski grupni projekti. Materijal je prvenstveno napisan kako bi se olakšalo pripremanje studenata za kolokvij i ispit. Nastavni materijal koristi nužan paket materijala za podršku u fakultetskoj nastavi **DIPUM 2 / e**, od kojih su najvažniji **P**-kodovi **DIPUM Toolbox 2**, te originalne slike za obradu korištene u knjizi:



Digital Image Processing Using MATLAB 2nd Ed.

Gonzalez, Woods, and Eddins

© 2009

1. UVOD – Što je to digitalna obrada i analiza slike?

Ne postoji striktna granica između obrade i analize slike.

Digitalna obrada slike (Digital Image Processing) je disciplina u kojoj je ulaz i izlaz nekog procesa slika. (Postavlja se pitanje: a što onda s prosječnim intezitetom slike? – to je broj)

- ⊡ Razumijevanje slike može se definirati kao proces koji iz slike izvlači nekakvu informaciju o slici, a u svrhu razumijevanja slike.
- ⊡ Karakteristika je da je samo na ulazu slika.
- ⊡ Umjetna inteligencija (AI) – gdje spada?
- ⊡ Cilj AI: imitiranje ljudske inteligencije uz pomoć računalnog vida.

Računalni vid je računalno imitiranje ljudskog vida, uključujući učenje i donošenje odluka obzirom na vizualni ulaz.

Primjena obrade slike:

- ⊡ Medicinske primjene
  - Radiologija (RTG, PET-nuklearne, CT, MR, UVZ : obrada i analiza)
  - Poboljšanje slika, rekonstrukcija iz projekcija (CT), analiza (pronalaženje organa, zloćudnih bolesti), registracija slika
- ⊡ Daljinska istraživanja (*remote sensing*): snimanja satelita, iz aviona...
  - Geologija (nalazišta nafte), poljoprivreda, meteorologija, ekologija, vojne i policijske primjene
- ⊡ Industrijske primjene

- Nadzor i mjerenje proizvodnih procesa
- Vizualna kontrola kvalitete
- Upravljanje proizvodnih procesa
- Inteligentni strojevi u proizvodnji
- Autonomna vozila

⊡ Komunikacije i arhiviranje

- Prikaz slike/videa uz pomoc bitova
- Kompresija sike/videa (sa i bez gubitaka)
- Prijenos slike (video konferencije, internet)
- Baze slikovnih podataka

⊡ Potrošaćka elektronika

- Digitalne foto/video kamere (detekcija lica, kompresija slika)
- Skeneri, video telefoni, projektori, mobilni telefoni, dlanovnici, prijenosna računala

⊡ Nuklearna fizika

⊡ Biologija

⊡ Mikroskopija

⊡ Radar, sonar

⊡ Vojne primjene

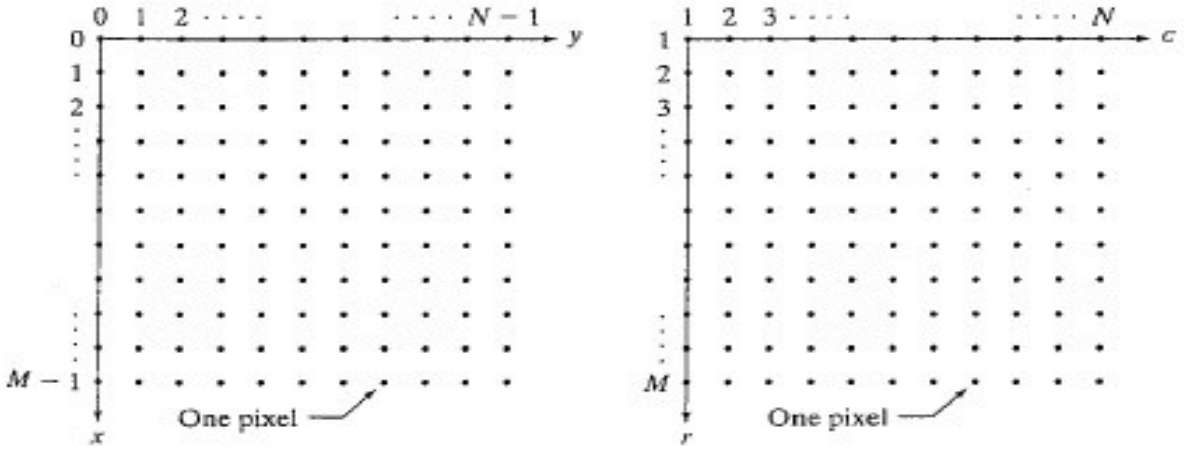
⊡ Policijske primjene

⊡ Astronomija

DIGITALNA PREZENTACIJA SLIKE

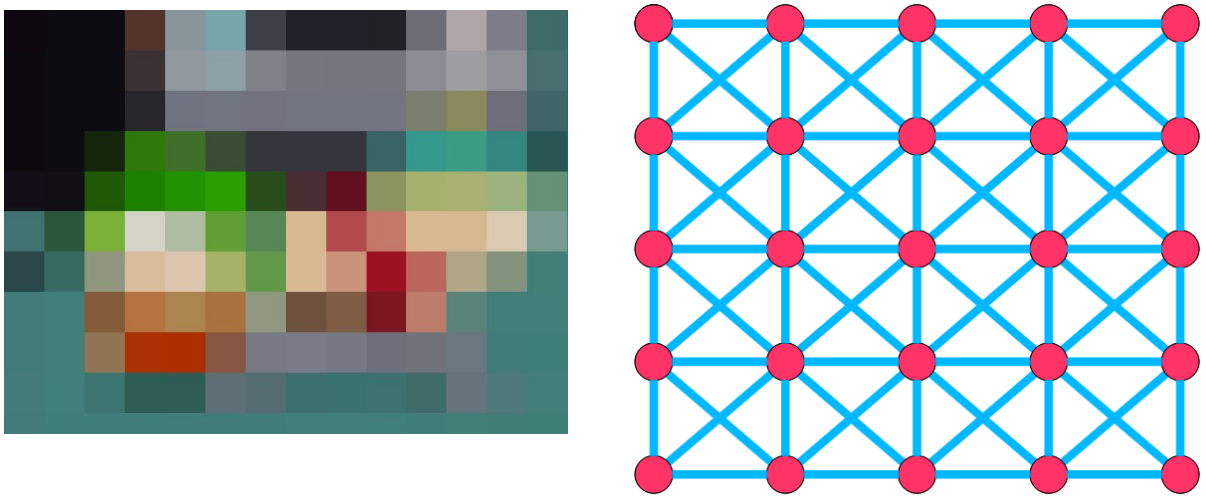
Slika je 2D numerički niz, odnosno 2D funkcija *f(x,y)* , gdje su *x* i *y* **planarne koordinate**, dok je *f* na bilo kojoj poziciji koordinata *(x,y)* **intezitet** slike u toj točki.

- ⊡ Termin: sive slike, korisit se kod monokromatskih slika
- ⊡ Slike u boji se sastoje od tri kombinacije 2D nizova
- ⊡ Slike: matrice veličine (*M x N*) sa realnim vrijednostima brojeva (rezultat uzorkovanja i kvantizacije)



SLIKE KAO MATRICE

Uzorkovanje slike je digitaliziranje vrijednosti koordinata, dok je kvantizacija digitaliziranje vrijednosti amplituda. Za sliku možemo reći da je digitalna kada su *x*, *y* i *f* konačne diskretne vrijednosti.



Slika 1. Prikazivanje slike preko piksela.

KLASE PODATAKA

Tablica 1. Kalse podataka u Matlabu.

Name	Description
`double`	Double-precision, floating-point numbers in the approximate range -10^{308} to 10^{308} (8 bytes per element).
`uint8`	Unsigned 8-bit integers in the range [0, 255] (1 byte per element).
`uint16`	Unsigned 16-bit integers in the range [0, 65535] (2 bytes per element).
`uint32`	Unsigned 32-bit integers in the range [0, 4294967295] (4 bytes per element).
`int8`	Signed 8-bit integers in the range [−128, 127] (1 byte per element).
`int16`	Signed 16-bit integers in the range [−32768, 32767] (2 bytes per element).
`int32`	Signed 32-bit integers in the range [−2147483648, 2147483647] (4 bytes per element).
`single`	Single-precision floating-point numbers with values in the approximate range -10^{38} to 10^{38} (4 bytes per element).
`char`	Characters (2 bytes per element).
`logical`	Values are 0 or 1 (1 byte per element).

TIPOVI SLIKA

Slike mogu biti intezitetne, binarne, indeksirane i RGB.

- ⊡ Intezitetne slike
  - Elementi matrice su skalirane vrijednosti inteziteta
- ⊡ Binarne slike
  - logički niz nula i jedinica
  - **B = logical (A);** →*sve nule postaju logičke nule, a vrijednosti različite od nule su logičke jedinice*
  - **islogical(C)** → *provjera je li matrica* **C** *logička matrica*
- ⊡ Indeksirane slike
- ⊡ RGB slike

Slika 2. RGB slika (lijevo) i binarna slika (desno).

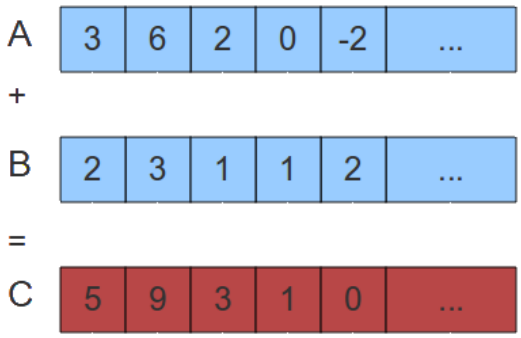
KONVERZIJA IZMEĐU KLASA PODATAKA I TIPOVA SLIKA

Tablica 2: Konverzija između klasa podataka i tipova slika u Matlab progamu.

Konverzija iz - u	Matlab naredba:
Intensity/indexed/RGB format – binary format	dither()
Intensity format – indexed format	gray2ind()
Indexed format to intensity format	ind2gray()
Indexed format – RGB format	ind2rgb()
Matricu u intensity format putem skaliranja	mat2gray()
RGB format – intensity format	rgb2gray()
RGB format – indexed format	rgb2ind()

Indeksiranje nizova, vektora i matrica

Vektori se zbrajaju klasičnom vektorskom algebrom, na način:



Matrice se mogu prikazati i kao vektori i kao sekvence vektora redaka odvojenih točka-zarezom.

A(2,3) = 10

A(:,3) = ?

A(3,:) = ?

A(1:2,1:3) = ?

A(end, end) = ?

B = A(:) → pretvaranje matrice u vektor

d = ndims(A) -> funkcija koja vraća broj dimenzija nekog niza

1 16	5 2	9 3	13 13
2 5	6 11	10 10	14 8
3 9	7 7	11 6	15 12
4 4	8 14	12 15	16 1

A

Slika 3. Matrica A.

VAŽNI STANDARDNI NIZOVI

U Matlabu se važniji nizovi geneririaju gotovim funkcijama. Ovo su neke od najvažnijih:

- **zeros(M,N)**
- **ones(M,N)**
- **true(M,N)**
- **false(M,N)**
- **magic(M)**
- **rand(M,N)**
- **randn(M,N)**
- **magic(3)**

ans =

8 1 6

3 5 7

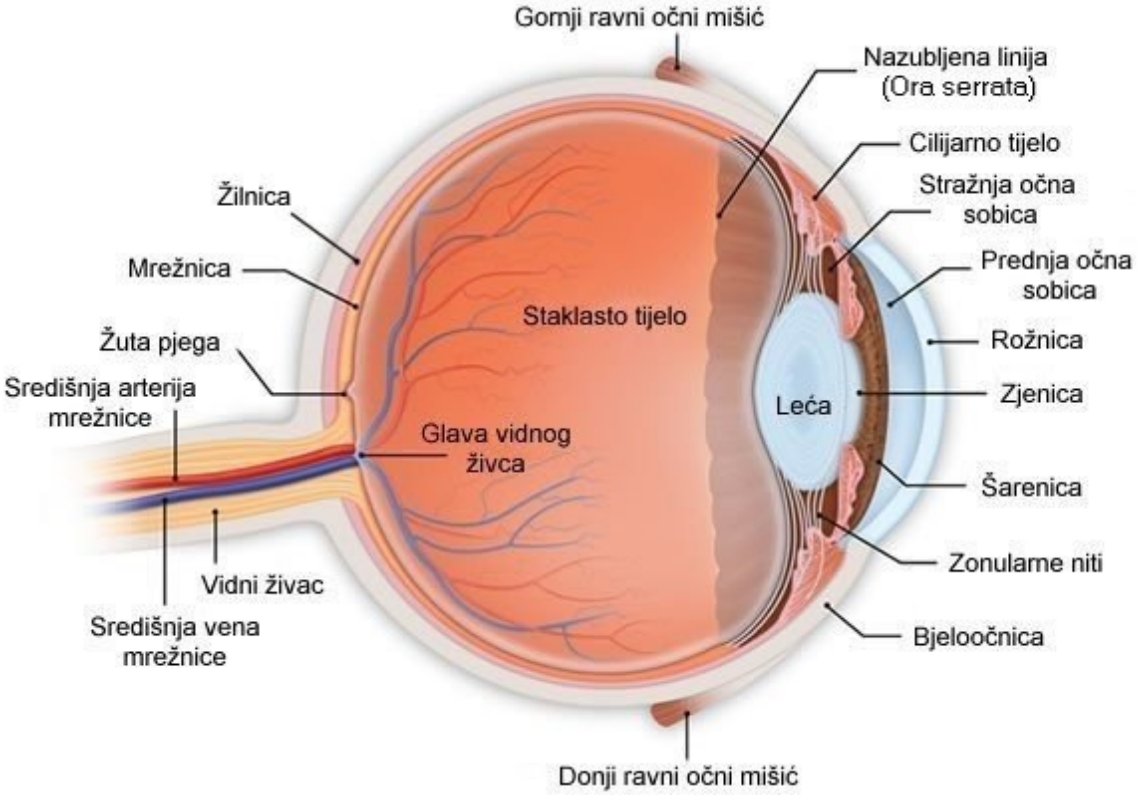
4 9 2

2. ELEMENTI VIZUALNE PERCEPCIJE

Struktura ljudskog oka

Oko je sfera promjera cca 20 mm, dok je sama zjenica 2-8 mm promjera širenja. Vidljiva svjetlost je elektromagnetski val, a svjetlo koje vidimo kombinacija više valnih duljina.

Fiziologija oka



*Slika 4. Fiziologija ljudskog oka (*Izvor: *hr.wikipedia.org* , *www.optometrija.net)*

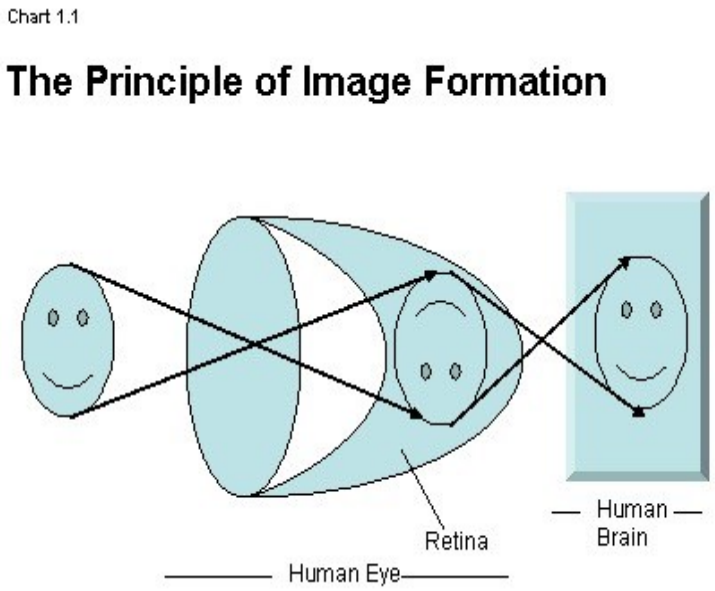
Mrežnicu sačinjavaju čunjići (R,G,B) i štapići.

Metamerizam: pojava kod koje se dvije razlicite boje pod odredjenim osvjetljenjem cine kao jedna te ista ili dvije iste boje pod nekim drugim osvjetljenjem postaju razlicite.

FORMIRANJE SLIKE U OKU

Udaljenost između centra leće i retine varira od 17mm-14mm, a njome upravljaju mišići.

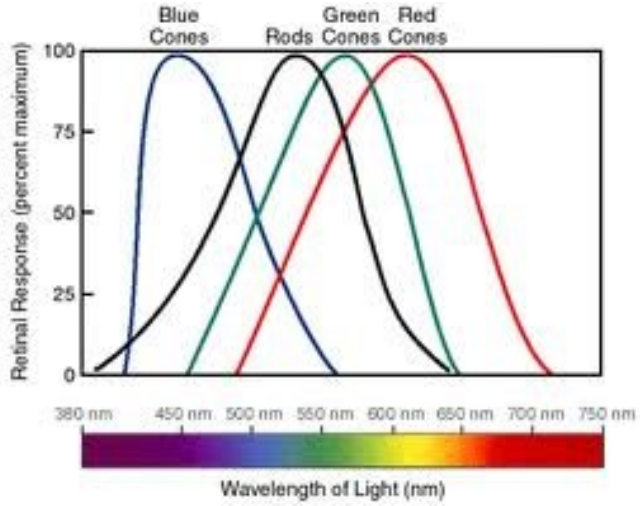
Slika pada na retinu te dolazi do ekscitacije receptora svjetla, koji energiju svjetla transformira u električne impulse i očnim živcem prenose do mozga.



Slika 5. Formiranje slike.

Optičke iluzije su posljedica iskorištavanja nesavršenosti ljudskog vizualnog sustava. Još uvijek nisu u potpunosti istražene i shvaćene.

Svjetlost i elektromagnetski (EM) spektar



Slika 6. Valne duljine i vidljivi dio EM spektra.

Energija je proporcionalna frekvenciji. Veća frekvencija znači više energije po fotonu. Radio valovi imaju najmanje energije, dok mikrovalovi više, pa onda sve više i više energije imaju infracrveni, vidljiva svjetlost, UV, X-zrake i Gamma zrake koje imaju najviše energije.

Fizika pretvaranja svjetlosnog spektra u LMS:

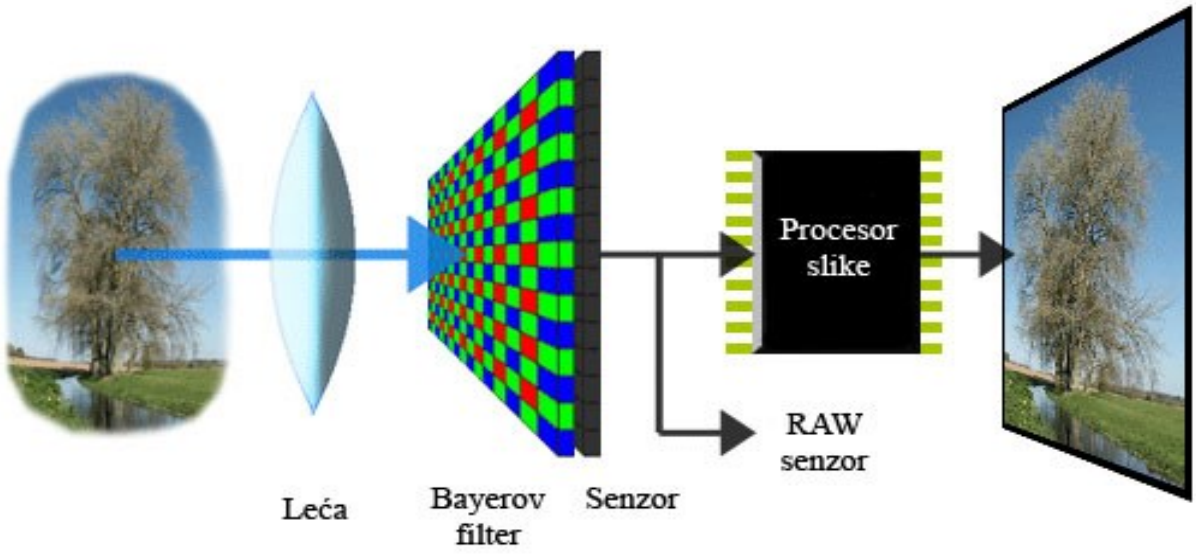
L (450-700)nm; M (450-650)nm; S (400-530)nm

LMS označava nijansu, zasićenje i sjajnost.

AKVIZICIJA SLIKE

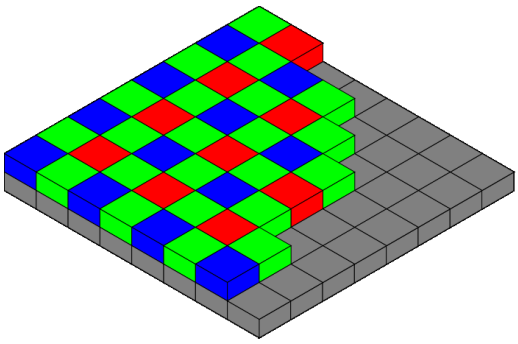
Postupak dobivanja slike putem digitalnog uređaja nazivamo akvizicijom slike. Kada svjetlost padne na neki objekt ona se reflektira i kroz zakrivljenu leću pada na Bayerov filter fotosenzora. Taj filter za svaki piksel pamti po samo jednu boju (crvenu, zelenu ili plavu), intezitet svjetla pretvara u napon i pohranjuje ga na RAW senzor.

Prilikom očitanja sa senzora nadodaje se šum te signal kao takav se dalje komprimira u željeni format (.jpeg, .tiff, .png i sl.).



Slika 7. Postupak akvizicije slike.

Bayerov uzorak



Slika 8. Bayerov uzorak fotosenzora.

Bayerov uzorak fotosenzora (Colour Filter Array – CFA), patentiran je 1975, od strane firme Kodak. Prikazuje način raspoređivanja RGB filtera boja na kvadratnu mrežu fotosenzora.

Uzorak filtera (Bayer) je 50 % zelen, 25 % crven i 25 % plav. Zelenih senzora ima više iz razloga što je ljudsko oko osjetljivije na zelenu boju nego na bilo koju drugu.

Generativni model slike

Generativni model: proces kojim se generiraju slike (npr. niže rezolucije).

Slika 8. Koraci generativnog modela:

(a) visoko-rezolucijska planarna površina,

(b) geometrijska transformacija,

(c) optičko zamućivanje i zamućivanje gibanjem,

(d) smanjivanje rezolucije.

Matrična forma modela:

f(x,y) je leksikografsko izmjenjivanje redosljeda piksela u $\bar{f}$

M_n geometrijska transformacija, operator skaliranja i PSF (funkcija raspršenja točke –*point spread function*)

α_n i β_n su skalarni iluminacijski parametri

η je šum

$$g_n = \alpha_n M_n \bar{f} + \beta_n + \eta_n$$

Generativni model svih *N* slika posložen je horizontalno (za svaku sliku jedan red) da bi sačinjavao linearni sustav:

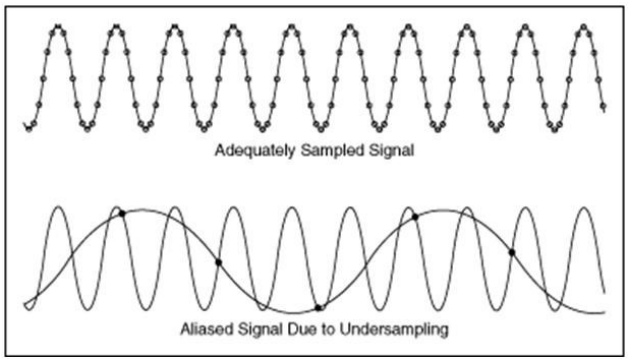
$$\begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{N-1} \end{bmatrix} = \begin{bmatrix} \alpha_0 & & & \\ & \alpha_1 & & \\ & & \ddots & \\ & & & \alpha_{N-1} \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ M_{N-1} \end{bmatrix} * \bar{f} + \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{N-1} \end{bmatrix} + \begin{bmatrix} \eta_0 \\ \eta_1 \\ \vdots \\ \eta_{N-1} \end{bmatrix}$$

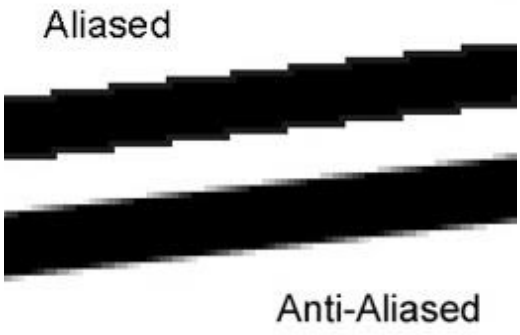
Ili skrećeno napisano:

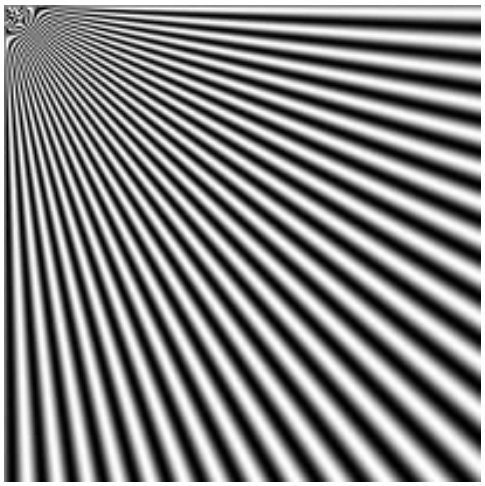
$g = \Lambda_\alpha M\bar{f} + \beta + \eta$

Aliasing

Shannonov teorem uzorkovanja: ako se funkcija uzorkuje s vrijednošću jednakom ili većom od njene dvostruke najveće frekvencije, moguće je rekonstruirati kompletnu originalnu funkciju iz njezinih uzoraka. Ako je funkcija poduzorkovana, onda fenomen poznat kao ALIASING korumpira uzorkovanu sliku.



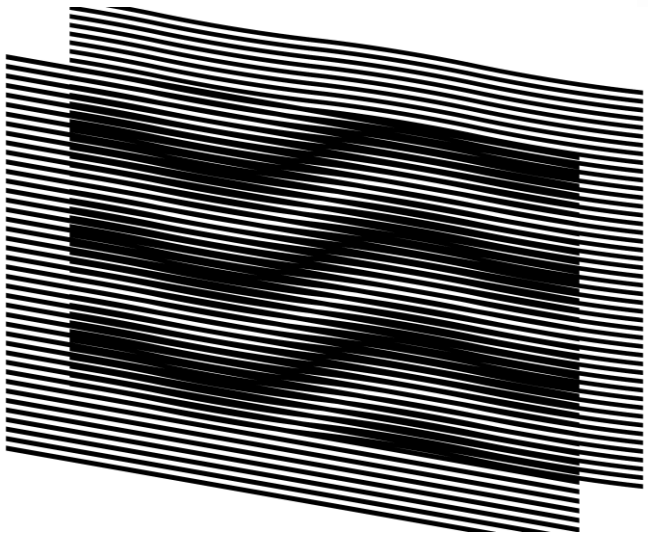


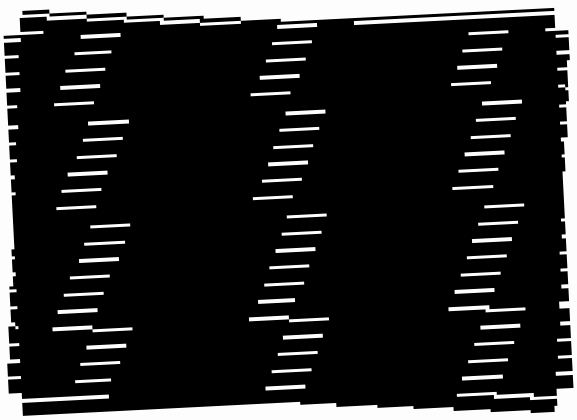


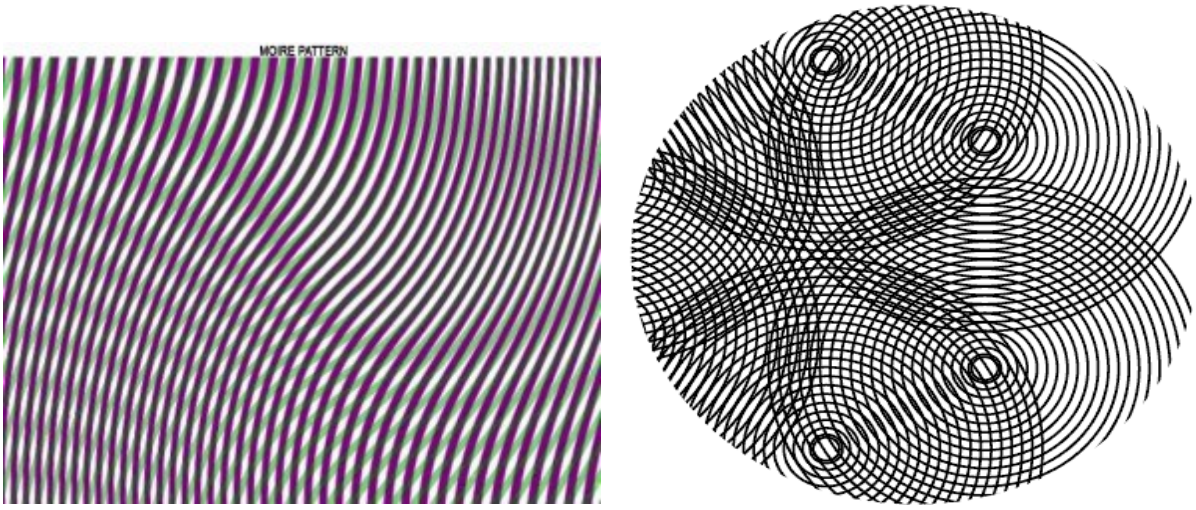
Slika 9. Primjeri aliasinga.

Moiré uzorak

Moiré efekt nastaje kao poslijedica interferencije kada se dvije mreže preslože pod nekim kutem.



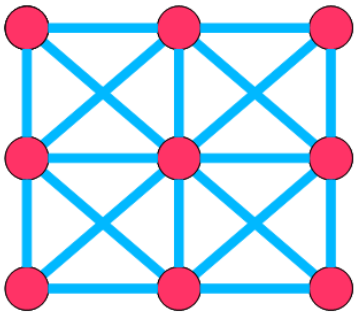




Slika 10. Primjeri Moiré efekta.

Osnovni odnosi među pikselima

Susjedstvo odnosno udaljenosti između piksela definiraju se preko Euklidove udaljenosti: $D(P,Q)=[(x-s)^2+(y-t)^2]^{1/2}$



Slika 11. Crveni krugovi predstavljaju piksele u slici, dok plave linije predstavljaju udaljenosti definirane Euklidovim relacijama.

Operacije na slikama na bazi piksela

Opreacije na bazi piksela mogu se podijeliti na: linerane, nelinearne, bezmemorijske, transformacije vrijednosti točaka, linearne po segmentima, histogramske, aritmetičke i logičke.

- Linearne operacije mogu se prikazati preko relacije:

$$H(af + bg) = aH(f) + bH(g)$$

gdje su f i g bilo koje dvije slike, a i b bilo koja dva skalara, dok je H linearni operator.

Rezultat primjene linearnog operatora na sumu dviju slika (koje su pomonožene nekim skalarima), identičan je primjeni operatora na svaku sliku posebno, množeći rezultat određenim skalarom. Npr. operator koji računa sumu K slika je linerani operator.

- Nelinerane operacije

Karakteristika nelinearnih operacija je da izlazna vrijednost piksela ovisi samo o ulaznoj vrijednosti ulaznog piksela na tom mjestu.

- Bezmemorijske operacije

Kod bezmemorijskih operacija se ulazna vrijednost točke preslikava u izlaznu vrijednost točke prema nekoj transfromaciji.

- Osnovne transformacije vrijednosti točaka:
  - negativ slike,
  - logaritamske i
  - eksponencijalne transformacije.
- Transformacije linearne po segmentima:
  - Rastezanje kontrasta
  - Ograničavanje (clipping)
  - Izdvajanje prozora
  - Modeliranje histograma
- Aritmetičke i logičke operacije

Kod operacije koje se vrše nad pikselima izlazna vrijednost piksela ovisi samo o ulaznoj vrijednosti ulaznog piksela na tom mjestu.

- Logaritamske transformacije

Kod logaritamskih transformacija opseg točaka slike je tako velik da je samo nekoliko točaka vidljivo. ***log*** funkcija ističe točke male vrijednosti relativno u odnosu na točke velike vrijednosti.

- Eksponencijalne transformacije (gama korekcija)

Jedna od najčešće korištenih eksponencijalnih transformacija je gama korekcija.

- Rastezanje kontrasta

Povećanje kontrasta koristi se kod naglašavanja detalja u slici. Niski kontrast je rezultat nejednolikog osvjetljenja scene ili slabog dinamičkog opsega senzora. Ima izrazito malu razliku između svijetlih i tamnih dijelova slike što dovodi do teže interpretacije sadržaja. Za dobivanje bolje slike potrebno je uski pojas sivih tonova "razvući" na širi interval, što nazivamo rastezanjem kontrasta.

- Clipping (ograničavanje)

Koristi se za uklanjanje šuma, odnosno uklanjanje vrijednosti signala koje leže u nekom intervalu [a, b].

- Thresholding (postavljanje praga)

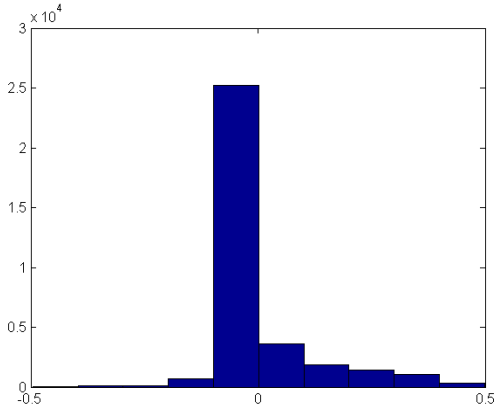
Postavlja se prag kod kojeg se vrijednosti manje od ***th*** preslikavaju u 0, a veće vrijednosti u 1.

Slika 12. Izvlačenje rubova u slici metodom thresholdinga.

- Modeliranje histograma
  - Histogram prvog reda: relativna frekvencija pojave različitih vrijednosti točaka u slici.

- Tehnike modeliranja histograma: mijenjanje slike tako da se dobije histogram željenog oblika
- Izjednačavanje histograma
- Histogrami viših redova (histogram drugog reda i sl.)

- Aritmetičke i logičke operacije
  - zbrajanje, oduzimanje,
  - unarna operacija- Not, binarne operacije- And, Or



Slika 13. Histogram slike.

Prostorne operacije

Prostorne operacije na slikama su one opracije kod kojih izlazna vrijednost točke ovisi o ulaznim vrijednostima susjedstva te točke.

Prostorno usrednjavanje je najčešće operacija koja se koristi za uklanjanje šuma, dok zauzvrat daje dosta zamućenu sliku.

Najčešće korištene prostorne operacije su:

- Median filtar:

Izlazna vrijednost točke je median piksela sadržanih unutar prozora filtra. Median skupa brojeva se dobije na način da se brojevi poredaju po veličini te se kao rezultat odabere broj koji je rangiran kao srednji. Median filtar je dobar za binarni šum, a loš za Gaussov šum.

- Uklanjanje neoštrine

Uklanja se nisko frekvencijska komponenta proporcionalna neoštrom dijelu slike. Ovo je u suštini ekvivalentno dodavanju gradienta (tj. visoko frekvencijske komponente signala) slici.

- Linearno filtriranje:
    - Nisko propusno (NP) – uklanjanje šuma i interpolacija,
    - Visoko propusno (VP) – ekstrakcija rubova i pooštravanje,
    - Pojasno propusno (PP) – poboljšanje rubova uz prisutnost šuma,
    - Pojasna brana (PB)

Kod linearnog filtriranja VP filtri koriste se i za ekstrakciju značajki u slikama koje ne moraju biti samo rubovi.

- Interpolacija slika

  Interpolacija je dodavanje piksela između već postojećih. Interpolacijom dolazi do podizanja rezolucije.

- Gradijent slike

Slika 14. Gradijent(bijelo) piksela (crno) u slici.

3. GEOMETRIJSKE TRANSFORMACIJE I REGISTRACIJA SLIKA

Geometrijske transformacije nad slikama

Najjednostavniji pristup reprezentaciji slika je rastava na jednostavne komponente transformacija:

- 2-D linearne transformacije
- 2-D ortogonalne transformacije

Matrični zapis 2-D ortogonalnih transformacija

Kod linearnih transformacija uvijek postoji par IZRAVNA i INVERZNA transformacija. Dok IZRAVNA transformacija daje niz koeficijenata, INVERZNA transformacija koristi te iste koeficijenete za dobiti originalnu sliku inverznim putem.

Nekolineranost (dekorelacija) je tendencija koeficijenata transformacije da budu nekolerirani.

Geometrijske transformacije slike

Da bi geometrijske transformacije mogle biti moguće potrebno je piksele u slikama predstaviti ***homogenim koordinatama***:

$$\begin{bmatrix} \mathrm{x}' \\ \mathrm{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Za vrijednost piksela u slici *f* na *x*-tom retku i *y*-tom stupcu pristupa se na način:

f(x,y,1)

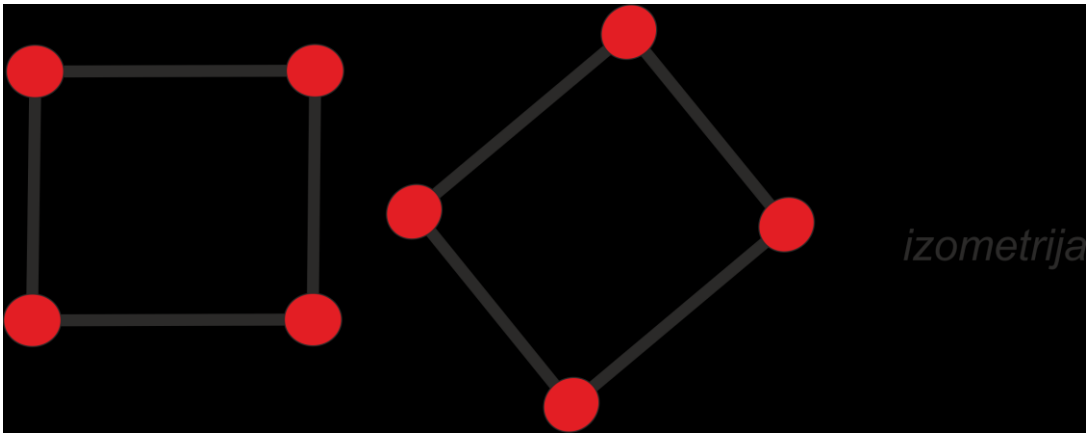
Za inverzno računanje koordinata piksela u slici potrebno je gornju matricu dimenzija 3×3 invertirati. Kada to ne bi bila kvadratna matrica nebi ni postojala inverzna matrica, te se zato koristi trodimenzionalna prezentacija 2D vektora što se naziva ***prikaz pomoću homogenih koordinata.***

Geometrijske transformacije se mogu podijeliti na:

- Translacija
- Rotacija
- Skaliranje
- Afina transformacija
- Izometrija
- Projekcijska transformacija

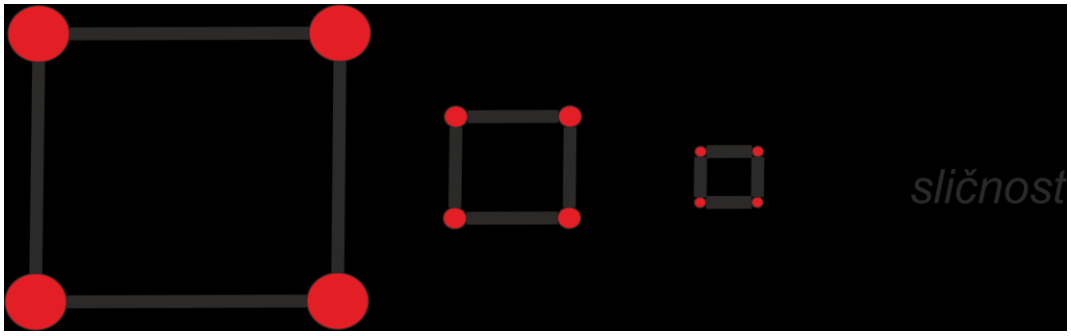
Izometrija – translacija i rotacija

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



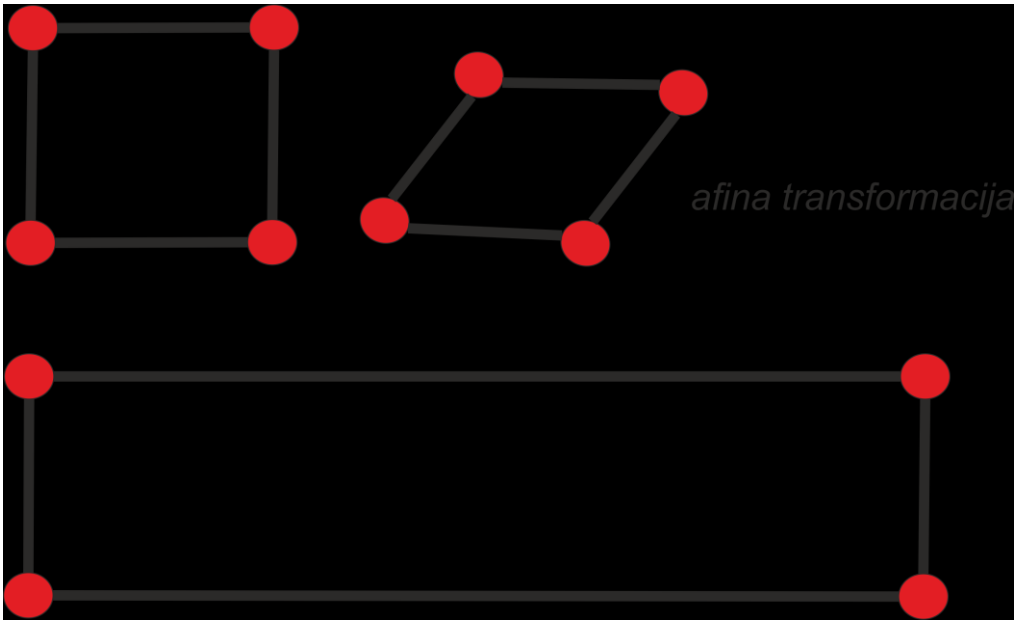
Sličnost – zumiranje

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



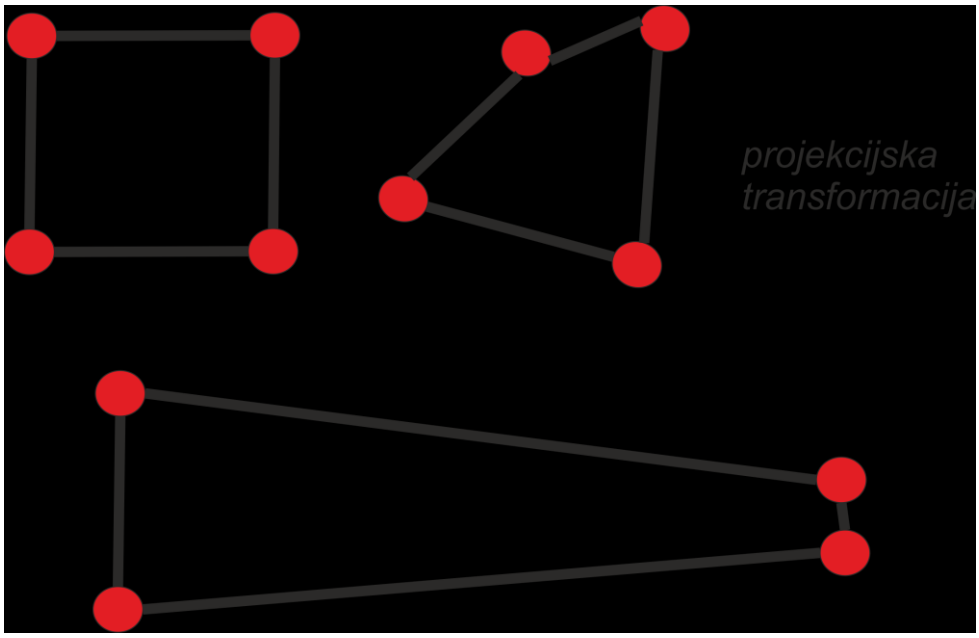
Afina transformacija – nesingularna linearna transformacija koja prethodi translaciji

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Projekcijska transformacija – generalna nesingularna linearna transformacija homogenih koordinata

$$x' = \begin{bmatrix} A & t \\ v^T & v \end{bmatrix} x \qquad v = (v_1, v_2)^T$$



Diskretne transformacije

- Fourierova transformacija
- Kosinusna transformacija
- Karhunen-Loeve transformacija

Registracijski parametri

Registracije u slici koriste se za pronalaženje objekta u slici, praćenje objekta, spajanje slika, šivanje slika i još mnogo toga.

Matematička relacija koja prikazuje registracije između dvije slike dana je jednadžbom:

$$\mathbf{u}_i' = \mathrm{H}(\theta)\,\mathbf{u}_i$$

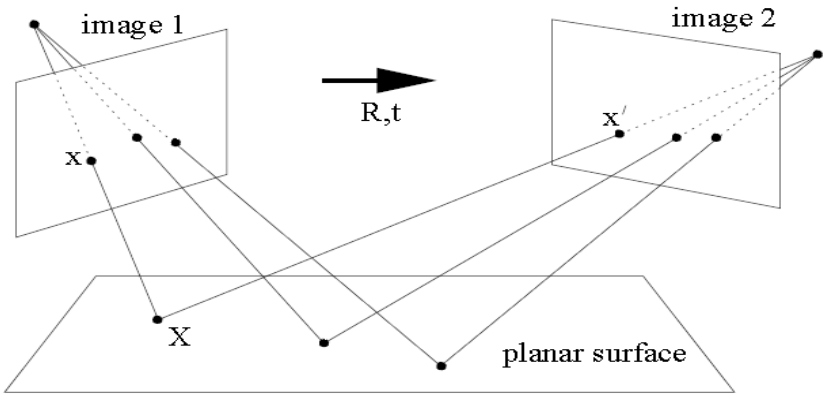


Figure 3.1: **Images of planes**. A planar homography is induced between the two images of a plane taken from different viewpoints (related by a rotation R and translation t). The scene point X is projected to points x and x′ in the two images. The image points are related by $x' = Hx$.

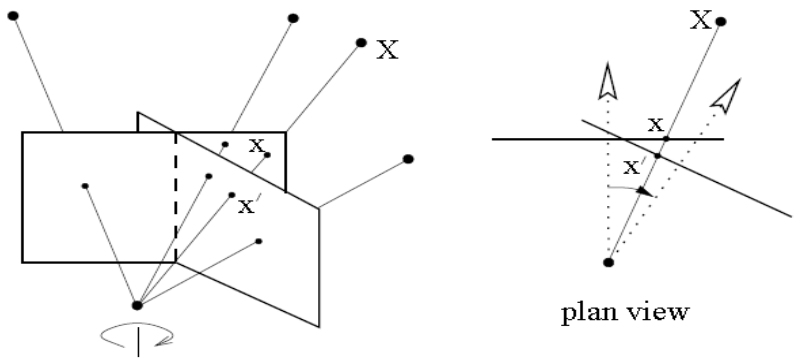


Figure 3.2: **Rotation about the camera centre**. As the camera is rotated the points of intersection of the rays with the image plane are related by a planar homography. Image points x and x′ correspond to the same scene point X. The image points are related by $x' = Hx$.

Izračun parametara homografije

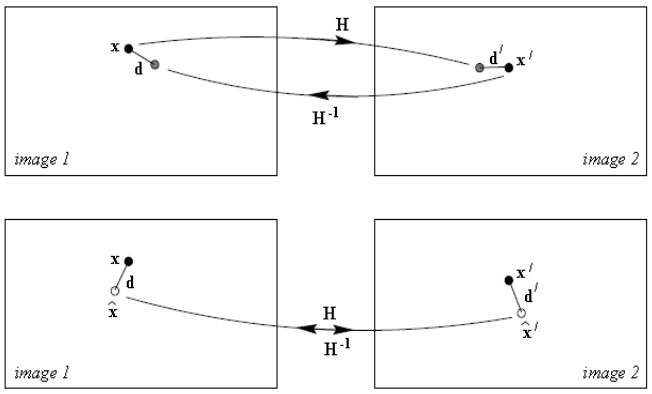
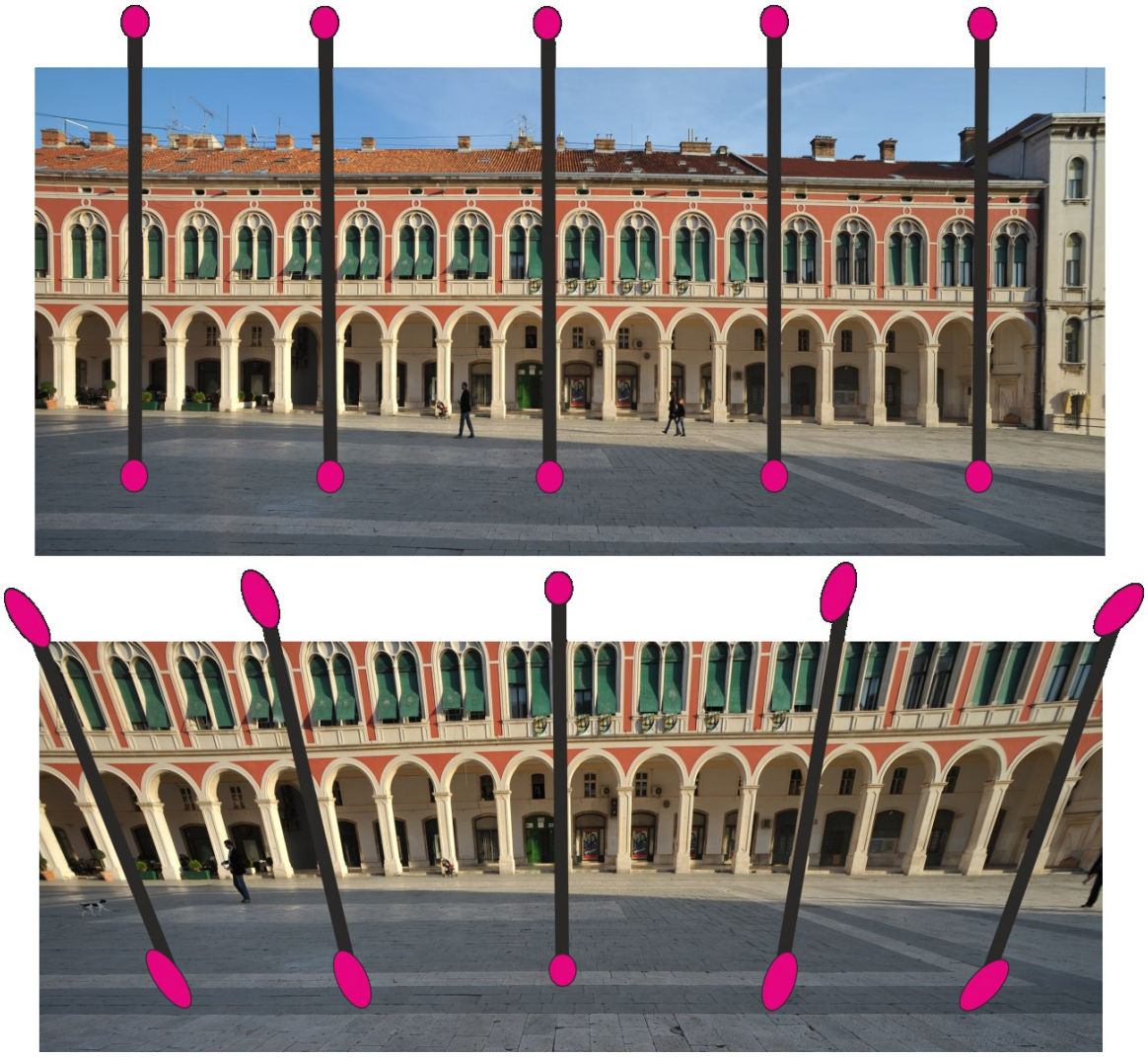


Figure 3.3: *(Top)* Homography computation using the forward-backward transfer distance metric of equation (3.4). The metric is simple to use, but does not give the maximum likelihood (ML) estimate of H. *(Bottom)* The ML estimator of equation (3.9) minimizes the reprojection error between the pre-image point correspondence $(\hat{x}, \hat{x}')$ and the observed interest points (x, x').

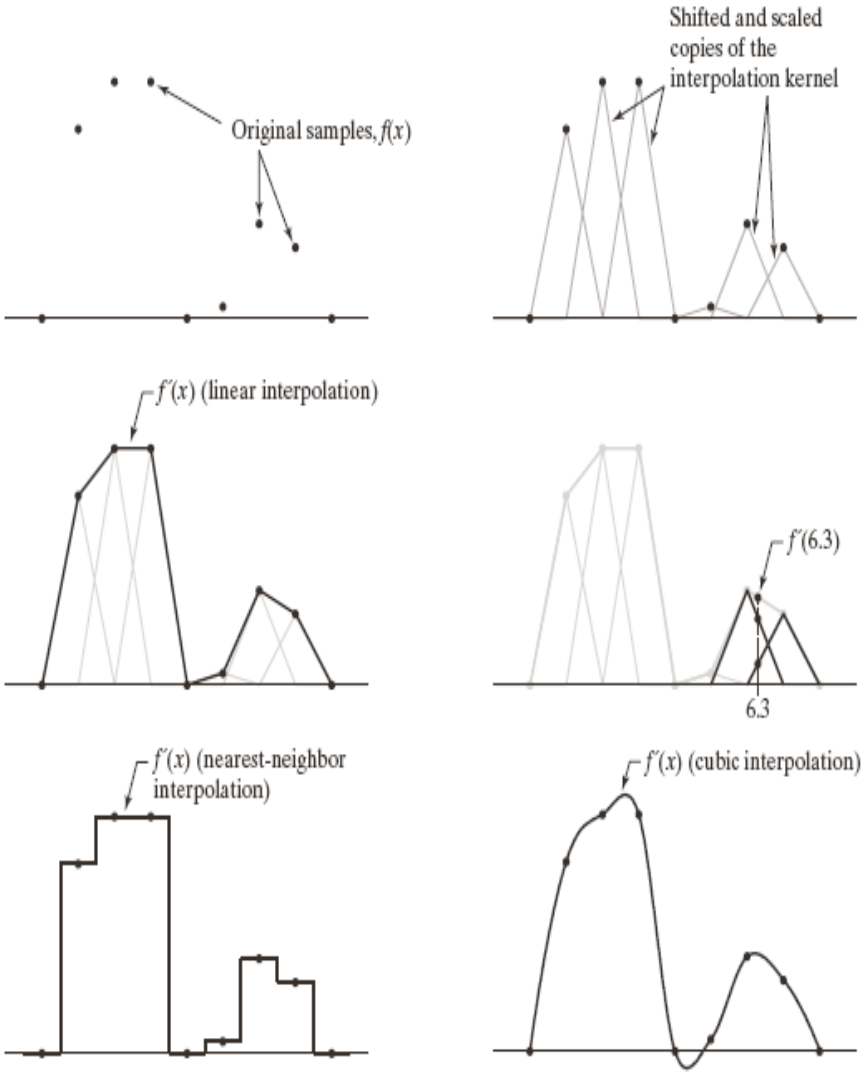
Registracije



Slika 15. Izračun parametara registracije.

Interpolacija slika

Interpolacija je dodavanje novih točaka između već postojećih točaka u slici. Najpoznatije metode interpolacije su interpolacija ponavljanjem redova i stupaca, linearna interpolacija (prvo uzduž redova, a zatim uzduž stupaca), bilinearna interpolacija i konvencionalna interpolacija.



a b
c d
e f

FIGURE 6.15 (a) Original samples, $f(x)$. (b) Copies of triangular interpolation kernel, $h_T(x)$, shifted and then scaled by the values of $f(x)$. (c) Sum of the shifted and scaled interpolation kernels, resulting in $f'(x)$ as computed using linear interpolation. (d) Computing $f'(6.3)$ as $f(6)\,h_T(0.3) + f(7)\,h_T(0.7)$. (e) $f'(x)$ computed using nearest-neighbor interpolation. (f) $f'(x)$ computed using cubic interpolation.

4. OBRADA SLIKA U BOJI

Osnove obrade slika u boji temelje se na prostornim procesima, vezanim za koordinate piksela u slici. Sve se svodi na vektore i matrice. Transformacije između prostora boja su česte i neophodne.

Rezanje boja definira se kao osvjetljavanje specifičnog ranga boja u slici, u svrhu odvajanja objekta od njegovog okruženja.

Korekcije tona i boje koriste se u svrhu poboljšanja fotografije i reprodukcije boje, ponajviše kod izrade fotografija.

Obrada histograma najčešće se koristi za širenje inteziteta boja, i to uniformno, tako da boje ostanu nepromijenjene (najčešći postupak za HSI prostor boja). Za zaglađivanje slika u boji koristimo prostorno filtriranje. Dok za pooštravanje slika u boji upotrebljavamo Laplacian.

Što se tiče direktnog rada na vektoru boja, segmentacija slike je proces koji odvaja sliku u regije. Te segmentirane regije slike spadaju u određene klastere objekata, te je to osnova za prepoznavanje u računalnom vidu. Najčešće se koristi u HSI prostoru boja, dok se najbolji rezultati dobivaju u RGB prostoru boja. Jednom segmentirane slike i klasterirani objekti dalje mogu ići u obradu analize oblika, analize pokreta i same registracije slika.

Detekcija rubova najbolje rezultate daje upotrebom gradijentnih operatora.

U svakom kanalu slike u boji šum ima istu karakteristiku, ali je isto tako moguće i da kanali boja budu različito utjecani od strane šuma. Npr. ako elektronika jednog kanala boje nije u funkciji, onda nam na tom kanalu izostaje šum.

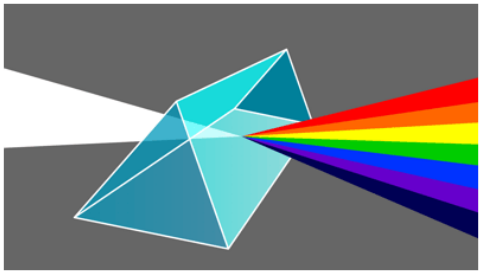
PREZENTACIJA SLIKA U BOJI

Full-colour processing su slike dobivene s punim senzorom boje (TV kamera, skener).

Pseudo-colour processing primjenjuje se za dodjeljivanje boje partikularnom monokromatskom intezitetu.

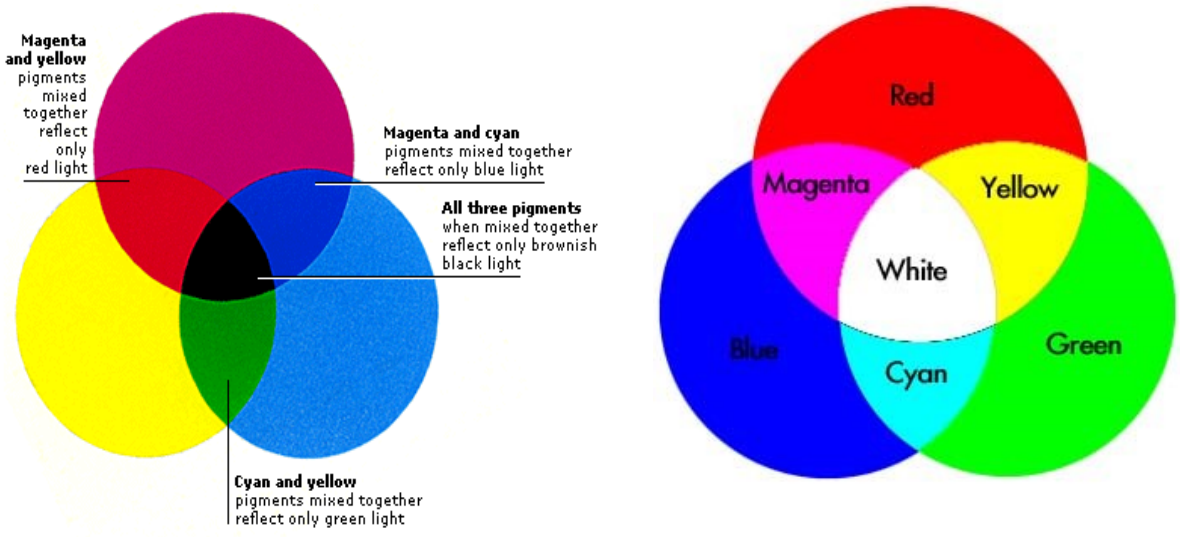
Godine 1666. Sir Isaac Newton je otkrio da kada zraka sunca prođe kroz staklenu prizmu, zraka koja kroz tu prizmu s druge strane izbija nije bijela, već se sastoji od kontinuiranog spektra boja, od ljubičaste do crvene. Uočio je da se spektar dijeli na šest širokih regija:

- ljubičastu
- plavu
- zelenu
- žutu
- narančastu
- crvenu



Slika 16. Razlaganje bijele svijetlosti na spektar boja uz pomoć prizme.

Primarne i sekundarne boje



Slika 17. Primarne (desno) i sekundarne (lijevo) boje.

MODELI BOJA

Prostor boja ili sustav boja je specifikacija modela boja u koordinatnom sustavu gdje se svaka boja predstavlja s jednom točkom.

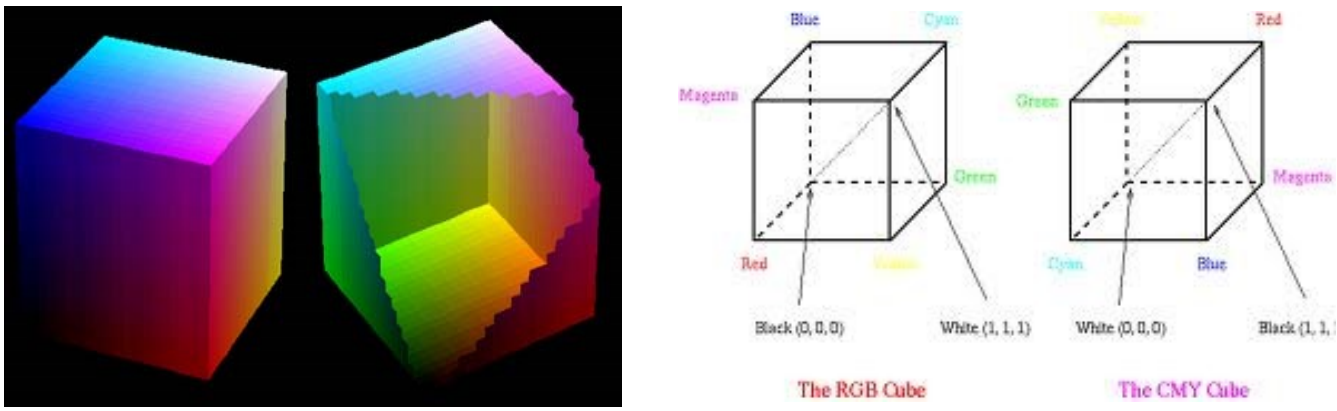
Modeli boja orjentirani na hardwareu, a usmjereni na manipulaciju boja potrebnu za digitalnu obradu slike su RGB, CMY, CMYK, HSI.

Modeli boja za slike su:

- RGB: baziran na Kartezijevom koordinatnom sustavu

Ishodište sustava je crna boja, dok je najveća dijagonala bijela boja.

Full colour image: 24-bit RGB kocka boje

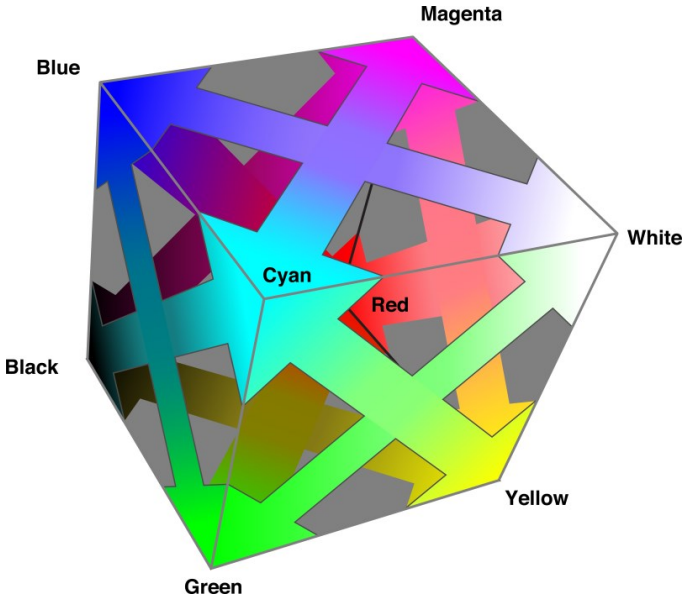


Slika 18. RGB prostor boja.

- CMY (sekundarne boje ili primarni pigmenti)

Interna konverzija koju koristi većina printera je u CMY prostor boja, s nadodanom crnom bojom CMYK (uz pretpostavku da su vrijednosti boja normirane u rangu [0 1]).

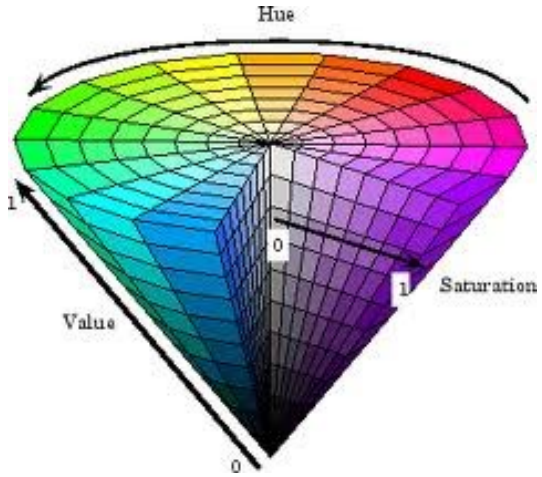
$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$



Slika19. CMY prostor boja.

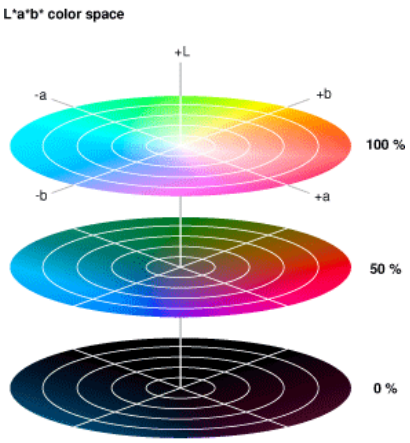
- Softverski orijentirani modeli boja

HSB, HLS, HSV, HSI



Slika 20. HSV prostor boja.

- L*a*b* prostor boja



*Slika 21. L*a*b model boja.*

- Modeli boja za video signal

Y’UV, Y’IQ, Y’PbPr, Y’CbCr

RGB

YCbCr

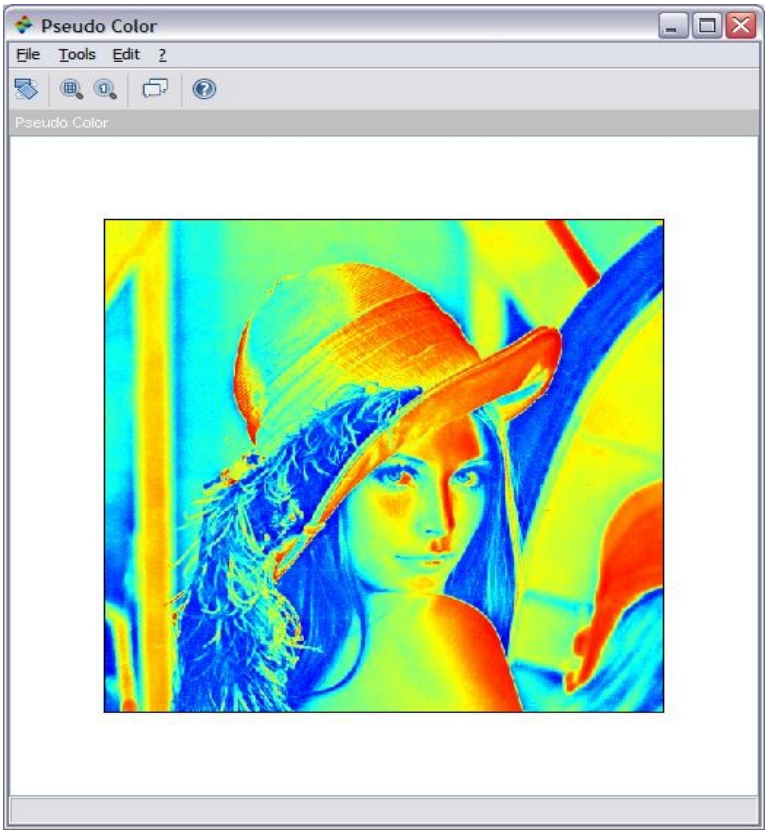
CMY

Slika 22. Modeli boja za video signal.

Obrada slika pseudo koloriranjem

Pseudocolour označava “lažnu boju”, a postupak pseudokoloriranje jest dodjeljivanje boje nivoima sive, temeljem određenog kriterija.

Rezanje inteziteta (density) je dodjeljivanje različitih boja svakom “odresku” ravnine u prostoru.



Slika 23. Pseudokoloriranje u Matlabu.

5. TRANSFORMACIJE INTEZITETA I PROSTORNO FILTRIRANJE

Algoritmi za obradu slike mogu se podijeliti na prostorne i frekvencijske algoritame. Dok prostorni algoritmi vrše direktnu manipulaciju nad pikselima u slici, frekvencijski se baziraju na modificiranju Fourierove transformacije slike

Funkcije transformacije inteziteta

Osnove transformacije na svim slikama su traženje negativa slike, logaritamske transformacije i transformacije potenciranja:

$$s = T(r)$$

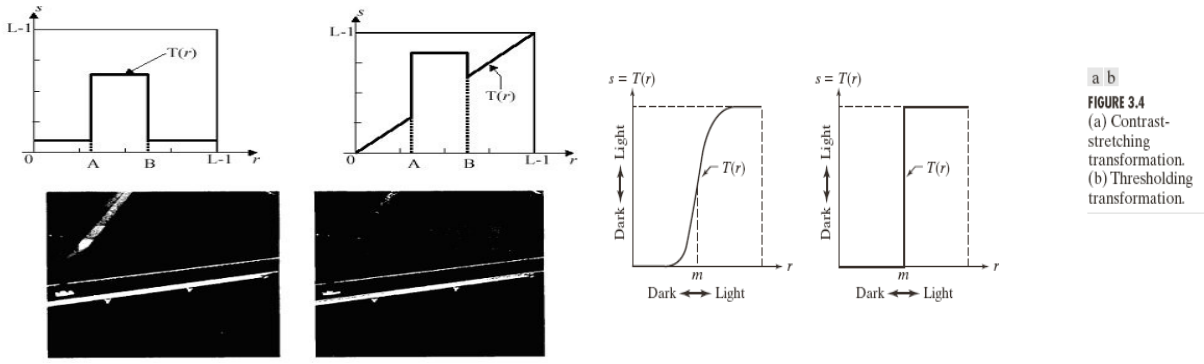
Gdje je T-transformacija

s – piksel nakon transformacije

r – piksel prije transformacije

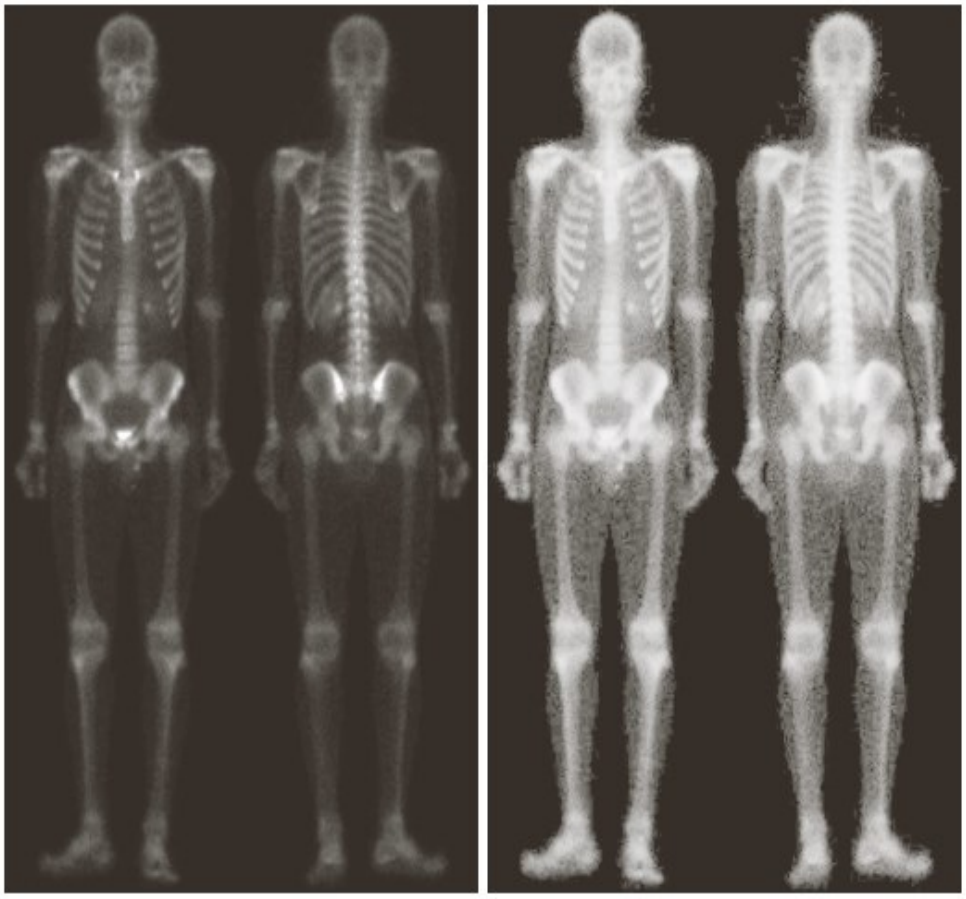
Transformacije funkcijama lineranim po dijelovima:

Rastezanje konstrasta kao transformacija koje je linearna po dijelovima ima za cilj povećanje dinamičkog ranga sivih tonova slike koja se procesuira. To se postiže rezanje sivih nivoa, dok se za isticanje specifičnog ranga sivih tonova koristi bit-plane rezanje, u kojemu samo pet najviših redova sadrži vidljive podatke.



a b
FIGURE 3.4
(a) Contrast-stretching transformation. (b) Thresholding transformation.

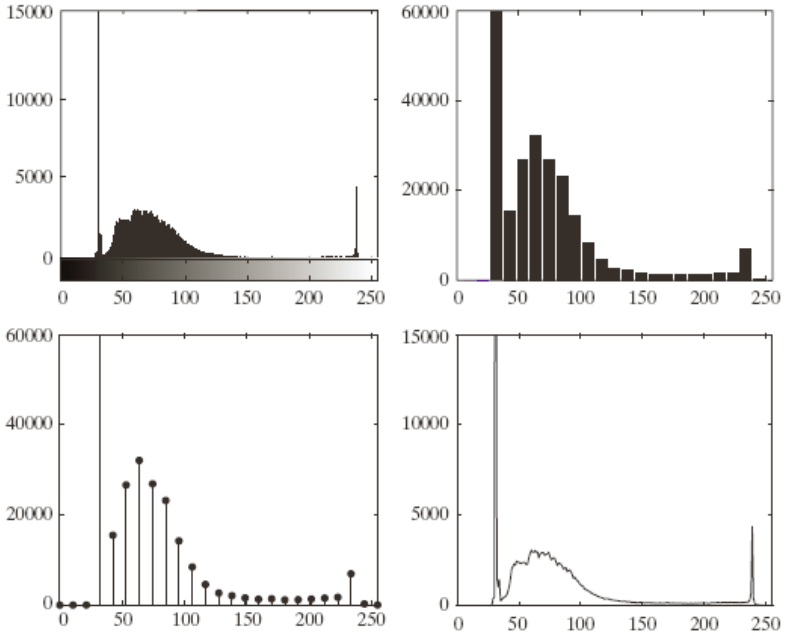
Slika 24. Rastezanje kontrasta.



a b
FIGURE 3.6
(a) Bone scan image. (b) Image enhanced using a contrast-stretching transformation. (Original image courtesy of G. E. Medical Systems.)

Slika 25. Poboljšanje vidljivosti slike uz pomoć rastezanja kontrasta.

Obrada histograma spada u obradu slika u realnom vremenu.

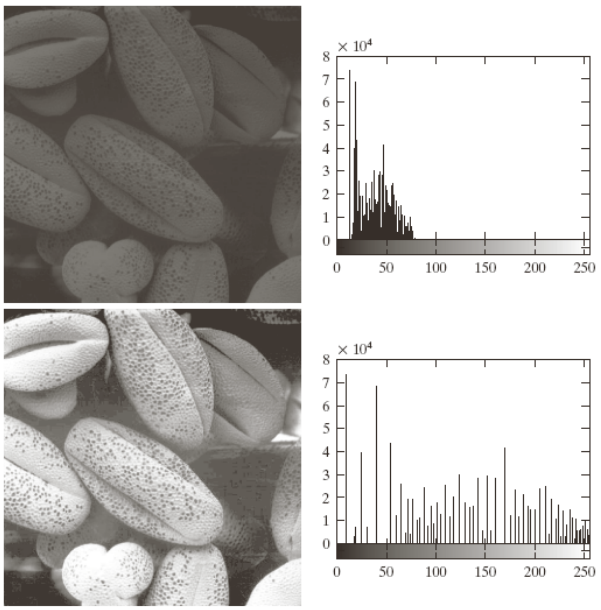


a b
c d
FIGURE 3.7 Various ways to plot an image histogram. (a) `imhist`, (b) `bar`, (c) `stem`, (d) `plot`.

Slika 26. Histogram slike.

Ekvilizacija (izjednačavanje) histograma

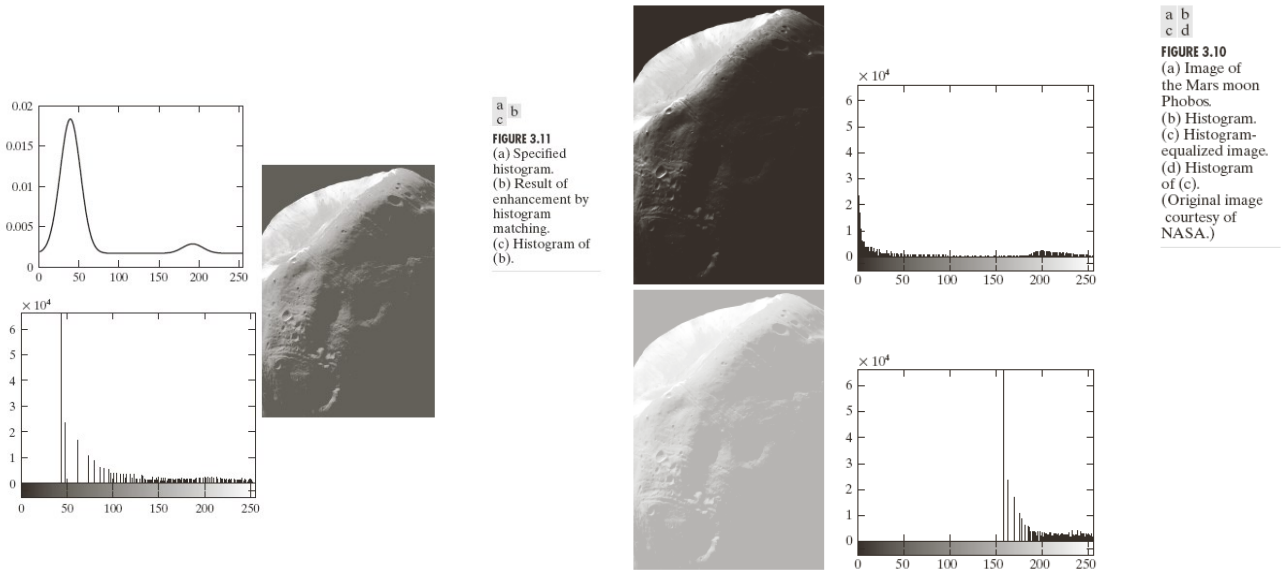
Ekvilizacija (izjednačavanje) histograma automatski određuje funkciju transformacije koja određuje izlaznu sliku na način da izlazna slika ima uniformni histogram.



a b
c d
FIGURE 3.8 Illustration of histogram equalization. (a) Input image, and (b) its histogram. (c) Histogram-equalized image, and (d) its histogram. The improvement between (a) and (c) is evident. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra.)

Slika 27. Ekvilizacija histograma.

Preklapanje (specifikacija) histograma



Slika 28. Poboljšanje slike Phobosa (Marsovog mjeseca) upotrebom ekvilizacije histograma.

Prostorno filtriranje (linearno-nelinerano)

Obrada piksela u susjedstvu sastoji se od:

1. Odabira centralne točke (x,y)
2. Izvođenja operacije koja uključuje samo prije definirano susjedstvo oko (x,y)
3. Rezultat te opercije je izlaz procesa u toj točki
4. Ponavljanje koraka 1-4 za svaki piksel u slici

- Linearno prostorno filtriranje

Linearno prostorno filtriranje je filtriranje izvođeno direktno na pikselima slike. U Matlabovom Image Processing Toolbox-u vrši se korištenjem funkcije *fspecial.*

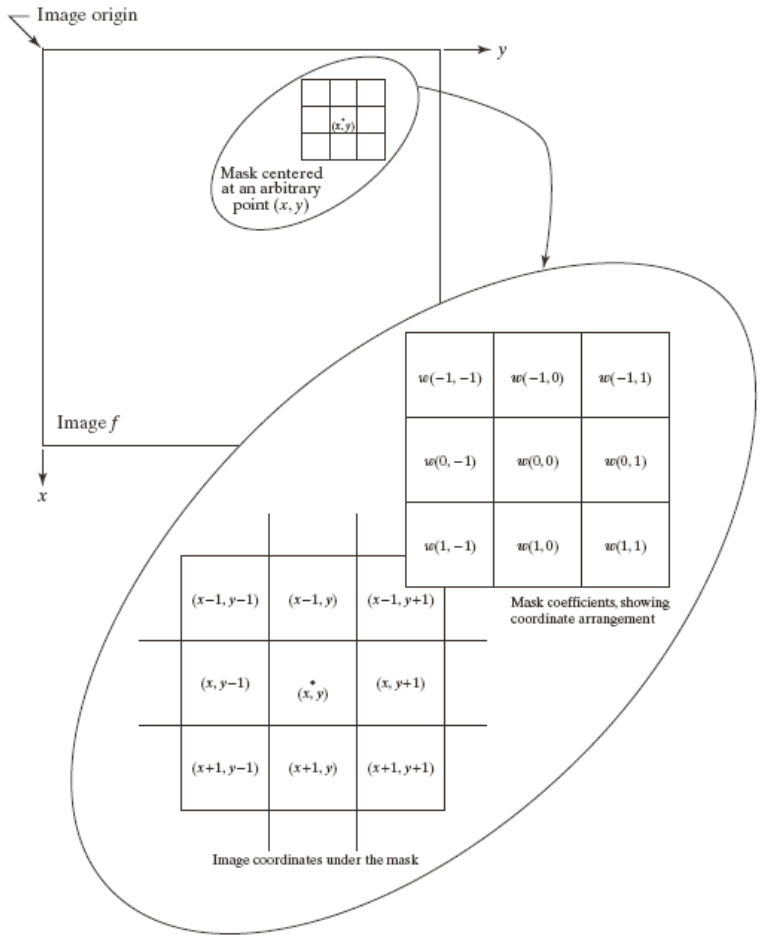


FIGURE 3.13
The mechanics of linear spatial filtering. The magnified drawing shows a 3× 3 filter mask and the corresponding image neighborhood directly under it. The image neighborhood is shown displaced out from under the mask for ease of readability.

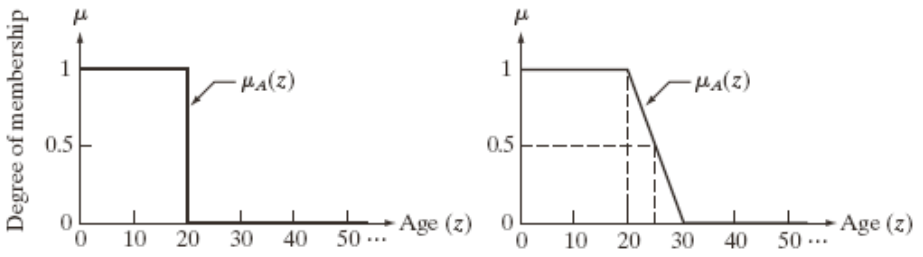
- Nelinearno prostorno filtriranje

Nelinearno prostorno filtriranje također je bazirano na operacijama sa susjednim pikselima u slici. Međutim, za razliku od lineranog filtriranja (računanje sume umnožaka), nelinearno filtriranje je temeljeno na nelinearnim operacijama, koje uključuju piksele u susjedstvu obuhvaćene filterom.

Za Matlab u Image Processing Toolbox-u funkcije za filtriranje su:
nlfilter, colfilt, ordfilt2, medfilt2

Fuzzy (neizrazite tehnike) u transformaciji intenziteta

Fuzzy tehnika daje veću fleksibilnost i postepenu tranziciju. Uveo ju je Zadeh 1965. godine. Neizrazita (fuzzy) logika formalizira neprecizne informacije.



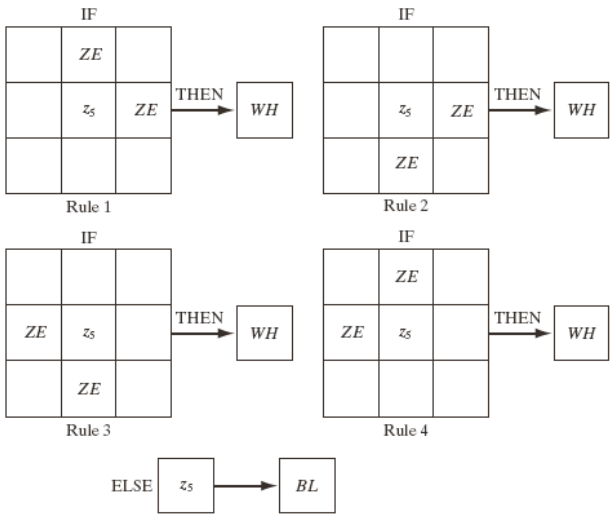
a b

FIGURE 3.20
Membership functions of (a) a crisp set, and (b) a fuzzy set.

Upotreba neizrazitog skupa za transformacije inteziteta poštuje određena pravila.

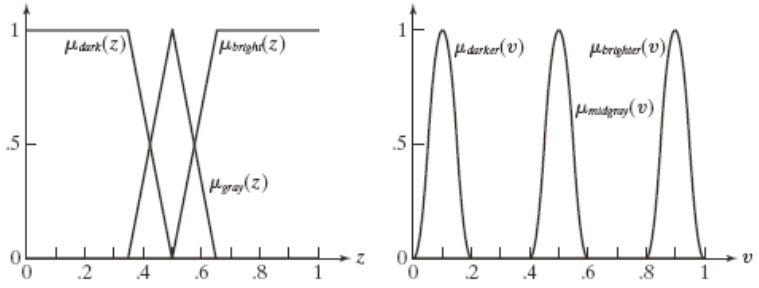
- Pravila:
  - Ako je piksel taman, ONDA ga napravi tamnijim
  - Ako je piksel siv, ONDA ga napravi više sivim
  - Ako je piksel svjetao, ONDA ga napravi svjetlijim

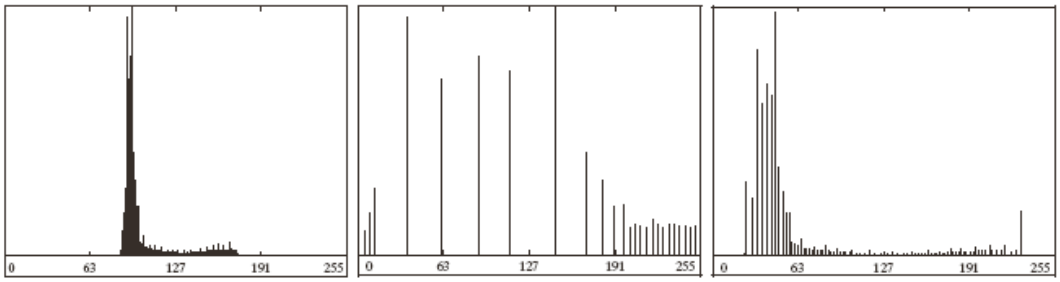
FIGURE 3.32
Fuzzy rules for boundary detection.



a b

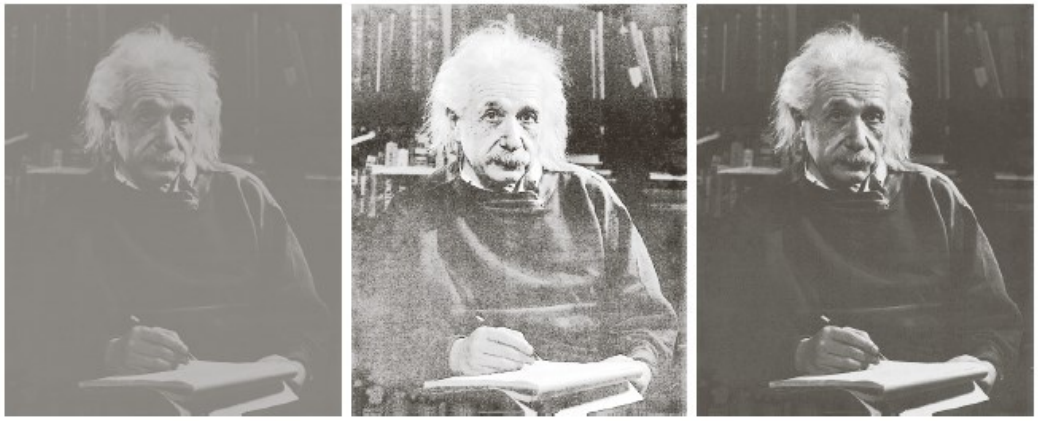
FIGURE 3.27
(a) Input and (b) output membership functions for fuzzy, rule-based contrast enhancement.





a b c

FIGURE 3.29 Histograms of the images in Fig. 3.28(a), (b), and (c), respectively.



a b c

FIGURE 3.28 (a) Low-contrast image. (b) Result of histogram equalization. (c) Result of fuzzy, rule-based, contrast enhancement.

6. FILTRIRANJE U FREKVENCIJSKOM PODRUČJU

Godine 1807. Fourierova idea da se funkcije koje se periodično ponavljaju mogu prezentirati kao sume sinusa i kosinusa različitih frekvencija, od kojih se svaka množi s različitim težinama, dočekana je s ismijavanjem. Danas, je dokazano da Fourierove tehnike omogućuju značajan i praktičan način za učenje i implementaciju različitih tehnika poboljšanja slike i na njima počiva cjelokupna frekvencijska analiza signala.

2D Diskretna Fourierova transformacija

Ako je f(x,y) digitalna slika veličine MxN. 2D diskretna Fourierova transformacija (DFT) može se pisati kao:

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(ux/M + vy/N)}$$

gdje je F(u,v) diskretna Fourierova transformacija, x = 0,1...M-1, y = 0,1...N-1

Inverzna diskretna Fourierova transformacija

Dana je relacijom:

$$f(x,y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) e^{j2\pi(ux/M + vy/N)}$$

DFT i njen inverz u praksi se računaju preko brze Fourierove transformacije (FFT). U MATLABu FFT slike dobiva se upotrebom funkcije *fft2*(f). Fourierov spektar dobiva se upotrebom funkcije *abs(F)*, koja računa veličinu svakog elementa niza.

Filtriranje u frekvencijskoj domeni

Osnova svakog filtriranja bilo u prostornoj ili frekvencijskoj domeni leži na teoremu konvolucije, simbolički zapisanom:

$$f(x,y)h(x,y) \Leftrightarrow H(u,v) * F(u,v)$$

Gdje simbol * označava konvoluciju dviju funkcija, a *F(u,v)* konstituira Fourierov transformacijski par.

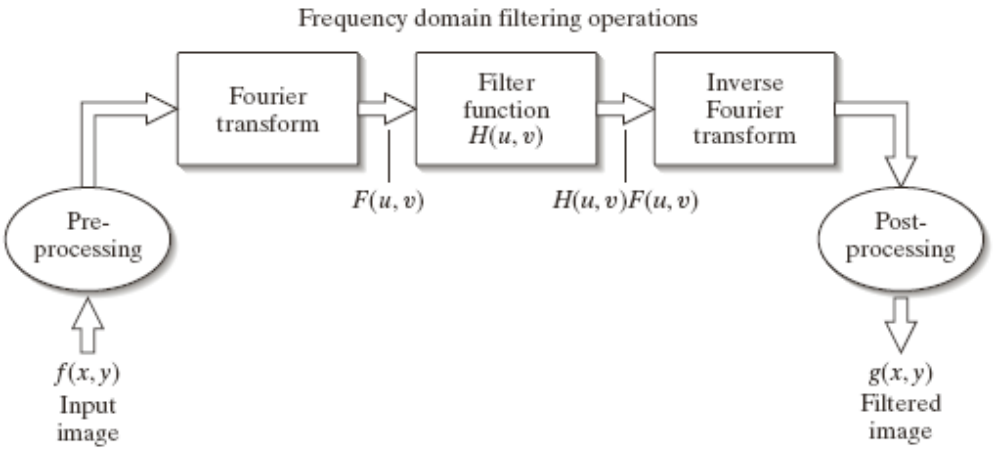


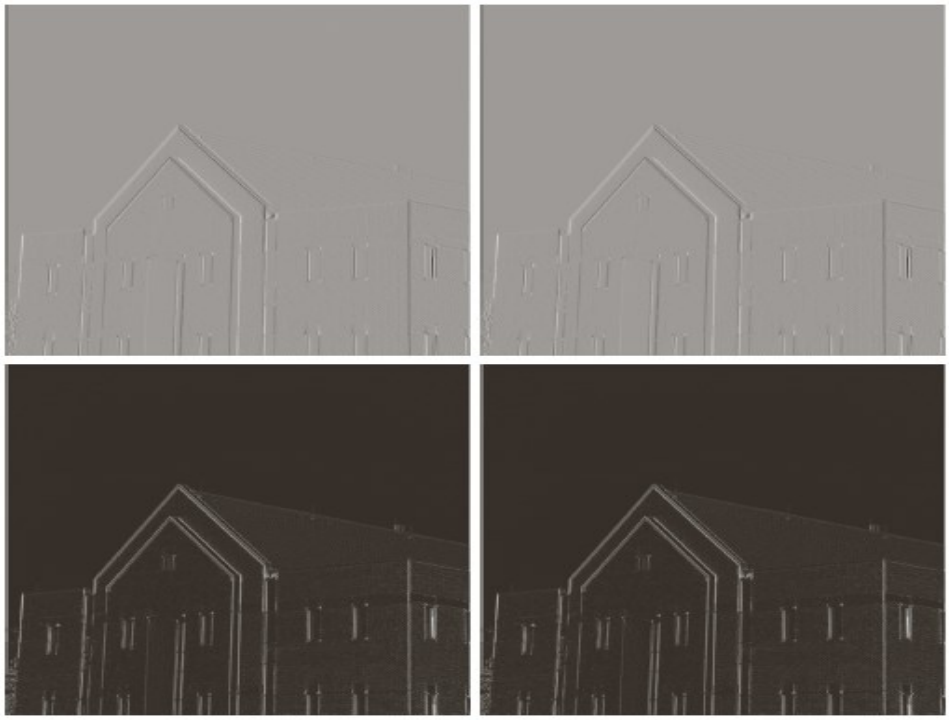
FIGURE 4.8 Basic steps for filtering in the frequency domain.

a b

FIGURE 4.9
(a) A gray-scale image. (b) Its Fourier spectrum.

a b
c d

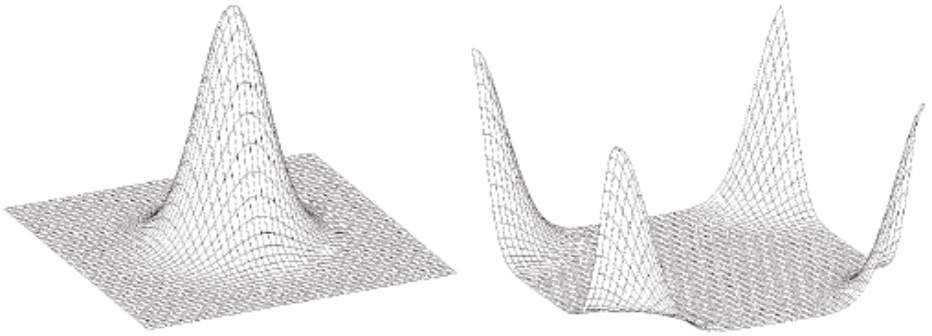
FIGURE 4.11
(a) Result of filtering Fig. 4.9(a) in the spatial domain with a vertical Sobel mask. (b) Result obtained in the frequency domain using the filter shown in Fig. 4.10(b). Figures (c) and (d) are the absolute values of (a) and (b), respectively.



Filtri u frekvencijskoj domeni

- Niskopropusni filtar: zaglađivanje slike
- Visokopropusni filtar: izoštravanje slike

- Tehnike selektivnog filtriranja



a b

FIGURE 4.4 Transfer functions of (a) a centered lowpass filter, and (b) the format used for DFT filtering. Note that these are frequency domain filters.

Niskopropusno filtri u frekvencijskoj domeni (NP)

Vrše zaglađivanje, a idealni NP filtar ima funkciju prijenosa:

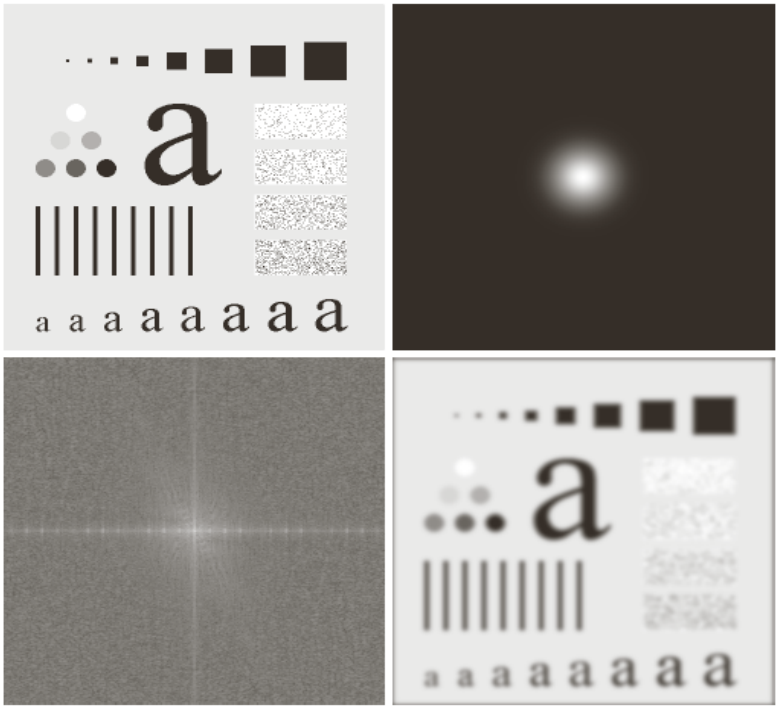
$$H(u,v)=\begin{cases}1 & ako \quad D(u,v)\le D_0\\ 0 & ako \quad D(u,v)>D_0\end{cases}$$

Gdje je D_0 pozitivni broj, a D(u,v) udaljenost od točke (u,v) do centra filtra.

Jedni od poznatijih su Butterworthov NP filtar (BLPF) i Gausov NP filtar (GLPF).

TABLE 4.1 Lowpass filters. D_0 is the cutoff frequency and n is the order of the Butterworth filter.

Ideal	Butterworth	Gaussian
$H(u,v)=\begin{cases}1 & \text{if } D(u,v)\le D_0\\ 0 & \text{if } D(u,v)>D_0\end{cases}$	$H(u,v)=\dfrac{1}{1+\left[D(u,v)/D_0\right]^{2n}}$	$H(u,v)=e^{-D^2(u,v)/2D_0^2}$



a b
c d
FIGURE 4.13 Lowpass filtering. (a) Original image. (b) Gaussian lowpass filter shown as an image. (c) Spectrum of (a). (d) Filtered image.

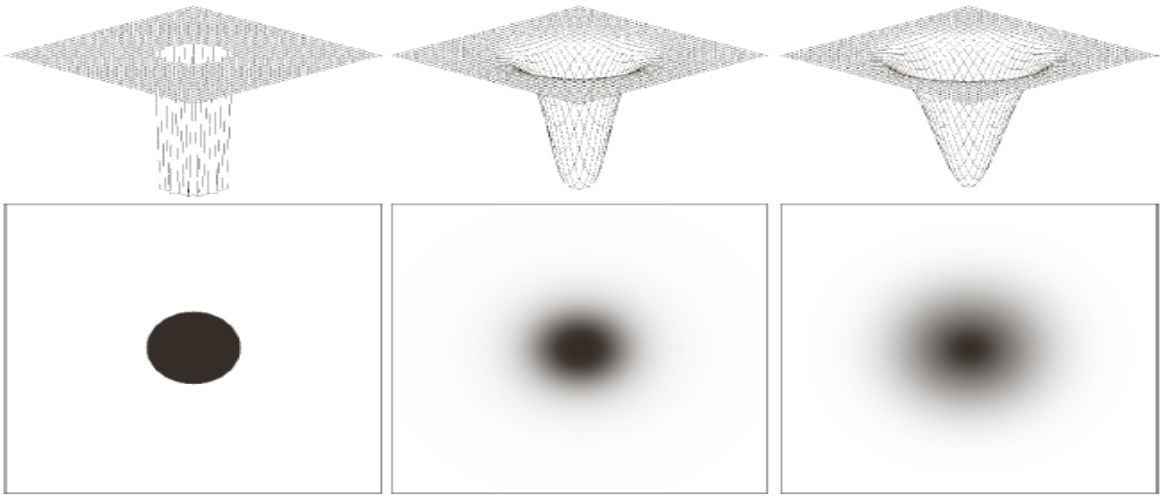
Visokopropusni filtri u frekvencijskoj domeni (VP)

VP filtri vrše izoštravanje slike, na način da prigušuje niske frekvencije, a visoke frekvencije Fourierove transformacije ostavljaju nepromijenjenima. Ako je $H_{LP}(u,v)$ prijenosna funkcija niskopropsunog filtra, onda je prijenosna funkcija visokopropusnog filtra:

$$H_{HP}(u,v) = 1 - H_{LP}(u,v)$$

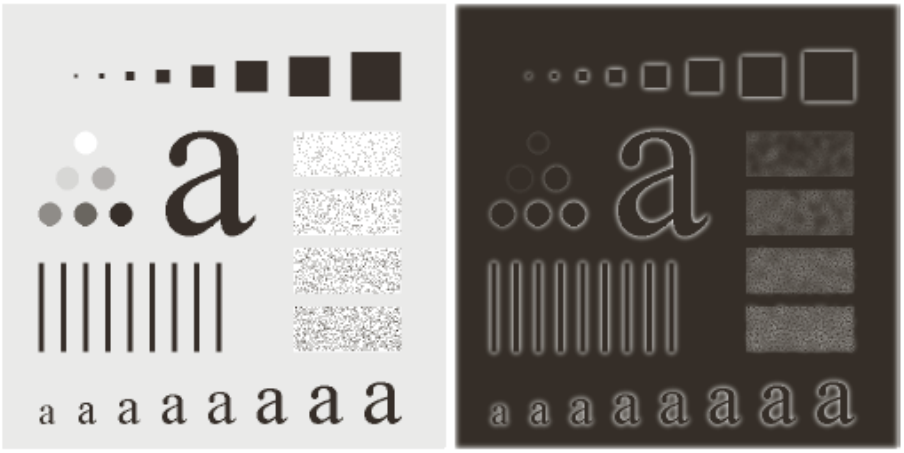
TABLE 4.2 Highpass filters. D_0 is the cutoff frequency and n is the order of the Butterworth filter.

Ideal	Butterworth	Gaussian
$H(u,v) = \begin{cases} 0 & \text{if } D(u,v) \le D_0 \\ 1 & \text{if } D(u,v) > D_0 \end{cases}$	$H(u,v) = \dfrac{1}{1 + \left[D_0/D(u,v)\right]^{2n}}$	$H(u,v) = 1 - e^{-D^2(u,v)/2D_0^2}$



a b c
d e f

FIGURE 4.17 Top row: Perspective plots of ideal, Butterworth, and Gaussian highpass filters. Bottom row: Corresponding images. White represents 1 and black is 0.



a b

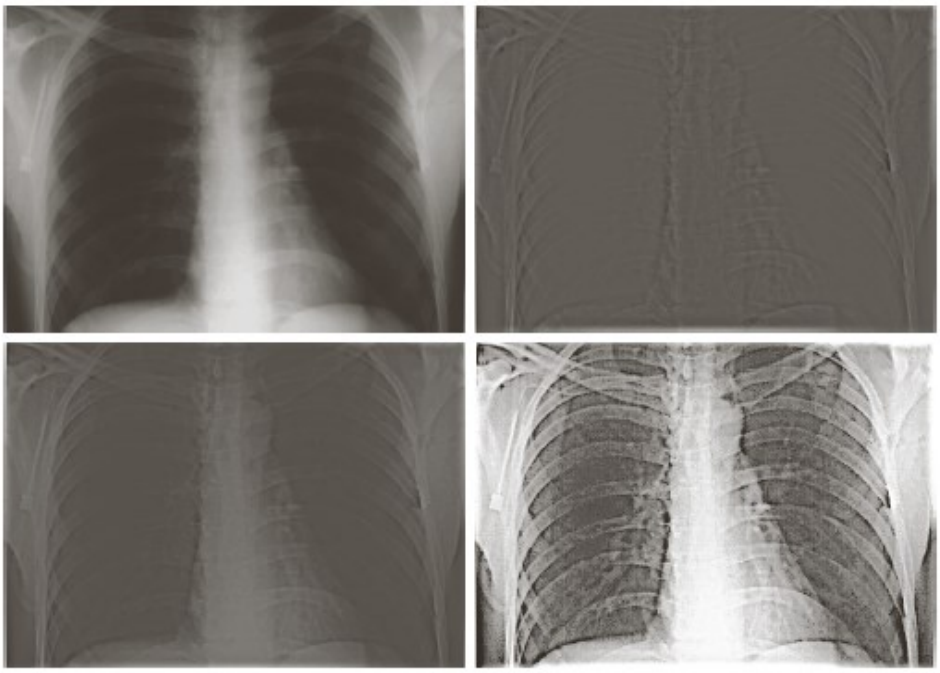
FIGURE 4.18
(a) Original image. (b) Result of Gaussian high-pass filtering.

Filtriranje s naglaskom na visoke frekvencije (HFE)

VP filtri reduciraju prosječnu vrijednost inteziteta u slici na 0 (izvlače samo rubove, ostalo je crno). Da bi se to kompenziralo uvodi se offset:

$$H_{HFE}(u,v) = a + bH_{HP}(u,v)$$

Konstanta koja množi pojačava visoke frekvencije. Također se pojačavaju i niske frekvencije, ali su one u odnosu na komponente visoke frekvencije neznatne i neprimjetne. a – offset, b – konstanta koja množi.



a b
c d

FIGURE 4.19 High-frequency emphasis filtering. (a) Original image. (b) Highpass filtering result. (c) High-frequency emphasis result. (d) Image (c) after histogram equalization. (Original image courtesy of Dr. Thomas R. Gest, Division of Anatomical Sciences, University of Michigan Medical School.)

Selektivno filtriranje

Za selektivno filtriranje koristi se pojasno propusni/nepropusni filtar. Filtar koji je pojasno propusan ili nepropusan pri cemu je taj pojas ekstremno uzak tako da bi u idelnom slucaju bio kao jedna linija naziva se notchpass ili notchband filtar. Idealno samo guši ili propušta jednu frekvenciju a ne pojas frekvencija.

TABLE 4.3 Bandreject filters. W is the "width" of the band, $D(u,v)$ is the distance from the center of the filter, D_0 is the radius of the center of the band, and n is the order of the Butterworth filter.

Ideal	Butterworth	Gaussian
$H(u,v)=\begin{cases}0 & \text{for } D_0-\frac{W}{2}\le D(u,v)\le D_0+\frac{W}{2}\\ 1 & \text{otherwise}\end{cases}$	$H(u,v)=\dfrac{1}{1+\left[\dfrac{WD(u,v)}{D^2(u,v)-D_0^2}\right]^{2n}}$	$H(u,v)=1-e^{-\left[\frac{D^2(u,v)-D_0^2}{WD(u,v)}\right]^2}$

a b

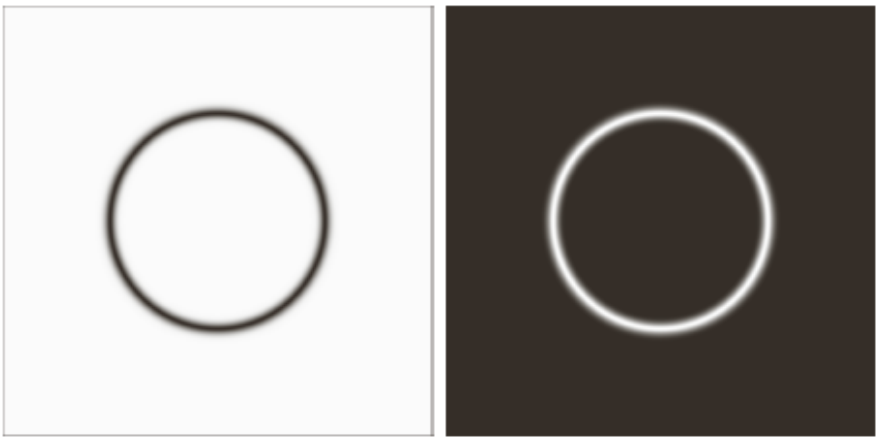


FIGURE 4.20
(a) A Gaussian bandreject filter. (b) Corresponding bandpass filter. The filters were generated using $M = N = 800$, $D_0 = 200$, and $W = 20$ in function `bandfilter`.

7. OBNAVLJANJE I REKONSTRUKCIJA SLIKA

Tehnike poboljšanja slike su heurističke, bazirane na manipulaciji slike u svrhu iskorištavanja psihofizičkog aspekta ljudskog vizualnog sustava.

Model degradacije/obnavljanja slika

Degradacija slike je generativni proces koji se modelira kroz funkciju degradacije kojoj je nadodan šum:

$$g(x,y) = H[f(x,y)] + \eta(x,y)$$

Ako znamo g(x,y), ponešto o H i ponešto o šumu funkcija cilja je estimacija originalne slike. Cilj je biti što bliže originalnoj slici.

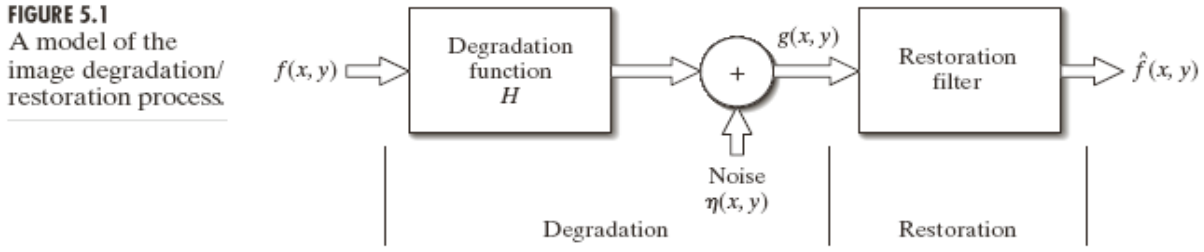


FIGURE 5.1
A model of the image degradation/ restoration process.

H je linearni, prostorno neovisni proces, a u prostornoj domeni degradirana slika dobivena generativnim procesdom je:

$$g(x,y) = h(x,y) * f(x,y) + \eta(x,y)$$

Gdje je h(x,y) prostorna reprezentacija funkcije degradacije.

U frekvencijskoj domeni, generativni proces opisuje se relacijom:

$$G(u,v) = H(u,v) * F(u,v) + N(u,v)$$

gdje velika slova označavaju Fourierove transformacije odgovarajućih oznaka u prostornoj domeni.

Slika 29. Generativni proces dobivanja slike niske rezolucije.

Model šuma (generiranje i dodavanje šuma na sliku)

Najvažniji dio restauracije slike je sposobnost simuliranja ponašanja šuma i efekta koji on ostavlja na slici. Šum nije ovisan o koordinatama slike.

Za Image Processing Toolbox funkcije za rad sa šumom su:

g = imnoise(f, type, parameters)

r = imnoise2(type, varargin)

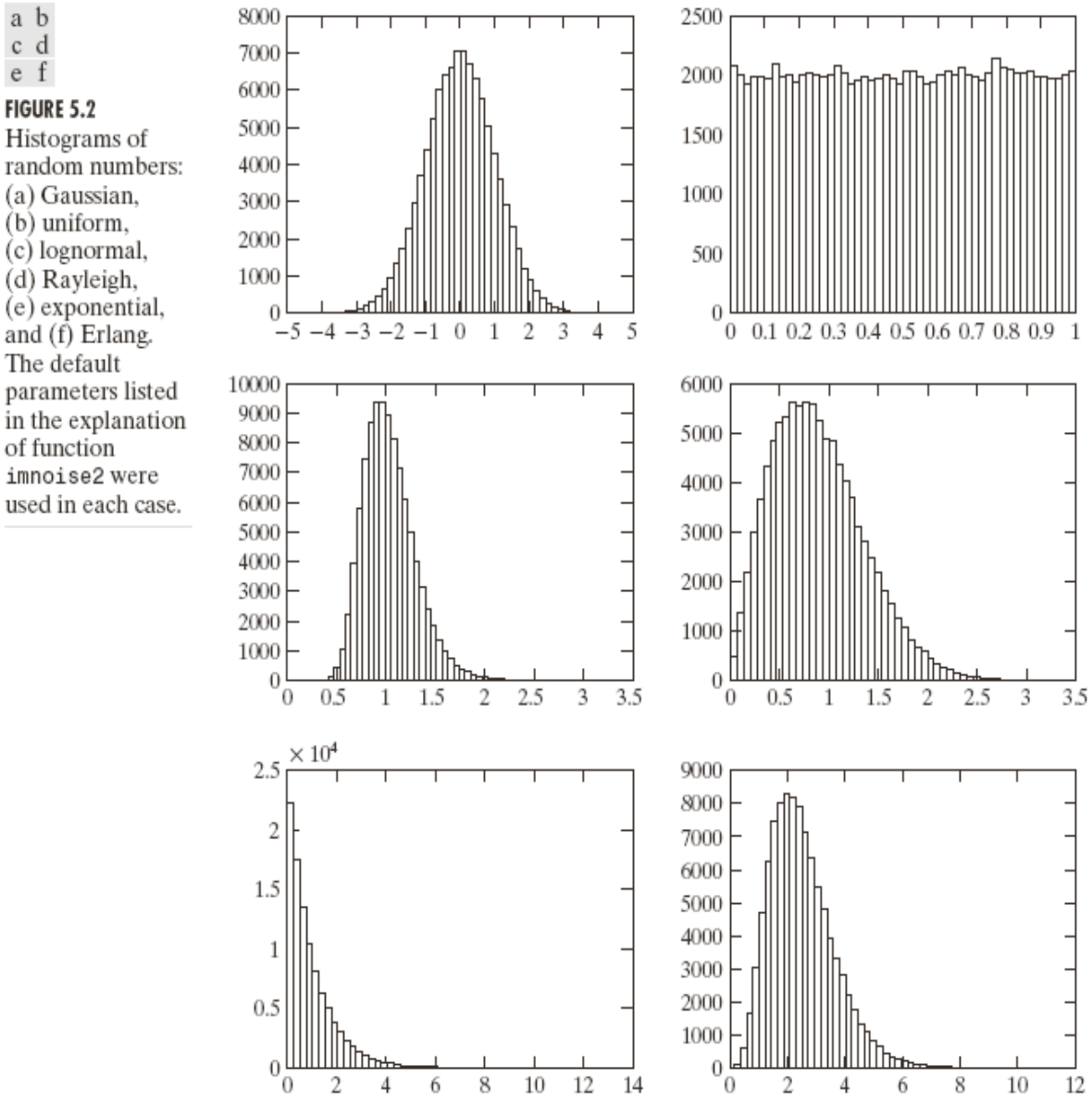
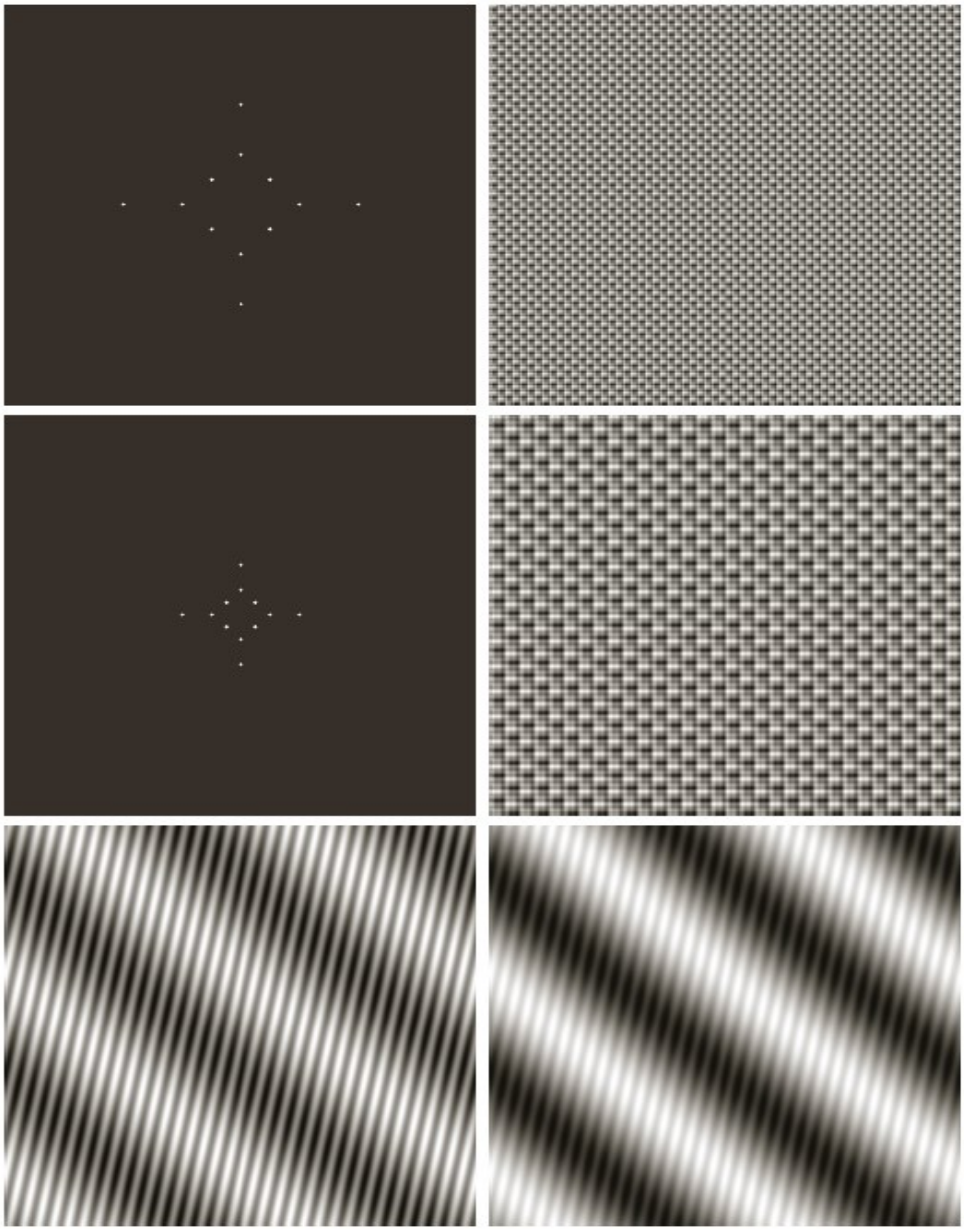


FIGURE 5.2
Histograms of random numbers: (a) Gaussian, (b) uniform, (c) lognormal, (d) Rayleigh, (e) exponential, and (f) Erlang. The default parameters listed in the explanation of function `imnoise2` were used in each case.

Slika 30. Histogram slučajnih brojeva. Podaci su generirani upotrebom funkcije imnoise2.

Periodičan šum

Periodičan šum je rezultat električne i/ili elektroničko – mehaničke interferencije tijekom akvizicije slike. On je i jedini prostorno ovisni šum. Izbjegava se filtriranjem u frekvencijsku domenu.



a b
c d
e f

FIGURE 5.3
(a) Spectrum of specified impulses. (b) Corresponding sine noise pattern in the spatial domain. (c) and (d) A similar sequence. (e) and (f) Two other noise patterns. The dots in (a) and (c) were enlarged to make them easier to see.

Slika 31. Spektar i uzorak prostornog sinusnog šuma .

Estimiranje parametara šuma

Uobičajen način za estimiranje parametara šuma je analizom Fourierovog spektra. Periodičan šum uzrokuje frekvencijske šiljke, koji se čak mogu uočiti vizualnom inspekcijom. Estimiranje parametara direktno iz slike vrši se kroz selekciju pozadine bez objekata (region of interest – ROI), tako da varijabilnosti u tom dijelu slike budu poslijedica šuma:

B = roipoly(f, c, r)

gdje je f slika, a c i r vektori odgovarajućih koordinata stupaca i redova.

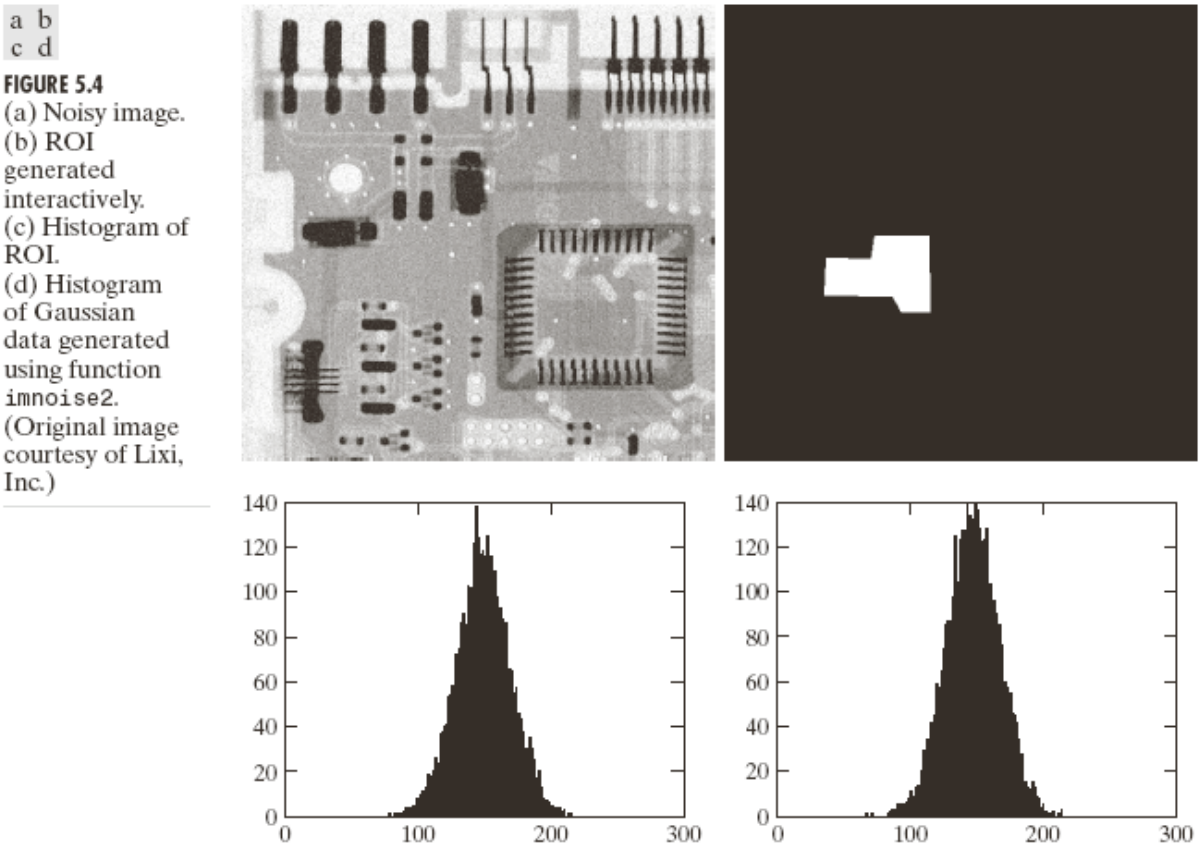


FIGURE 5.4
(a) Noisy image. (b) ROI generated interactively. (c) Histogram of ROI. (d) Histogram of Gaussian data generated using function `imnoise2`. (Original image courtesy of Lixi, Inc.)

Obnavljanje slike degradirane jedino šumom – prostorno filtriranje

U ovom slučaju generativni model glasi:

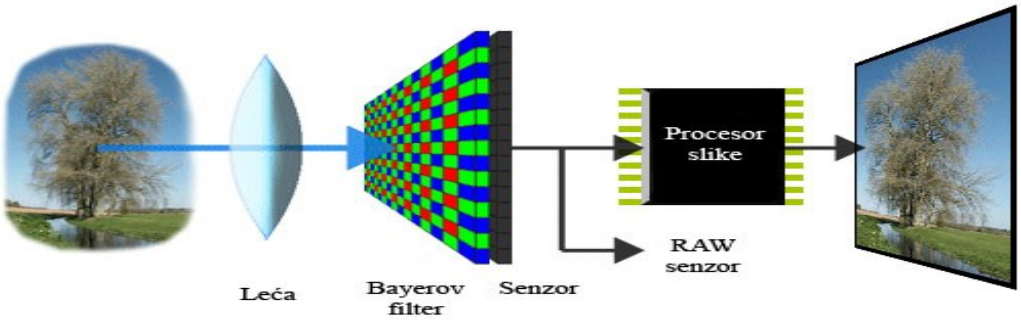
$$g(x,y) = f(x,y) + \eta(x,y)$$

Korištena metoda je prostorno filtriranje, a koriste se filteri za prostorno filtriranje i adaptivni prostorni filteri.

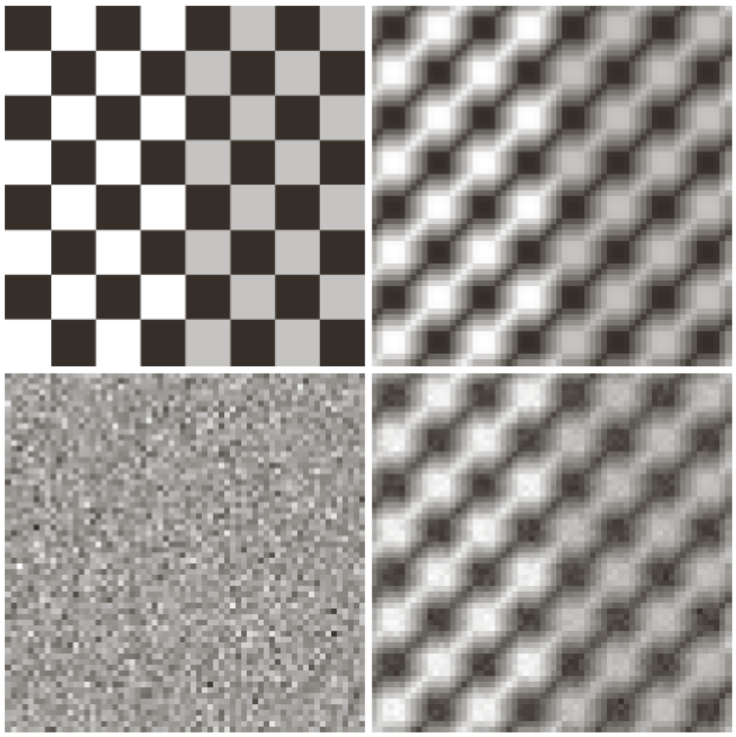
Generativni model degradacije

Postoji još jedan problem kod akvizicije (tj. u generativnom modelu) slike, a to je zamućenje na krajevima leće ili zamućenje zbog relativnog gibanja između scene i senzora.

Image Processing Toolbox za generiranje zamućenja koristi *PSF = fspecial('motion', len, theta).*



Slika 32. Proces akvizicije slike.



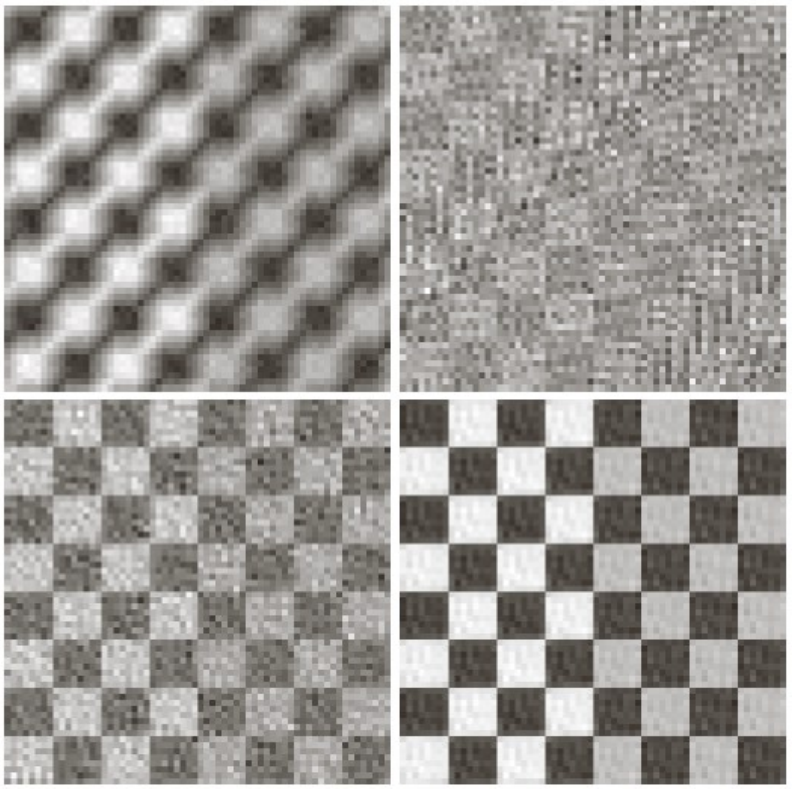
a b
c d
FIGURE 5.7
(a) Original image. (b) Image blurred using `fspecial` with `len = 7`, and `theta = -45` degrees. (c) Noise image. (d) Sum of (b) and (c).

Wiener filter

Ovaj filter osmišljen je već 1942.g i još uvijek je jedan od najboljih načina linearne restauracije slike. Njegov način rada sažet je kroz relaciju:

$$e^2 = E\{(f - \hat{f})^2\}$$

Gdje je *E* vrijednosni operator, *f* je originalna slika.



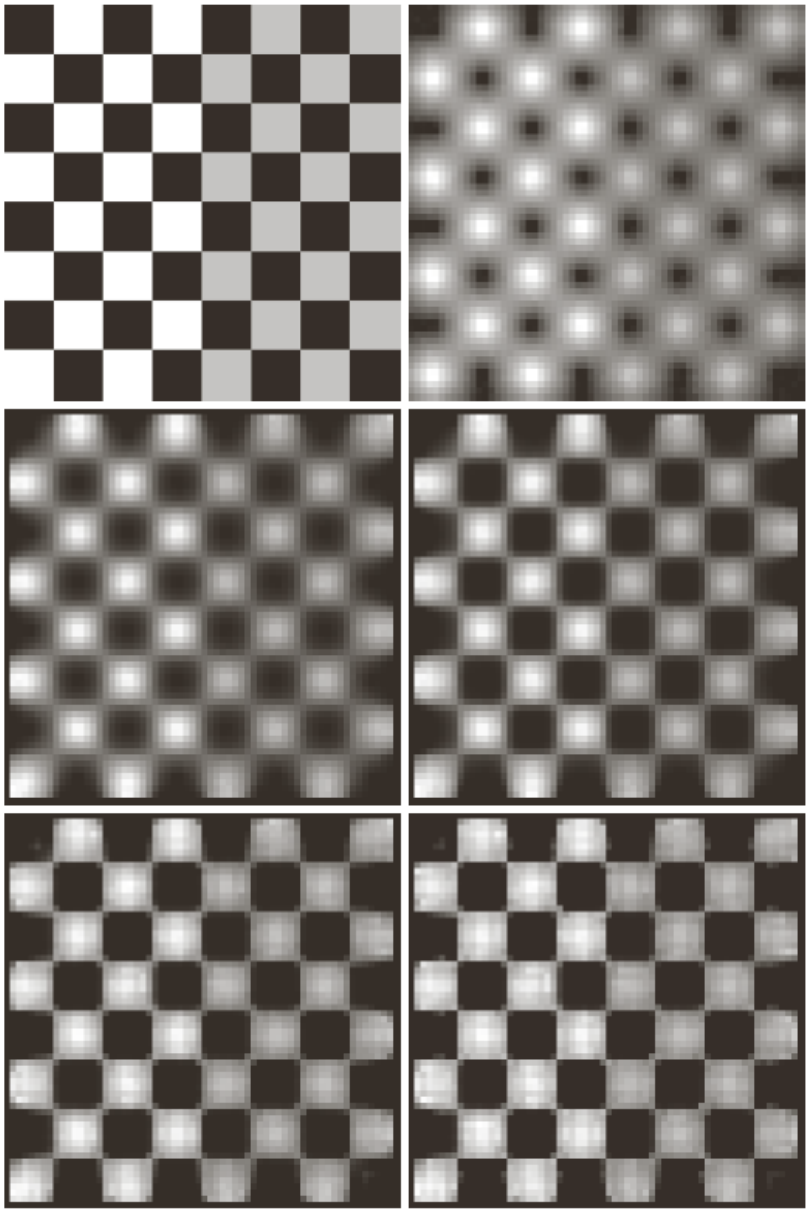
a b
c d

FIGURE 5.8
(a) Blurred, noisy image. (b) Result of inverse filtering. (c) Result of Wiener filtering using a constant ratio. (d) Result of Wiener filtering using autocorrelation functions.

Lucy – Richardson algoritam

Lucy – Richardson algoritam je nelinearna iterativna tehnika restauracije slike. Daje bolje rezultate od linearnih tehnika restauracije. Richardson (1972) i Lucy (1974) su radili neovisno jedan o drugome pa se postavlja pitanje bi li se trebao zvati Lucy – Richardson ili Richardson – Lucy. ☺

Proizlazi iz Maximum Likelihood statistike, gdje se slika modelira s Poissonovom statistikom.



a b
c d
e f

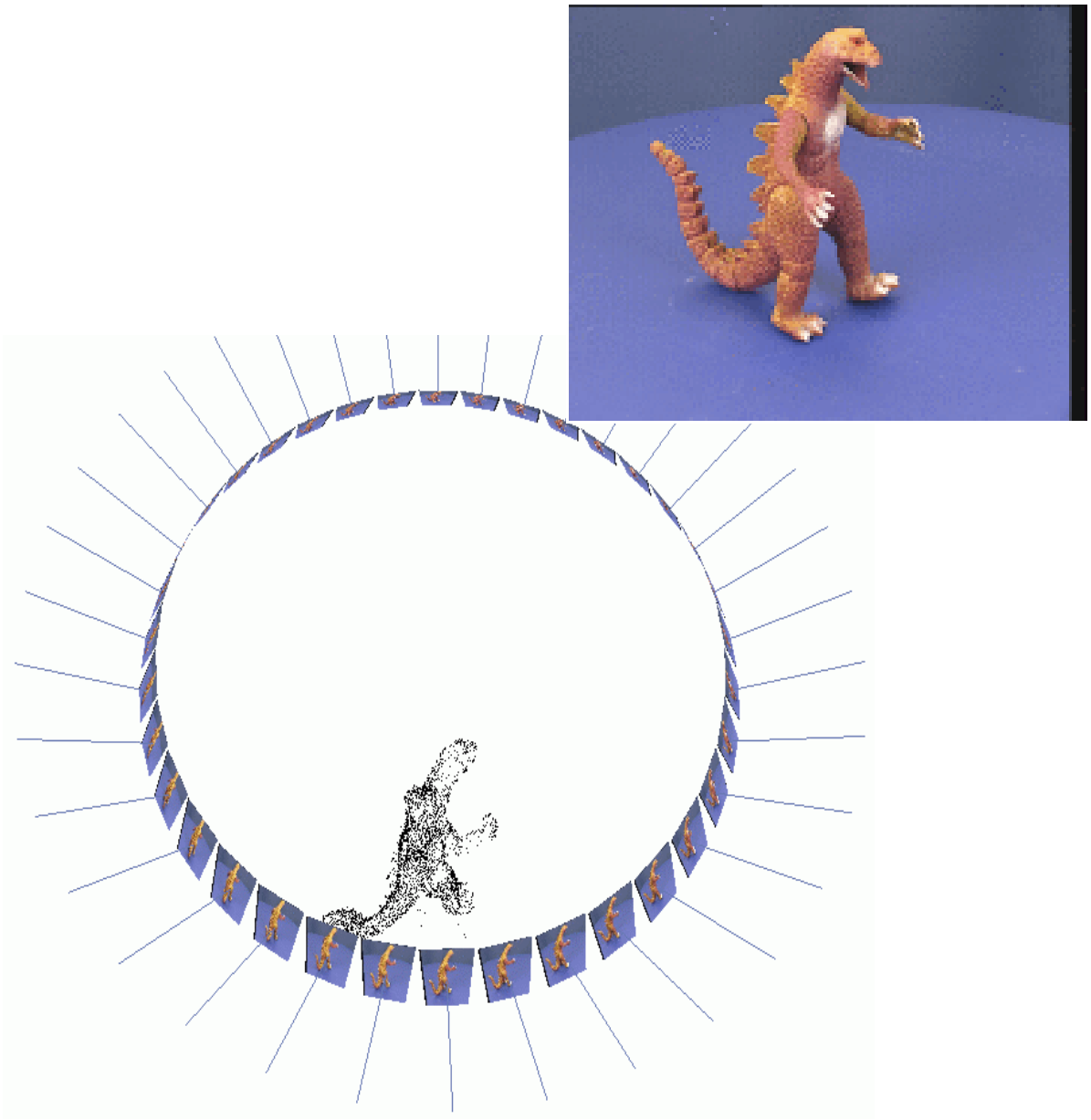
FIGURE 5.10 (a) Original image. (b) Image blurred and corrupted by Gaussian noise. (c) through (f) Image (b) restored using the L-R algorithm with 5, 10, 20, and 100 iterations, respectively.

Slijepa dekonvolucija

Za konvoluciju kažemo da je slijepa kada funkcija raspršenja točke (PSF) nije poznata. Ovje se koristi statistička metoda *Maximum Likelihood Estimation* koja se rješava iterativno.

N-slikovna restauracija

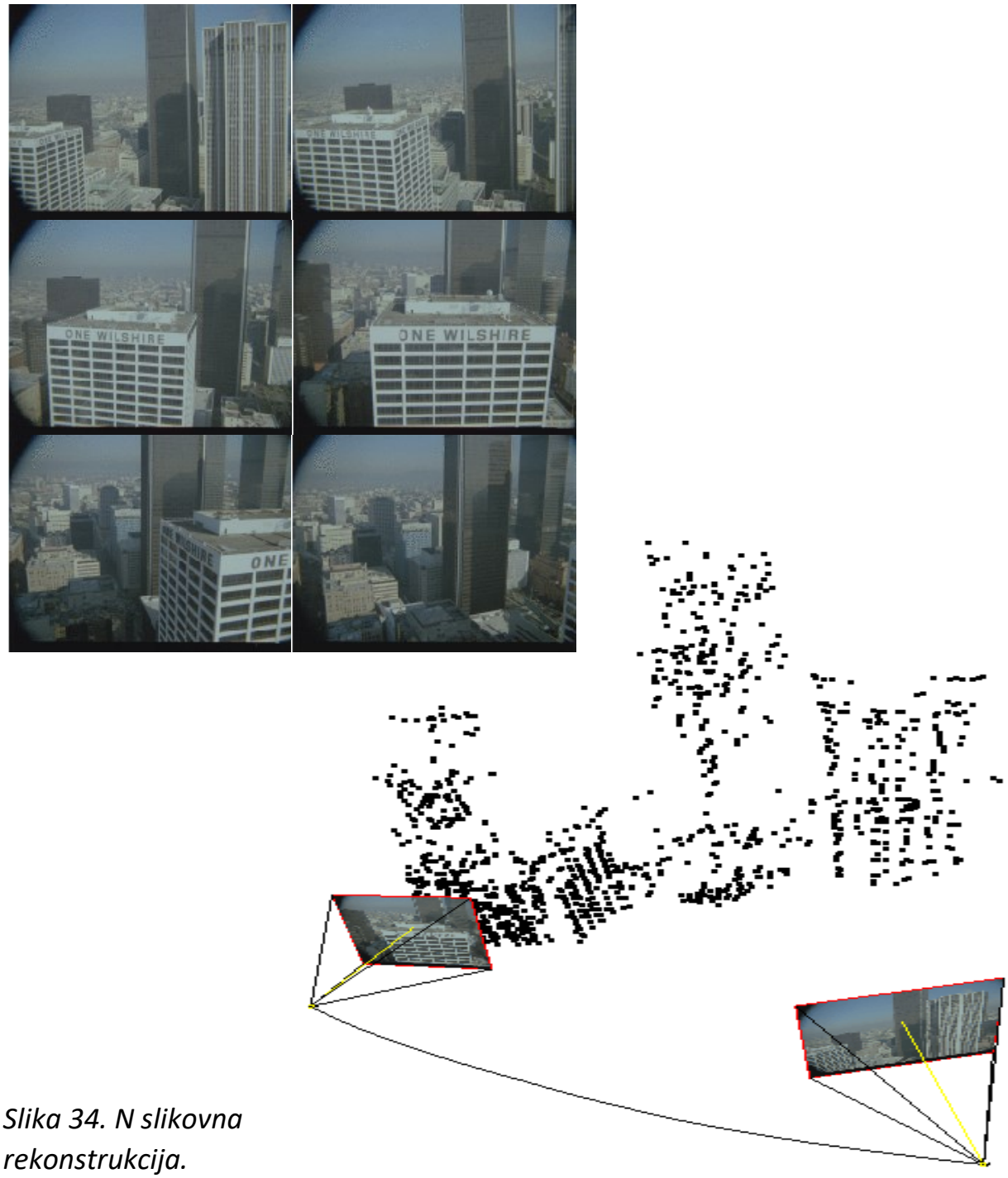
Za restauraciju slike iz N slika najprikladnije je korištenje afine kamere tj. afinih transformacija, koje se i najčešće koriste za velike udaljenosti scene. Zahtjev da sve točke moraju biti vidljive u svim slikama mora biti ispunjen. Faktorizacijski algoritam koji se koristi je *Singular Value Decomposition*, a iterativna metoda *Grupirajuće podešavanje.*



Slika 33. Rekonstrukcija iz N slika.

Geometrija iz četiri slike i n-slikovna rekonstrukcija

Upotreba n – slikovnih rekonstrukcijskih tehnika omogućava automatsko izvođenje rekonstrukcije iz poprilično dugačkih sekvenci slika.



Slika 34. N slikovna rekonstrukcija.

8. KOMPRESIJA SLIKA I VIDEA

Kompresija se odnosi na smanjenje informacija potrebnih za prikaz neke slike. Postiže se odstranjenjem jedne od tri osnovne redundancije:

- redundancije koda
- prostorne i/ili vremenske redundancije (sekvence slika)
- irelevantnih informacija, tj. informacija koje ljudsko oko samo po sebi odbacuje zbog nesavršenosti

Pozadina

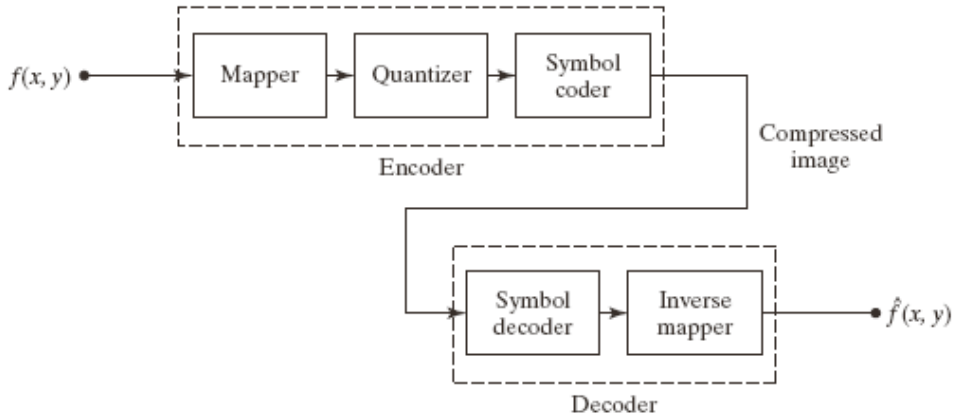


FIGURE 9.1
A general image compression system block diagram.

Ako n1 i n2 označavaju jedinice nositelja informacije (bitova), u originalnoj i kodiranoj slici, respektivno, onda se kompresija može izraziti preko omjera:

$$C_R = \frac{n1}{n2}$$

Kodiranje redundancije - Huffmanovo kodiranje

Kodiranje redundancije je najčešće kada se sivi nivoi slike kodiraju prirodnim binarnim kodom. Huffmanovo kodiranje je kodiranje bez

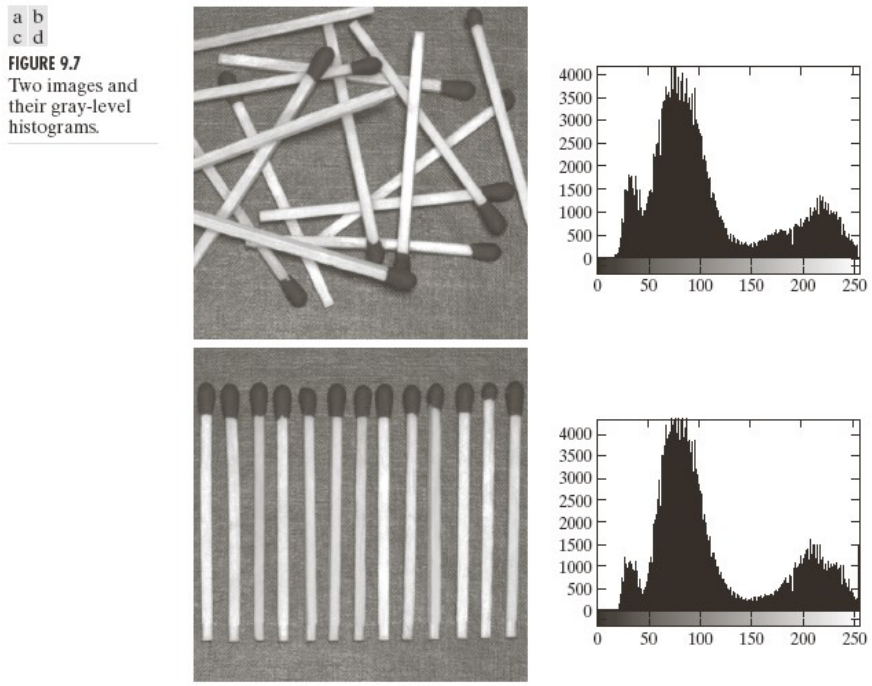
gubitaka. Simboli se kodiraju obzirom na to koliko se često pojavljuju (eng. *variable length coding*). Metoda stabla binarno dodjeljuje kod.

Tablica 2. Primjer Huffmanovog kodiranja.

A	1
B	6
C	7
D	2
E	8

Prostorna redundancija

Slike imaju virtualno identične histograme (trimodalne), otkrivajući postojanje tri dominantna ranga sivih nivoa, koji se u potpunosti ne preklapaju.

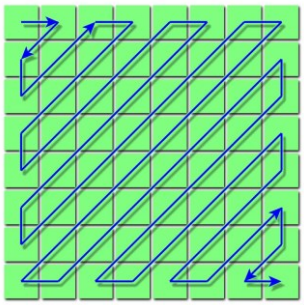


Irelevantne informacije

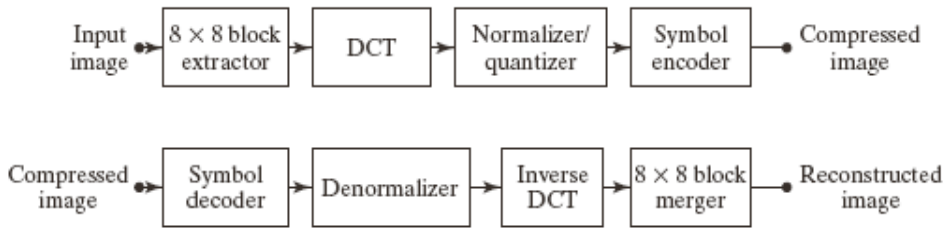
Irelevantne informacije su one informacije koje ljudski vizualni sustav ne može percipirati. Uklanjanje irelevantnih informacija naziva se kvantizacija, a rezultat je gubitak kvantitete, ne kvalitete. Kodiranje s gibicima je ireverzibilni proces, a postiže se poboljšanje kvantizacije sive skale.

JPEG

JPEG je standard razvijen 1992.g. Cilj mu je bio digitalna kompresija i kodiranje kontinuiranih mirnih slika (*Digital compression and coding of continuous – tone still images).* Osnovni JPEG mod je mod s gubitkom dijela informacije. Koristi diskretnu kosinusnu transfromaciju (DCT), što je kodiranje bez gubitaka. Kvantizacija smanjuje broj bita potrebnih za prijenos koeficijenata što dovodi do gubitaka. Također, koristi i ZigZag skeniranje, gdje kvantizirani DCT koeficijenti imaju vrijednosti 0 u donjem desnom dijelu matrice. ZigZag mapira 8x8 vektor DCT koeficijenata u 1x64 vektor. Manje važni koeficienti (0) su na kraju vektora. RLC se vrši na AC komponentama, a nule u vektoru se preskaču.



Slika 35. ZigZag mapiranje.



a
b

FIGURE 9.11
JPEG block diagram:
(a) encoder and
(b) decoder.

a b

FIGURE 9.12
(a) The JPEG default normalization array. (b) The JPEG zigzag coefficient ordering sequence.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

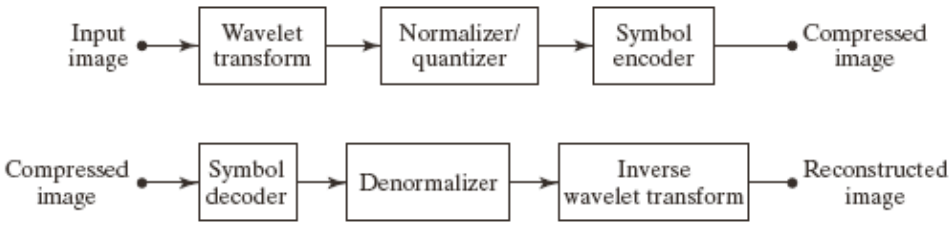
Nadalje, koristi se entropijsko kodiranje koeficijenata (Huffman). Ovaj način kodiranja je mod bez gubitaka. Nema gubitaka u kvaliteti. Kodira se i prenosi samo razlika u odnosu na stvarnu vrijednost, a samo se prvi piksel prenosi kao stvarna vrijednost.

JPEG 2000

JPEG 2000 umjesto DCT-a koristi wavelet transformaciju.

a
b

FIGURE 9.14
JPEG 2000 block diagram: (a) encoder and (b) decoder.



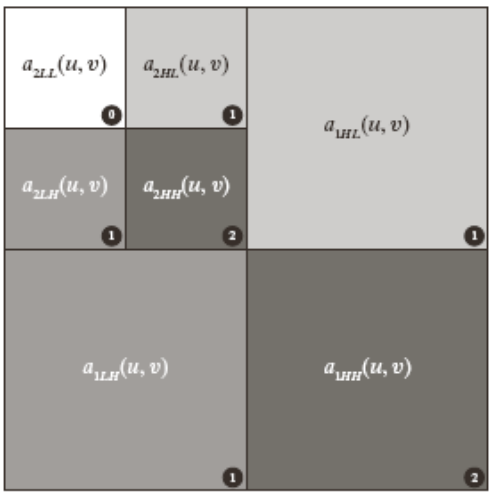
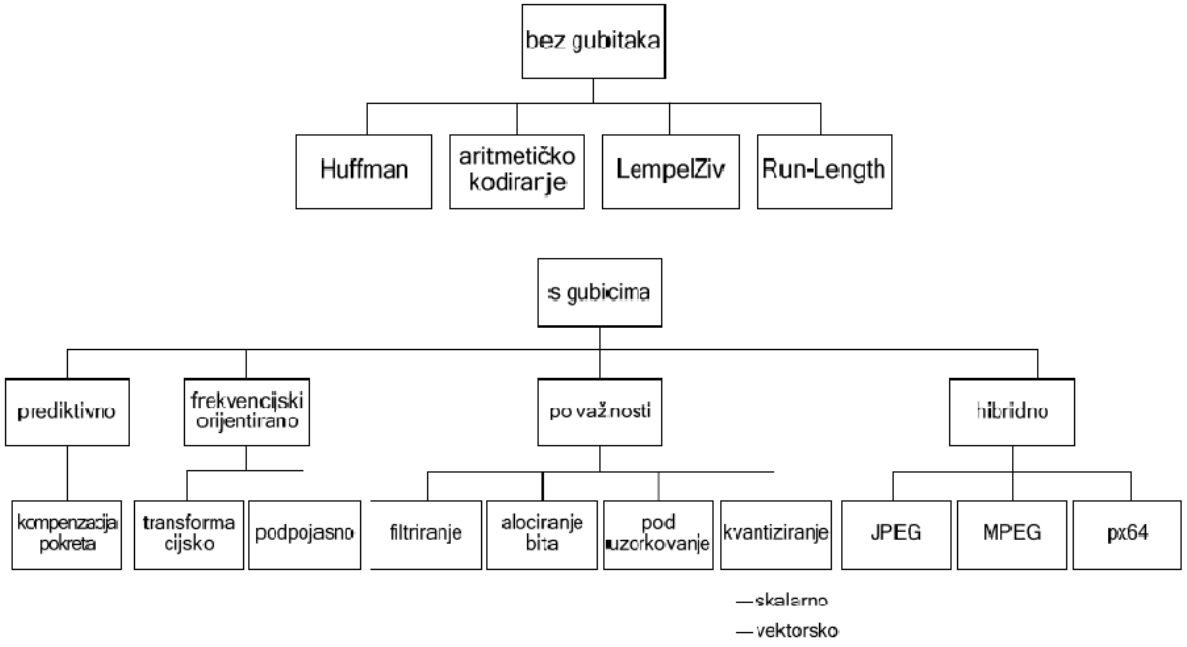


FIGURE 9.15
JPEG 2000 two-scale wavelet transform coefficient notation and analysis gain (in the circles).

Video kompresija

Niz brzo mijenjajućih okvira (*frameova*) zahtijeva prijenos jako velike količine podataka preko interneta. Kompresijske tehnike mogu se podijeliti na:

- kompresija s gubicima (lossy)
- kompresija bez gubitaka (lossless)



Slika 36. Podijela kompresijskih tehnika.

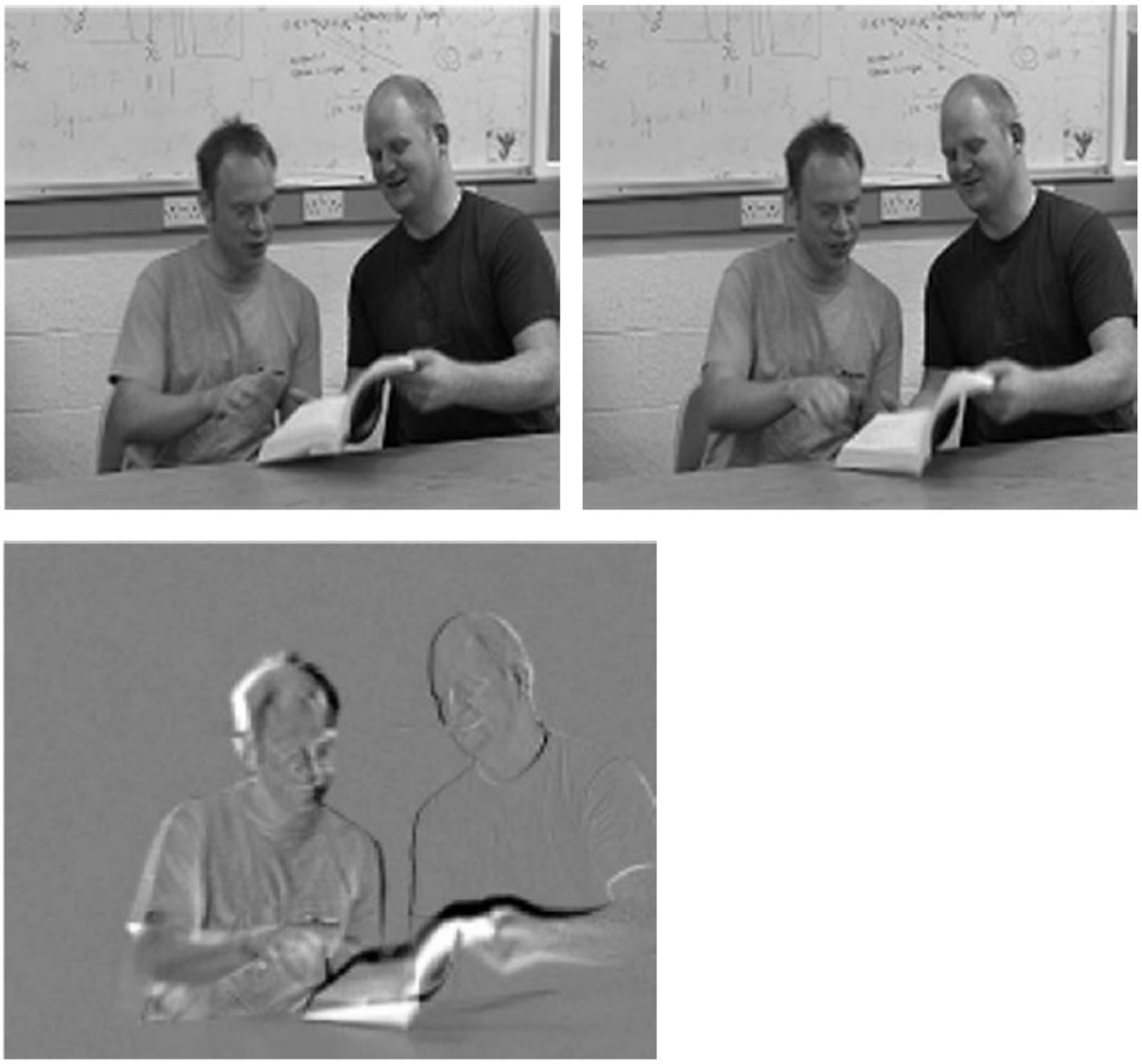
Video koder/dekoder

Video koder uklanja vremensku i prostornu redundanciju zbog velike korelacije susjednih okvira, obzirom da susjedni pikseli imaju slične ili iste vrijednosti. Sastoji se od modela vremena, modela prostora i entropijskog kodera.

Model vremena

Ulaz u model je nekomprimirani video. Model koristi sličnost susjednih okvira, tako da predvidi trenutni okvir, vrši kompenzaciju pokreta, a na

izlazu daje rezidual, odnosno razliku predviđenog i stvarnog okvira kao i set parametara koji određuju vektor pokreta.



Slika 37. Dva susjedna framea i njihova razlika.

Model prostora

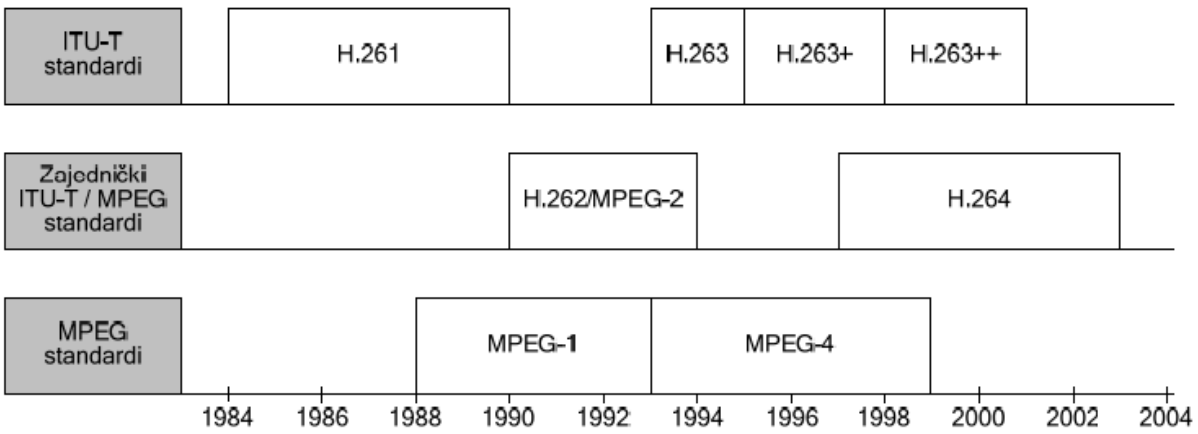
Vrši transfromaciju u područje u kojem je lakše vršiti kvantizaciju, te izmjenjuje redosljed koeficijenata po njihovoj važnosti. Obavlja i predviđanje u prostoru. Koristi diskretnu kosinusnu transformaciju (DCT) i diskretnu valnu transformaciju (*discrete Wavelet transform* - DWT).

Entropijski koder

Za zadatak ima prebacivanje simbola u oblik pogodan za prijenos, kao što su nule i jedinice.

Standardi za kompresiju video signala

- H.261
- H.263
- MPEG-1
- MPEG-2
- MPEG-4
- H.264/MPEG AVC
- H.265 i ostali



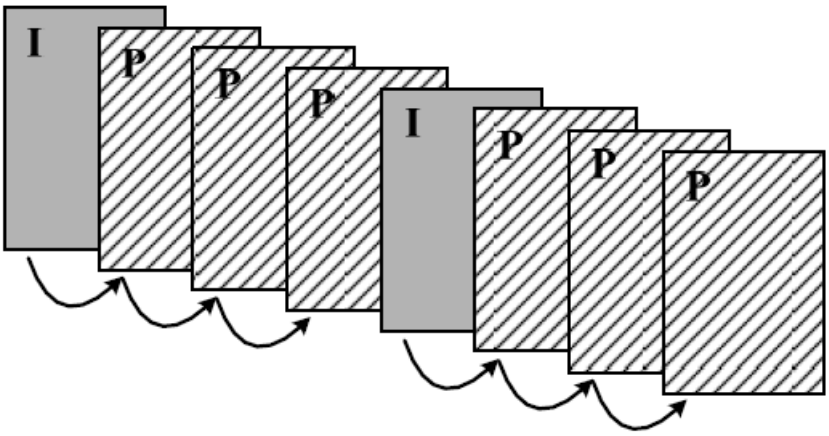
Slika 38. Kronološki razvoj kodera.

H.261

Video enkoder/dekoder za audio – vizualne potrebe s radom na brzini od px64Kb/s. Nerformalni naziv mu je px64. Njegov primarni dizajn mu je bio za videokonferencije i videotelefonije preko ISDN linija. Korisiti model boja YcbCr i 2 tipa kodiranja:

- Intra okvir (I) – DCT → mirna slika
- Inter okvir (P) – kodira se razlika u slikama/blokovima i to procjenom pokreta, te kompenzacijom pokreta na razini makrobloka (4 bloka od 8x8 piksela)

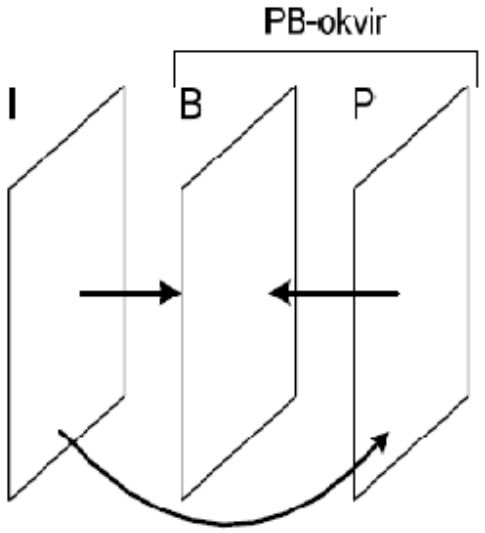
Dekoder procjenu pokreta vrši na rekonstruiranoj slici



Slika 39. Intra i Inter okvir.

H.263

H.263 je video standard za rad na malim brzinama. On je poboljšana verzija H.261 u smislu veće efikasnosti kompresije. Vektor pokreta moguće je i naći i izvan slike korištenjem rubnih piksela, za pokrete na rubu slike. U svakom makrobloku može koristiti do 4 vektora pokreta. Češće simbole kodira s manje bita. Kod PB okvira koncept “B” slike posuđen je od MPEG standarda.



Slika 40. I – P – B okvir kod H.263 standarda.

H.263+ i H.263++:

H.263+ i H.263++ su za fleksibilni video format. Njihova karakteristika je skalabilnost vremena, prostora kao i SNR. Ima poboljšani PB okvir, poboljšano intrakodiranje te filtar za uklananje efekta bloka (DF).

MPEG

(*Moving Picture Coding Expert Group)*

MPEG je nastao iz potrebe definiranja audio i video formata za novi medij – CD. MPEG-ova ima nekoliko i najbitniji su:

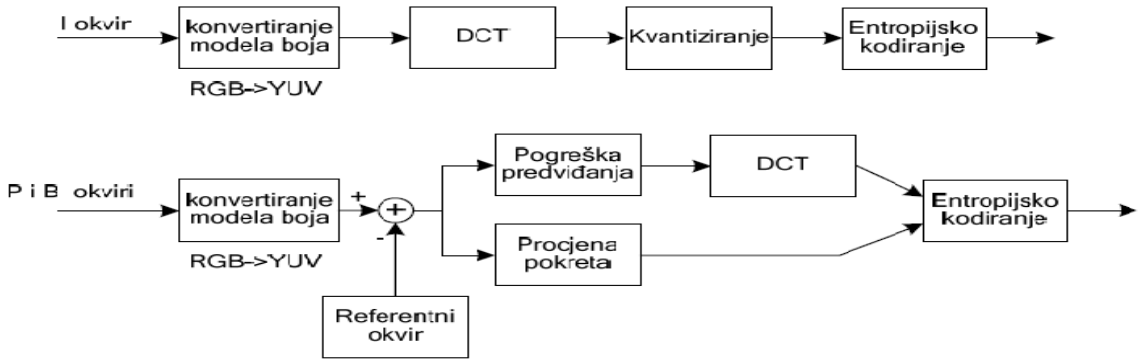
- Standard za transparentan i siguran prijenos multimedijskog sadržaja preko različitih mreža
  - **MPEG-21**
- Standard za pretraživanje, filtriranje, dohvaćanje i upravljanje multimedijskim sadržajem tako da je sadržaj opsan (bojom, teksturom, veličinom, opisom scene i sl.)
  - **MPEG-7**

- Standardi za kompresiju videa i audia
  - **MPEG-1, MPEG-2, MPEG-4**

    Sastoje se od 3 osnovna dijela: video, audio i sustav za sinkronizaciju i multipleksiranje videa i audia.
- **MPEG -3**? Za HDTV – što s njim?

MPEG – 1

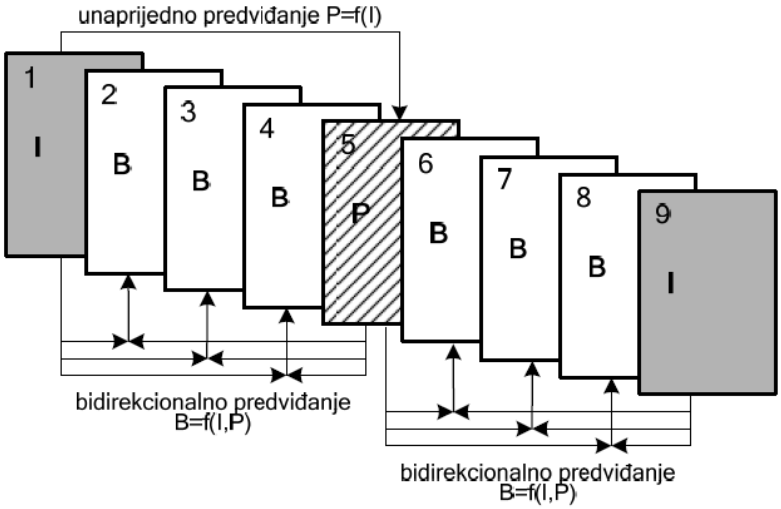
MPEG – 1 je standard za kodiranje pokretnih slika i pridruženog audia za digitalne medije pohrane na brzini do 1.5Mb/s. Podržava slike do veličine 4096x4096 s brojnim brzinama (okvir/s). Video dio standarda temelji se na eliminaciji prostorne(DCT) i vremenske redundancije (kompenzacija pokreta).



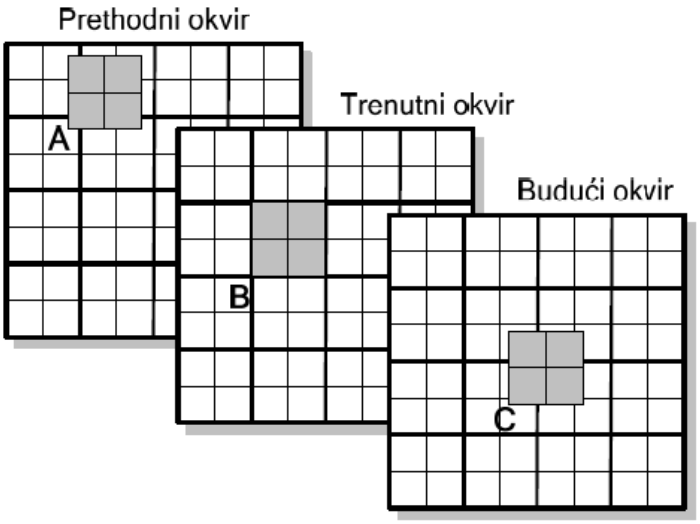
Slika 41. MPEG – 1 tok kodiranja.

Video sintaksa podržava 3 tipa kodiranih okvira:

- I okvir (DCT, najniža kompresija)
- P okvir (predicted frame, kodiran u odnosu na prošle I ili P)
- B okvir (najveća kompresija, komprimira se koristeći prošle i buduće I ili P okvire, Bidirectionally predictive coded frame)



Kod P i B okvira koristi se procjena pokreta

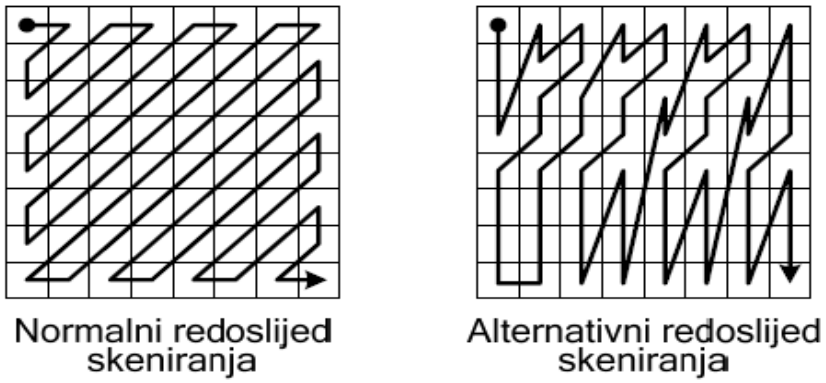


Slika 42. Okviri i procjena pokreta.

MPEG -2

MPEG – 2 je standard za digitalnu TV, šireg spektra primjena i veće kvalitete. Kodiranje videa s vrlo dobrom kvalitetom na brzini od oko 4 do 15 Mb/s. Podržava rezolucije 16384x16384, te brojne brzine. Koristi YCbCr model boja, diskretnu kosinusnu transformaciju (DCT) i

kompenzaciju pokreta. Također je dozvoljeno i korištenje alternativnog skeniranja.

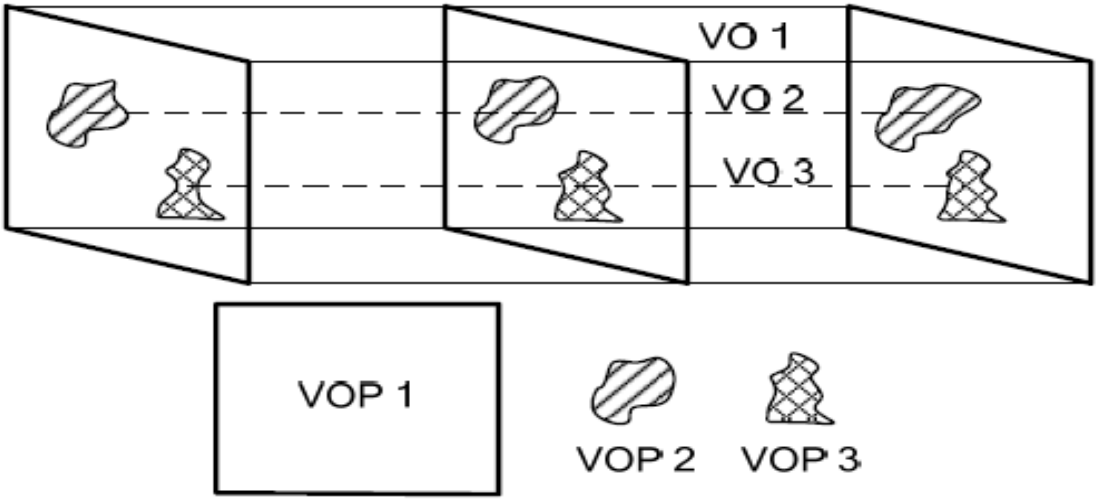


Slika 43. Normalni i alternativni slijed skeniranja piksela.

MPEG -4

MPEG – 4 standard je završen 1999. godine. To je standard koji omogućava manipulaciju s objektima (dijelovima slike). Napravljen je za ciljane aplikacije, kao računalne mreže, mobilne komunikacije, interaktivne aplikacije, vizualizacije, igre, internet. Kombinira elemente digitalne TV, interaktivne grafike i Weba. Dozvoljava odvojeno kodiranje svakog od objekata koji čine scenu. Video objekti i pozadina scene mogu se pojedinačno definirati i manipulirati.

Omogućava interaktivnost temeljenu na sadržaju, kompresiju temeljenu na sadržaju te univerzalni pristup za bežićne i druge mreže. Omogućava kodiranje prirodnog i sintetiziranog video signala. Temelji se na konceptu Video object plane (VOP) na način da vrši segmentaciju svakog frame-a u niz VOP-ova. Vremenski niz VOP-ova zove se video objekt (VO). Koristi i sintaksu za multipleksiranje i demultipleksiranje prijamnika da dekodira i združi različite VO u jedan frame.



Slika 44. Kodiranje svakog objekta i pozadine posebno.

H.264/MPEG AVC

Standard H.264 je po stupnju kompresije trenutno najbolji video koder. Njegovi sinonimi su MPEG -4 Part 10, H.264, MPEG AVC (*advanced video coding*), JVT i H.26L. MPEG AVC.

H.264 je koder opće namjene od mobilnog videa s niskim brzinama prijenosa do TV visoke rezolucije. Danas se koristi u DTV, DVD i sl.

Karakteristike pojedinih standarda:

- MPEG -2

  Kod intra okvira umjesto DCT koristi cjelobrojnu transformaciju (zbrajanje i shifting)

- H.264:

  Percepcija (poduzorkovanje krome, kvantizacija)

  Prostor (DCT)

  Vrijeme (procjena pokreta)

  Statistika (Huffmanovo kodiranje, aritmetičko kodiranje)

- MPEG2 vs H.264 (online video primjer)

⊡ H.265 (HVEC- high efficiency video coding; MPEG-H Part 2) – novo!!

OSTALI VIDEO KODERI

- VC – 1 (Windows Media Video) – ASF ili AVI kontejner
- RealVideo (videostreaming, .rm kontejner)
- DivX je varijanta MPEG-4 video standarda
- Audio Video Standard (AVS): Narodna Republika Kina
  - Cilj: smanjiti ovisnost o skupim stranim licencama
  - VLC video player

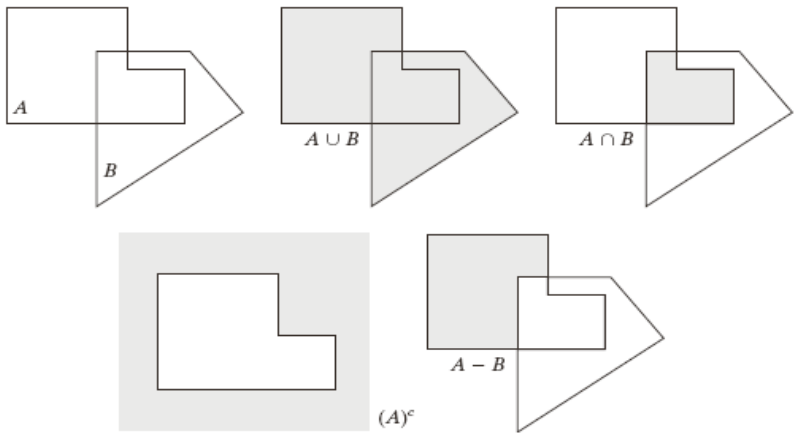
Formati video datoteka

Dok su koderi i dekoderi metode sažimanja i odpakiranja video datoteka, kontejneri su formati datoteke koji mogu sadržavati različite tipove podataka, komprimirane nekim od kodera. Napredniji kontejneri podržavaju: audio, video, titlove, meta-podatke.

- Poznatiji:
  - AVI (Audio Video Interleave - Microsoft)
  - MOV (Apple Quick Time)
  - MP4 (kontejner za MPEG-4)
  - ASF (kontejner za WMA i WMV)
  - RealMedia (RealVideo i RealAudio)
  - 3gp (pojednostavljena verzija MP4 za mobilne telefone)
  - FLV (Flash Video – vlasnički format za video distribuciju preko Interneta koristeći Adobe Flash Player)

9. MORFOLOŠKA OBRADA SLIKE

Termin matematička morfologija označava sredstvo za izvlačenje komponenti slike korisnih za reprezentaciju i opis oblika regije kao što su granice, kosturi i konveksna područja. Morfologija je kamen temeljac matematičkog aparata na kojima leži čitavo područje tehnika "izvlačenja" SMISLA koji je u slici. Na morfologiji se bazira i analiza oblika i analiza pokreta u slici.

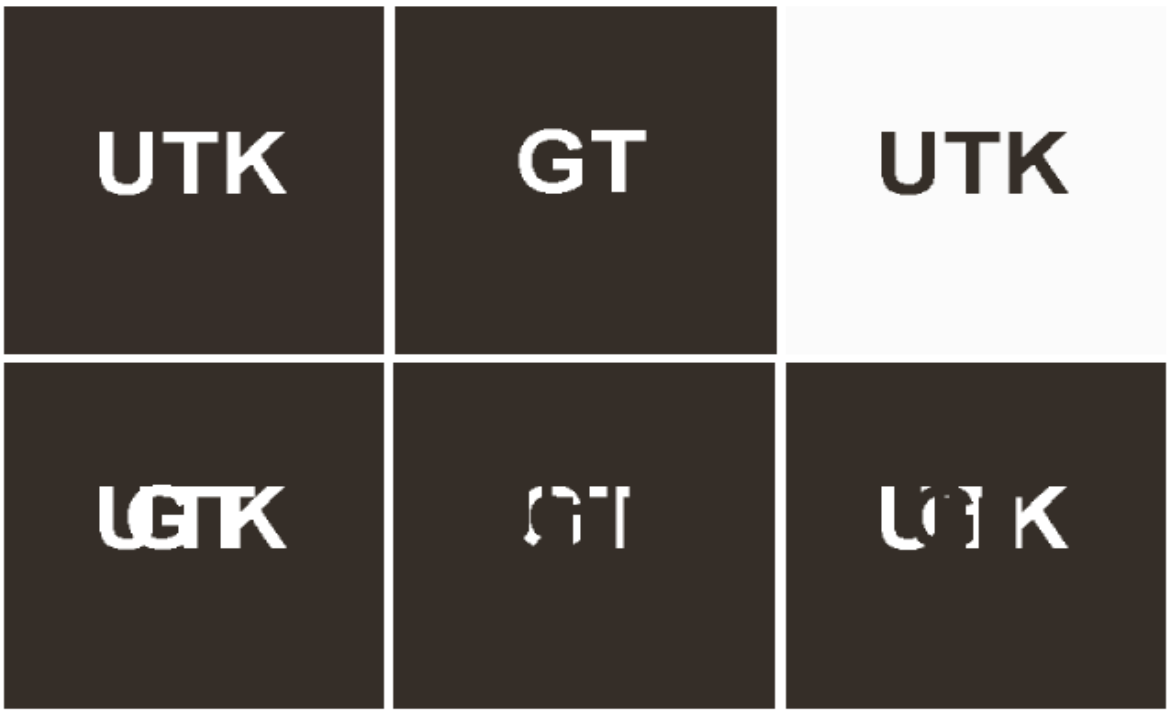


a b c
d e

FIGURE 10.1
(a) Two sets A and B. (b) The union of A and B. (c) The intersection of A and B. (d) The complement of A. (e) The difference between A and B.

Set Operation	MATLAB Expression for Binary Images	Name
$A \cap B$	A & B	AND
$A \cup B$	A | B	OR
A^c	~A	NOT
$A - B$	A & ~B	DIFFERENCE

TABLE 10.1
Using logical expressions in MATLAB to perform set operations on binary images.



a b c
d e f

FIGURE 10.3 (a) Binary image A. (b) Binary image B. (c) Complement ~A. (d) Union A | -B. (e) Intersection A & B. (f) Set difference A & ~B.

Dilatacija i erozija

Dilatacija i erozija su fundamentalne operacije morfološke obrade slike.

Dilatacija

Dilatacija je operacija koja podebljava objekte u slici, odnosno pravi objekte u slici većima. Način i količina zadebljanja kontroliraju se oblikom koji se naziva strukturni element. Svojstva dilatacije su asocijativnost i komutativnost.

Primjer 1:

korištenje dilatacije u Matlabu i rezultat prikazan na slici:

D=imdilate(A,B)

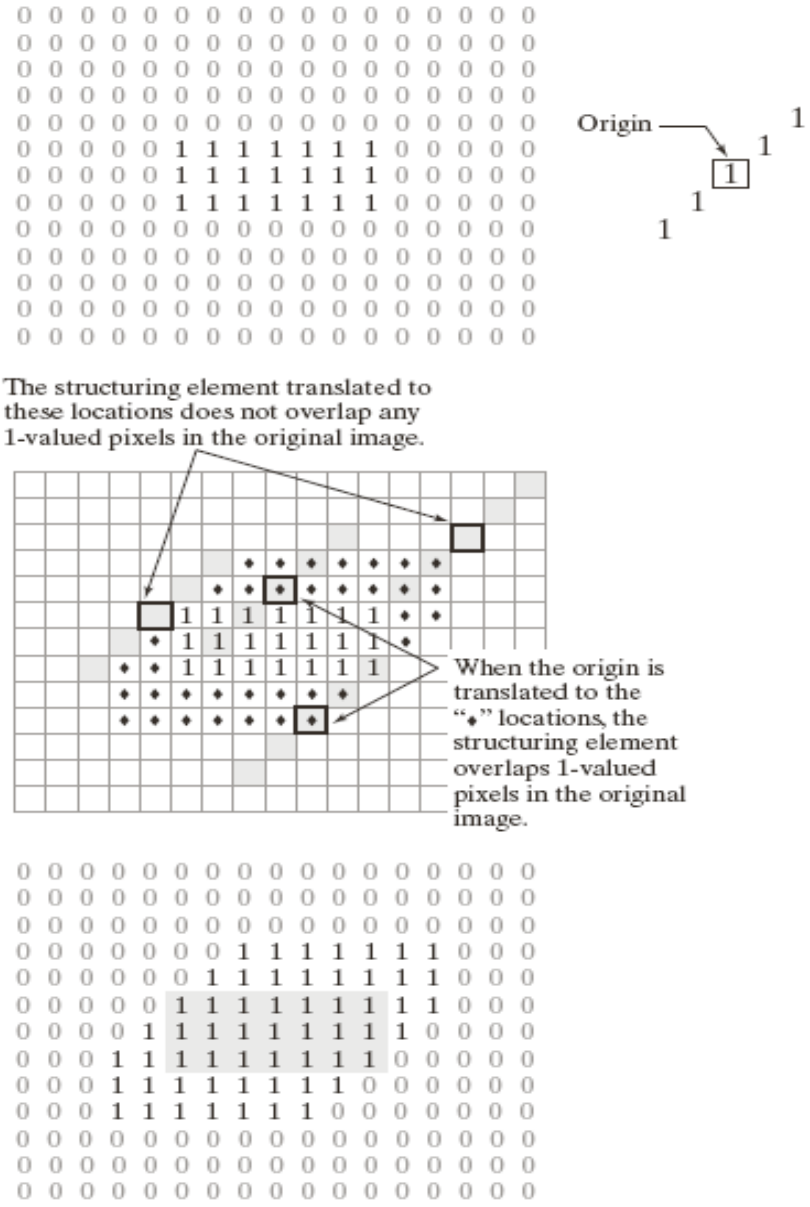
A=imread('broken_text.tif');

B=[0 1 0;1 1 1;0 1 0];

D=imdilate(A,B);

imshow(D);

figure, imshow(A)



Primjer 2:

D=imdilate(A,B)

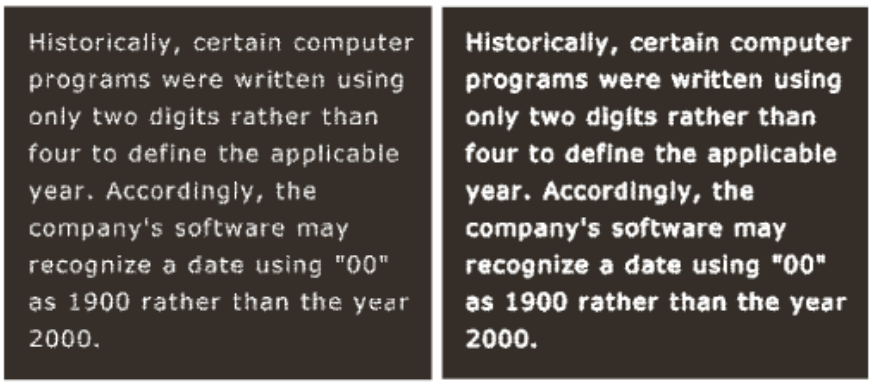
A=imread('broken_text.tif');

B=[0 1 0;1 1 1;0 1 0];

D=imdilate(A,B);

imshow(D);

figure, imshow(A)



a b
FIGURE 10.6
An example of dilation.
(a) Input image containing broken text.
(b) Dilated image.

Erozija

Erozija smanjuje tj. pravi objekte tanjima u binarnoj slici. Kao i kod dilatacije i kod erozije način i količina smanjenja objekta kontrolira se strukturnim elementom. Toolbox funkcija u Matlabu je *imerode().*

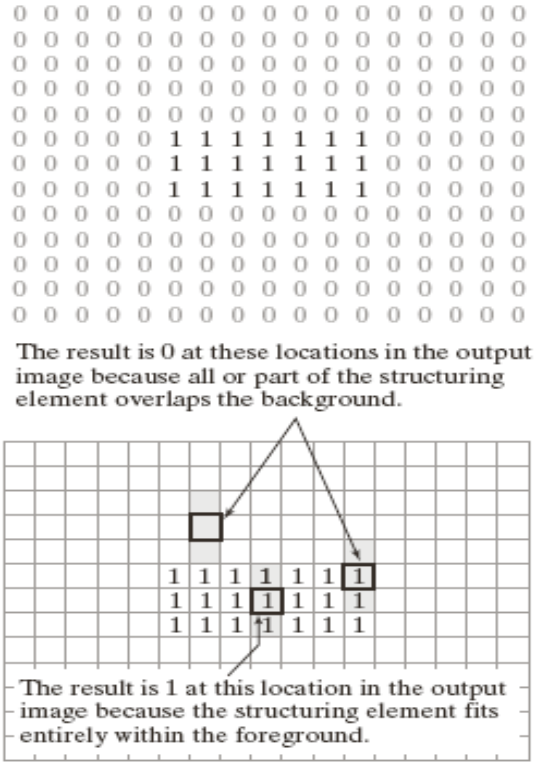
Primjer 1:

A=imread('wirebond_mask.tif');

se = strel('disk',10);

E10 = imerode(A,se);

imshow(E10)



a b
c d

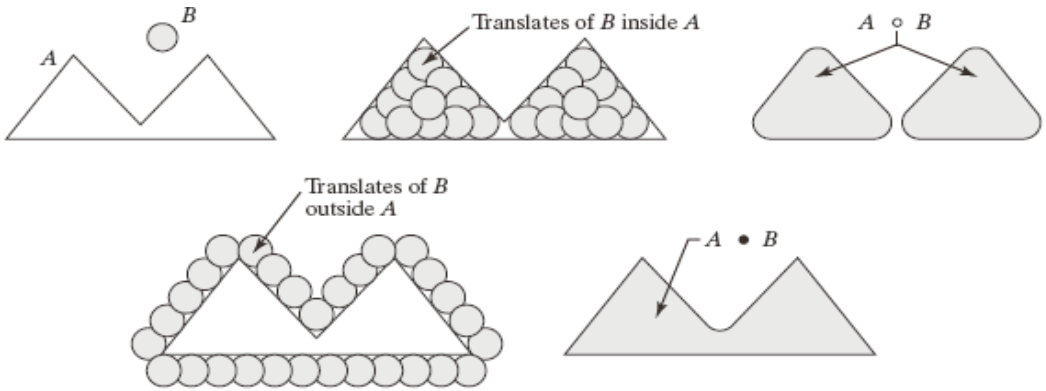
FIGURE 10.8
An illustration of erosion. (a) Original image of size 486×486 pixels. (b) Erosion with a disk of radius 10. (c) Erosion with a disk of radius 5. (d) Erosion with a disk of radius 20.

Otvaranje i zatvaranje slike

C = imopen(A,B);

C = imclose(A,B);

Otvaranje i zatvaranje slike se koristi za smanjnje šuma u slici.



a b c
d e

FIGURE 10.9 Opening and closing as unions of translated structuring elements. (a) Set A and structuring element B. (b) Translations of B that fit entirely within set A. (c) The complete opening (shaded). (d) Translations of B outside the border of A. (e) The complete closing (shaded).

a b c

FIGURE 10.11 (a) Noisy fingerprint image. (b) Opening of image. (c) Opening followed by closing. (Original image courtesy of the U. S. National Institute of Standards and Technology.)

Funkcija ***bwmorph()***

Ova funkcija interpretira različite morfološke operacije bazirane na kombinaciji dilatacije, erozije i LUT(look up table) operacija. Njena sintaksa je:

g = bwmorph(f, operation, n)

f – ulazna slika

n – broj iteracija

Najvažnije operacije koje može obaviti su:

- *Thinning* – reduciranje binarnih objekta na razinu debljine jednog piksela
- *Skeletonizing* – drugi način reduciranja binarnog objekta na razinu debljine jednog piksela

TABLE 10.3 Operations supported by function `bwmorph`.

Operation	Description
`bothat`	"Bottom-hat" operation using a 3×3 structuring element; use `imbothat` (see Section 10.6.2) for other structuring elements.
`bridge`	Connect pixels separated by single-pixel gaps.
`clean`	Remove isolated foreground pixels.
`close`	Closing using a 3×3 structuring element of 1s; use `imclose` for other structuring elements.
`diag`	Fill in around diagonally-connected foreground pixels.
`dilate`	Dilation using a 3×3 structuring element of 1s; use `imdilate` for other structuring elements.
`erode`	Erosion using a 3×3 structuring element of 1s; use `imerode` for other structuring elements.
`fill`	Fill in single-pixel "holes" (background pixels surrounded by foreground pixels); use `imfill` (see Section 11.1.2) to fill in larger holes.
`hbreak`	Remove H-connected foreground pixels.
`majority`	Make pixel p a foreground pixel if at least five pixels in $N_8(p)$ (see Section 10.4) are foreground pixels; otherwise make p a background pixel.
`open`	Opening using a 3×3 structuring element of 1s; use function `imopen` for other structuring elements.
`remove`	Remove "interior" pixels (foreground pixels that have no background neighbors).
`shrink`	Shrink objects with no holes to points; shrink objects with holes to rings.
`skel`	Skeletonize an image.
`spur`	Remove spur pixels.
`thicken`	Thicken objects without joining disconnected 1s.
`thin`	Thin objects without holes to minimally-connected strokes; thin objects with holes to rings.
`tophat`	"Top-hat" operation using a 3×3 structuring element of 1s; use `imtophat` (see Section 10.6.2) for other structuring elements.

Morfološka rekonstrukcija

Rekonstrukcija je morfološka transformacija koja upotrebljava dvije slike i strukturni element, umjesto samo jedne slike i strukturnog elementa. Druga slika je maska koja služi za ograničavanje transformacije. Strukturni element definira povezanost prve i druge slike. Ako je G maska, a F marker, rekonstrukcija G iz F, označena kao

$$R_G(F)$$

definira se slijedećom iterativnom procedurom:

- Inicijalizacija h1 da bude marker slika F

– Kreiramo strukturni element B = ones(3)

– Ponavljamo:

$$h_{k+1} = (h_k \oplus B) \cap C$$

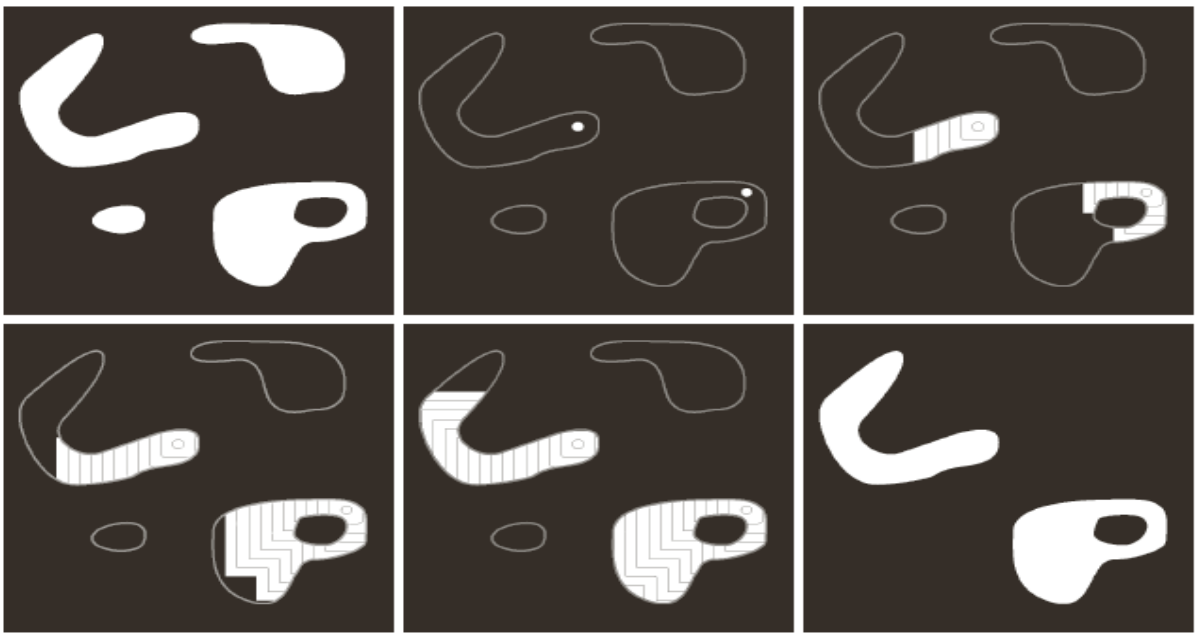
sve dok

$$h_{k+1} = h_k$$

– $R_G(F) = h_{k+1}$

– Marker F mora biti podskup od G:

Pogledajmo rezultat ove iterativne procedure:...



a b c
d e f

FIGURE 10.21 Morphological reconstruction. (a) Original image (the mask). (b) Marker image. (c)–(e) Intermediate result after 100, 200, and 300 iterations, respectively. (f) Final result. (The outlines of the objects in the mask image are superimposed on (b)–(e) as visual references.)

Morfologija sivih slika

Dilatacija sive slike svodi se na operator lokalnog maksimuma, gdje se maksimum pronalazi iz susjednih piksela određenih strukturnim elementom. Erozija sive slike svodi se na operator lokalnog minimuma, gdje se minimum pronalazi iz susjednih piksela određenih strukturnim elementom. Kombinacija dilatacije i erozije kod sivih slika koristi se npr. za dobivanje morfološkog gradijenta i to oduzimanjem erodirane slike od dilatirane slike. Morfološki gradijent je mjera za varijaciju lokalnih sivih nivoa u slici.

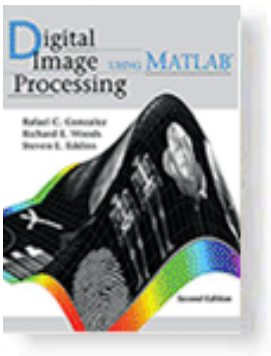
a b
c d

FIGURE 10.23 Dilation and erosion. (a) Original image. (b) Dilated image. (c) Eroded image. (d) Morphological gradient. (Original image courtesy of NASA.)

Laboratorijske vježbe iz kolegija
DIGITALNA OBRADA I ANALIZA SLIKA
(2016 – 2017)

dr.sc. Barbara Džaja

Napomena: Za rješavanje laboratorijskih vježbi nužan je paket materijala za podršku u fakultetskoj nastavi DIPUM 2 / e, od kojih su najvažniji P-kodovi DIPUM Toolbox 2, te originalne slike za obradu korištene u knjizi:



Digital Image Processing Using MATLAB 2nd Ed.

Gonzalez, Woods, and Eddins

© 2009

Laboratorijska vježba 1:

UPOZNAVANJE S MATLABOM

Učitavanje, prikaz i snimanje slike

Slike u Matlabu prezentirane su kao matrice. Sive slike kao dvodimenzionalna matrica, dok su slike u boji 3D ili više dimenzionalne matrice. Stoga, za obradu slika, u Matlabu se koriste sve funkcije koje služe za rad s matricama. Također, postoje i funkcije koje su namjenjene isključivo radu sa slikama, te su dio *Image Processing Toolboxa*. Popis tih funkcija dobije se korištenjem naredbe ***help images***.

```
>> help images
  Image Processing Toolbox
  Version 8.0 (R2012a) 29-Dec-2011
...
```

Da bi se mogle vršiti operacije nad slikama, potrebno je sliku najprije učitati. Matlab podržava sve važnije formate zapisa, a slika se učitava naredbom imread.

```
>> [img, map] = imread('canoe.tif');
>> whos
  Name       Size            Bytes  Class     Attributes
  img      207x346           71622  uint8
  map      256x3              6144  double
```

Slika je predstavljena 2D matricom *img* u kojoj su spremljene vrijednosti indexa elemenata u paleti *map* koja ima 256 boja. Učitanu sliku možemo pogledati kao i svaku ostalu varijablu unutar glavnog prozora Matlaba zadavanjem imena. Time samo ispisujemo numeričke vrijednosti elemenata matrice *img*, što nije pravi vizualni prikaz slike. Za vizualni prikaz slike koristi se naredba ***image***.

```
>> image(img)
>> colormap(map)
>> colorbar
```

Matlab kod pozivanja slike pretpostavlja da želimo prikaz uz standardnu paletu boja, stoga je skoro uvijek potrebno promijeniti paletu boja upotrebom naredbe ***colormap***. Odabranu paletu možemo pogledati na prikazanoj slici upotrebom naredbe ***colorbar***.

Ovo se odnosi na slike prikazane 2D matricama. Slike u boji se pohranjuju na način da svaku točku slike reprezentiramo sa tri vrijednosti inteziteta koji odgovaraju crvenoj (700 nm), zelenoj (546.1) i plavoj (435.8 nm) boji. Takva slika u Matlabu reprezentirana je 3D matricom, a kod njenog prikaza nije potrebno odabirati paletu.

```
>> [img, map] = imread('autumn.tif');
>> whos
  Name       Size              Bytes  Class     Attributes
  img      206x345x3           213210  uint8
  map        0x0                    0  double
>> image(img)
```

Zadatak:

Prikažite dobivenu sliku.

Slika u boji može se rastaviti na pojedine ravnine:

```
>> tocka = img(50,50,:);        % točka s koordinatama (50,50,:)
>> R = img(:,:,1);              % crvena ravnina
>> G = img(:,:,2);              % zelena ravnina
>> B = img(:,:,3);              % plava ravnina
```

Ako bi željeli istodobno prikazati sve tri komponente (R, G, B), potrebno je za svaku otvoriti novi prozor jer Matlab inače otvara uvijek u zadnjem otvorenom prozoru. Novi grafički prozor otvara se naredbom ***figure***, koja bez argumenata otvara novi prozor.

```
>> figure, image(R)
>> figure, image(G)
>> figure, image(B)
```

Zadamo li naredbi ***figure*** kao argument cijeli broj naredba otvara novi prozor s tim brojem, a ako takav prozor već postoji, onda on postaje aktivan. Naredba ***gcf*** daje broj aktivnog prozora.
Svaka 2D matrica u Matlabu može se pohraniti kao slika. Međutim, treba paziti na ograničenja standardnih formata računalnog zapisa slike. Učitana slika formata je ***uint8*** (cijeli broj zapisan s 8 bitova), dok je u Matlabu tipična matrica uglavnom ***double*** tipa. U tom slučaju, matricu koju želimo pohraniti moramo prvo prebaciti u ***uint8*** tip, skaliranjem i zaokruživanjem, a zatim koristimo naredbu ***imwrite***:

```
>>imwrite(B,'plava.jpg');
```

Zadatak:

Koristeći generator slučajnih brojeva kreirajte prvu dvodimenzionalnu matricu dimenzija 256×256, a koristeći operator dvotočku drugu matricu istih dimenzija u kojoj se vrijednosti linearno mijenjaju od –100 do 300. Prikažite ih korištenjem funkcija ***image*** i ***imagesc***, te ih snimite korištenjem funkcije ***imwrite***, sa i bez konverzije u ***uint8*** tip. Snimljene slike pregledajte u nekom pregledniku slika. Koja je razlika između funkcija ***image*** i ***imagesc*** ? Komentirajte razlike.

Konvertiranje jednog formata u drugi i računanje sa slikama

Tablica 1.1. Konverzija između pojedinih formata. (Potrebno je imati *Image Processing Toolbox*.) Unutar zagrada piše se ime slike koju se želi konvertirati.

Konverzija iz – u :	Matlab naredba:
Intensity/indexed/RGB format – binary format	dither()
Intensity format – indexed format	gray2ind()
Indexed format to intensity format	ind2gray()
Indexed format – RGB format	ind2rgb()
Matricu u intensity format putem skaliranja	mat2gray()
RGB format – intensity format	rgb2gray()
RGB format – indexed format	rgb2ind()

Naredba ***mat2gray*** je korisna ako imamo matricu koja predstavlja sliku u suštini ali su njene vrijednosti između npr. 0 i 1000. Tada naredba ***mat2gray*** automatski skalira sve vrijednosti elemenata matrice između 0 i 255 (kod klase ***uint8***) ili 0 i 1 (kod ***double*** klase).

Tablica 1.2. Podatak, tip i opseg klase u Matlabu

podatak	tip	opseg
int8	8-bit integer	-128 - 127
uint8	8-bit unsigned integer	0 - 255
int16	16-bit integer	-32768 - 32767
uint16	16-bit unsigned integer	0 - 65535
double	Double precision real number	Machine specific

Kada se slika u Matlabu procesuira, da bi se mogle vršiti nad njom operacije, potrebno ju je konvertirati iz ***uint8*** u ***double*** tip podatka:

```
>> I = im2double(img);
>> J = double(img);
>> Z = I - J;                    % tražimo razliku između vrijednosti matrica u I i J
>> nnz(Z)                        % funkcija nnz vraća broj elemata u Z različitih od nule
```

Razlika između ***im2double*** i ***double*** funkcije leži u tome što unkcija ***im2double*** pri konverziji slike ***uint8*** formata u rangu od 0-255 konvertira u double array ranga 0-1, dok korištenjem ***double*** funkcije treba podijeliti sa 255.

```
>> I=im2double(img);
>> J = double(img)/255;
>> Z = I - J;
>> nnz(Z)
```

Zadatak:

Koje još funkcije za konverziju postoje i koja je razlika između njih? (***double, im2single, im2int16, im2uint8, im2uint16***)

Funkcije max i min

Zadatak:

Kolike su maximalne i minimalne vrijednosti slika img, R, G, B?

Generator slučajnih brojeva

```
>> noise = randn(size(img));
>> imshow(noise)
```

Zadatak:

Koji još generatori postoje i koja je razlika među njima?

Informacije o slici daje funkcija ***imfinfo***:

```
>> imfinfo('cameraman.tif')
```

Laboratorijska vježba 2:

UNARNE OPERACIJE NA SLICI

Učitavanje, prikaz i snimanje slike

Unarne operacije na slici su matematičke operacije kod kojih je ulaz vrijednost točke, a izlaz promijenjena vrijednost točke, te se izvršava za sve točke u slici.
Ukoliko želimo da prikaz slike nakon unarne operacije bude smislen, neophodno je sliku skalirati. Slike tipa double treba skalirati na interval [0,1], dok one tipa int, treba skalirati na [0,255]. Naredba imagesc() ispisuje sliku koja je automatski skalirana na potrebni interval.

```
>> [img,map] = imread('moja_slika.ekstenzija');
>> img = double(img);    % većina funkcija traži double tip
>> imgU=sqrt(img);       % izvršimo unarnu operaciju
>> max(imgU(:))
ans =
15.9687                  % najveći element nije više 255

>> imagesc(imgU)         % odmah skaliramo i prikazujemo sliku
>> colormap(gray)        % ne zaboravimo promijeniti paletu
```

Zadatak:
3. Učitanu sliku pretvorite u crno-bijelu sliku. Izvršite na slici nekoliko unarnih operacija (logaritmiranje, korjenovanje i kvadriranje) te usporedite što se dogodilo s tamnijim dijelovima slike, a što sa svjetlijim.

BINARNE OPERACIJE NA SLICI

Kod binarnih operacija na ulazu imamo par slika nad kojima vršimo neku matematičku operaciju.

Kod binarnih operacija potrebno je prilagoditi veličine slika tako da bude moguće izvršiti operaciju. Pri tome imamo mogućnost izbora broja točaka, možemo npr. smanjiti veću sliku ili povećati manju, ili pak mijenjati obje. Potrebno je također pripaziti i da prostor boja koje obje slike koriste bude jednak (npr. RGB, HSV...), odnosno da obje slike budu indeksirane i imaju istu paletu boja ili da budu crno-bijele.

Primjer

```
>> [img1, map] = imread('moon.tif');
>> [img2, map] = imread('saturn.tif');
>> whos
```

Što je izlaz naredbe whos?

```
>> img2=imresize(img2,[537 358],'bilinear');
>> size(img2)     % sada je i prva slika 537x358
ans =
537358

>> img=double(img1)+double(img2);  % zbrajamo slike
>> imagesc(img); colormap(gray)  % prikazujemo rezultat
```

Zadatak:
1. Odaberite dvije crno-bijele slike različitih rezolucija te isprobajte nekoliko binarnih operacija (npr. zbrajanje, množenje i oduzimanje). Prikažite i objasnite rezultate.
2. Pokušajte isto napraviti za RGB-slike. Prikažite rezultate. Objasnite što se dogodilo.

Najčešća binarna operacija koja se koristi je oduzimanje slika s ciljem isticanja razlike. U toj primjeni obično je na jednoj slici uobičajena scena (pozadina), dok se na drugoj slici nalazi neki objekt ili pak više objekata koje želimo istaknuti. Uz pretpostavku da je pozadina nepromijenjena oduzimanjem dobivamo novu sliku na kojoj su najveće vrijednosti upravo na mjestima gdje se nalaze objekti koje želimo istaknuti.

Zadatak:
1. Učitajte slike angio0.tif i angio1.tif. Na tim slikama je snimka glave prije i nakon ubrizgavanja kontrastnog sredstva. Oduzmite te dvije slike te prikažite rezultat. Što ste dobili?

GAMA KOREKCIJA

Monitori s katodnom cijevi ulazne napone koji predstavljaju intenzitet ne preslikavaju linearno na zaslon, već je dobiveni intenzitet svjetla kojeg vidimo proporcionalan s potencijom ulaznog napona. Tipična vrijednost potencije je oko 2, što znači da sliku koju želimo prikazati na katodnoj cijevi monitora moramo prilagoditi za prikaz. To činimo tako da intenzitete ne preslikavamo linearno u napon, nego da vrijednosti najprije potenciramo s otprilike 1/2. Taj postupak se zove gama korekcija i uvijek se vrši prije prikaza na monitoru. Kako različiti monitori nemaju jednake karakteristike, za odredivanje točnih vrijednosti gama faktora potrebna je kalibracija. Gama korekcija dakle odgovara matematičkoj funkciji potenciranja, za koju smo potenciju označili slovom ϒ (gama). Za računanje gama korekcije u MATLAB-u se koristi funkcija *imadjust()*:

Primjer
```
>> [img,map]=imread('split.png');      % učitamo sliku
>> imshow(img)                         % prikažimo je
>> imgG=imadjust(img,[],[],0.5);       % računamo gamma korekciju
>> figure; imshow(imgG)                % usporedimo slike
```

Ukoliko je slika u RGB prostoru boja, sva tri kanala možemo mijenjati istovremeno naredbom imadjust() gdje za posljednji argument unosimo vrijednost gama korekcije. Ukoliko želimo mijenjati gama korekciju svakog kanala zasebno za poslijednji argument unosit ćemo vektor redak s tri komponente, od kojih je svaka gama korekcija pojedinog kanala.

Zadatak:
1. Odaberite nekoliko slika i isprobajte različite vrijednosti faktora gamma.
2. Učitajte sliku Fig0306(a).tif koja prikazuje scan cijelog kostura. Negativ slike dobijte naredbom:

```
>> g1 = imadjust(slika,[0 1], [1 0]);
```

Ovaj process ekvivalentan je dobivanju digitalnog negative slike, te je uvelike upotrebljiv za poboljšanje bijelih ili sivih detalja ugrađenih u dominantne tamne regije.
Negativ slike također se može dobiti upotrebom funkcije *imcomplement()*:

```
>> g2 = imcomplement(slika);
>> figure,imshow(g2)
```
Što radi naredba:

>> g3 = imadjust(slika,[0.5 0.75],[0 1]); ?

Ovakav način obrade koristan je za osvjetljavanje područja od interesa u slici. Konačno, upotrebom naredbe:

>> g4 = imadjust(slika,[],[],2);

Dobivamo više sivih tonova.

Zadatak:

Kao u gornjem primjeru iz slike Fig0303(a).tif vidljivo izdvojite sumljivo područje. Prikažite rezultate.

Laboratorijska vježba 3:

GEOMETRIJSKE TRANSFORMACIJE I REGISTRACIJE SLIKA

Geometrijske transformacije

Geometrijske transformacije mijenjaju prostorne veze između piskela u slikama. Slika može biti uvećana ili umanjena, rotirana, pomaknuta ili rastegnuta na različite načine. Geometrijske transformacije koriste se za stvaranje slika iz različitih kuteva, adaptiranje digitalnog videa iz jedne rezolucije u drugu, ispravljanje distorzija uzrokovanih geometrijom prostor-slika ili pak za mozaiciranje više slika u jednu panoramsku sliku.

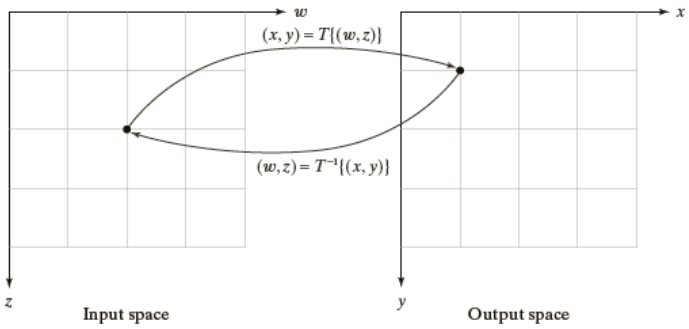
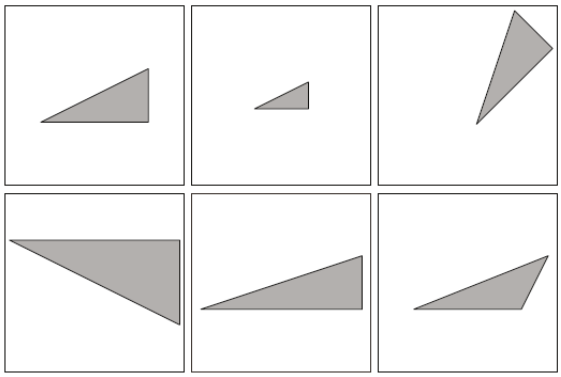


FIGURE 6.1 Forward and inverse transformation of a point for $T\{(w, z)\} = (w/2, z/2)$.

Slika 1. Transformacija točke (*Courtesy of Gonzalez, Woods, Eddins*)



a b c
d e f
FIGURE 6.4 Examples of similarity transformations. (a) Original object. (b) Scaled. (c) Rotated and translated. (d) Reflected and scaled. (e) Scaled horizontally but not vertically—not a similarity. (f) Horizontal shearing—not a similarity.

Slika 2. Primjeri transformacija(*Courtesy of Gonzalez, Woods, Eddins*)

Image Processing Toolbox u Matlabu koristi funkciju imtransform() za primjenjivanje geometrijske transformacije na sliku. Sintaksa je:

g = imtransform(f, tform)

Napravimo testnu sliku u obliku šahovskog polja koristeći naredbu *checkerboard()*:

```
>> f = checkerboard(50);
>> imshow(f)
```

Generirajmo strukturu za skaliranje (*tform*) i primjenimo je na testnu *checkerboard* sliku:

```
>> sx = 0.75;
>> sy = 1.25;
```

```
>> T = [sx 0 0; 0 sy 0;0 0 1]

T =

   0.7500         0         0
        0    1.2500         0
        0         0    1.0000

>> t1 = maketform('affine', T);
>> g1 = imtransform(f,t1);
```

Prikažimo skalirani rezultat rezultat:

```
>> imshow(g1)
```

Rotacija uz pomoć affine matrice uz kut *theta*:

```
>> theta = pi/6;
>> T2 = [cos(theta) sin(theta) 0;-sin(theta) cos(theta) 0; 0 0 1];
>> t2 = maketform('affine', T2);
>> g2 = imtransform(f, t2);
```

Prikažimo rotiranu sliku:

```
>> figure,imshow(g2)
```

Crni pikseli izlazne slike odgovaraju lokacijama izvan ulazne slike. U Matlabu se te crne regije mogu interpretirati uz pomoć bilo koje druge boje, ali se radi jednostavnosti uglavnom bira crna boja, odnosno funkcija *imtransform()* uvijek te piksele postavlja na vrijednost nula (crno).

Primjer projekcijske transformacije:

```
>> T3 = [0.4788 0.0135 -0.0009;0.0135 0.4788 -0.0009;0.5059 0.5059 1.0000];
>> tform3 = maketform('projective',T3);
>> g3 = imtransform(f,tform3);
```

Prikažimo rezultat:

```
>> figure,imshow(g3)
```

Zadatak:
Gornje primjere geometrijskih transformacija primjenite na jednu sliku u boji, po izboru. Prikažite kod uz rezultate.

Interpolacija slika

Interpolacija je proces konstruiranja kontinuirane funkcije iz diskretnih (uzorkovanih) podataka. Metode interpolacije razlikuju se po brzini i kvaliteti izlazne slike, a njihova usporedba radi se uz pomoć testa ponavljajuće rotacije. Slika se uzastupno rotira za 30^o dvanaest puta oko centralne točke uz pomoć funkcije *reprotate()*.

Primjer:

Učitajmo testnu sliku:

```
>> f = imread('cameraman.tif');
```

Funkcija *timeit()* mjeri vrijeme izvršavanja neke druge funkije, u našem slučaju *reprotate()* funkcije.

```
>> timeit(@() reprotate(f, 'nearest'))

ans =

   0.3959

>> timeit(@() reprotate(f, 'bilinear'))

ans =

   0.5845

>> timeit(@() reprotate(f, 'bicubic'))

ans =

   0.9344
```

Interpolacija metodom najbližeg susjeda je najbrža, dok je bikubična interpolacija najsporija.

Sada testirajmo izlaznu kvalitetu slike:

```
>> figure, imshow(reprotate(f, 'nearest'))
>> figure, imshow(reprotate(f, 'bilinear'))
>> figure, imshow(reprotate(f, 'bicubic'))
```

Interpolacija metodom najbližeg susjeda uništava glatkoću rubova, bilinearna interpolacija rezultira zamućenjem, dok bikubična interpolacija daje najbolji rezultat od tri testirane metode interpolacije.

Zadatak:
Gornje primjere testiranja interpolacije primjenite na jednu sliku u boji, po izboru. Prikažite kod uz rezultate.

Registracije slika bazirane na području

Kod registracija baziranih na području jedna slika – *template* pomiče se tako da pokriva neku drugu lokaciju u drugoj slici. Na toj drugoj lokaciji računaju se mjere sličnosti između preklopljenih područja. Za *template* sliku se kaže da je preklopljena na određenom području u drugoj slici ako se na istome mjestu nalazi i jedan i drugi maksimum razlike. Mjera sličnosti upotrebljavana kod registracije bazirane na području u slici je *normalizirana kros-korelacija* (*koeficijent korelacije*), gdje se izračun koeficijenta korelacije između slike i *templatea* računa na slijedeći način:

$$\gamma(x,y) = \frac{\sum_{s,t}\left[\omega(s,t) - \overline{\omega}\right]\left[f(x+s,y+t) - \overline{f}_{xy}\right]}{\sqrt{\sum_{s,t}\left[\omega(s,t) - \overline{\omega}\right]^2 \sum_{s,t}\left[f(x+s,y+t) - \overline{f}_{xy}\right]^2}}$$

ω je *template*, $\overline{\omega}$ je prosječna vrijednost elemenata u *templateu*, *f* je slika, $\overline{f}_{xy}$ je prosječna vrijednost slike na području gdje se *f* i ω preklapaju. Suma je uzeta preko vrijednosti *s* i *t*, na način da se slika i *template* preklapaju. Nazivnik u gornjoj relaciji normalizira metriku obzirom na varijacije u intezitetu. Vrijednost $\gamma(x, y)$ je u rangu [-1 1]. Velika vrijednost za $|\gamma(x, y)|$ znaći i dobro preklapanje između *templatea* i slike, kada je *template* centriran na koordinatama (x, y).
Funkcija sadržana u *Image Processing Toolboxu* za izračun koeficijenta korelacije je *normxcorr2()*, a njena sintaksa je:

g = normxcorr2(template, f)

Primjer:
Upotreba funkcija *normxcorr2()* i visreg() za pronalazak najboljeg preklapanja između slike i *templatea* kao i za registraciju slika.

Učitajmo slike Fig0621(a).tif i Fig0621(b).tif u workspace:

```
>> f1 = imread('Fig0621(a).tif');
>> f2 = imread('Fig0621(b).tif');
```

Template sliku izrežimo iz direktno iz jedne od slika, u proizvoljnoj veličini:

```
>> w = f2(200:400, 300:600);
>> imshow(w)
```

Upotrijebimo *normxcorr2()* za lociranje *templatea* u obje slike:

```
>> g1 = normxcorr2(w,f1);
>> g2 = normxcorr2(w,f2);
```

Pronađimo lokacije maksimalnih vrijednosti od g1 i g2 i oduzmimo lokacije in tako pronađimo translaciju:

```
>> [y1, x1] = find(g1 == max(g1(:)));
>> [y2, x2] = find(g2 == max(g2(:)));
>> delta_x = x1 - x2

delta_x =

  -194

>> delta_y = y1 - y2

delta_y =

    -2
```

Kada smo pronašli relativnu translaciju između slika, pronađimo i afinu transformaciju između njih, te slike proslijedimo kroz *visreg()* za vizualizaciju preklopljenih slika:

```
>> tform = maketform('affine', [1 0 0; 0 1 0; ...
```

delta_x delta_y 1]);

>> visreg(f1, f2, tform)

Što dobivamo na prikazu?
Iako u lijevom dijelu izlaza slike izgledaju dobro preklopljene, desno se vidi da nisu baš dobro preklopljene. To upućuje da geometrijska veza između dviju slika nije samo čista translacija.

Zadatak:
Što je to mozaiciranje slike?
Što znate o RANSACu?

Laboratorijska vježba 4:

OBRADA SLIKE U BOJI

RGB slike

RGB slika u boji je MxNx3 matrica brojeva, gdje svaki piksel u boji predstavlja trojku koja odgovara crvenoj, zelenoj i plavoj komponenti RGB slike na točno definiranoj prostornoj lokaciji.

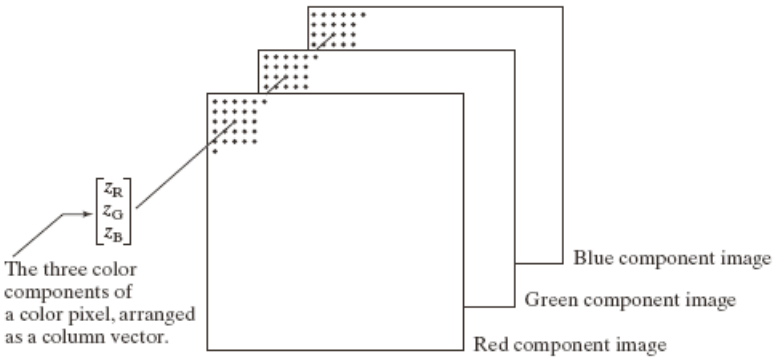


FIGURE 7.1 Schematic showing how pixels of an RGB color image are formed from the corresponding pixels of the three component images.

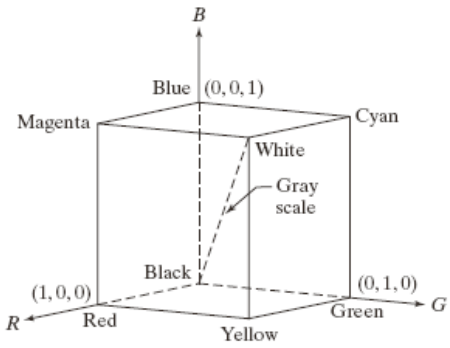
Slika 1. RGB piksel (*Courtesy of Gonzalez, Woods, Eddins*)

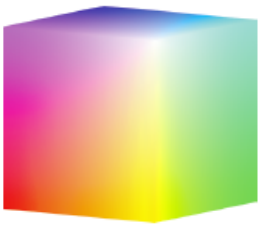
RGB prostor boja grafički se prikazuje kao kocka u boji, a može se vidjeti upotrebom funkcije *rgbcube(vx,vy,vz):*

>> rgbcube

a b

FIGURE 7.2 (a) Schematic of the RGB color cube showing the primary and secondary colors of light at the vertices. Points along the main diagonal have gray values from black at the origin to white at point (1, 1, 1). (b) The RGB color cube.





Slika 2. RGB kocka(*Courtesy of Gonzalez, Woods, Eddins*)

Funkcija *rgbcube(vx,vy,vz)* kao izlaz daje RGB kocku, gledano iz točke (vx,vy,vz).

Indeksirane slike

Indeksirane slike imaju dvije komponente: podatkovnu matricu cijelih brojeva *X* i matricu mape boja *map*.

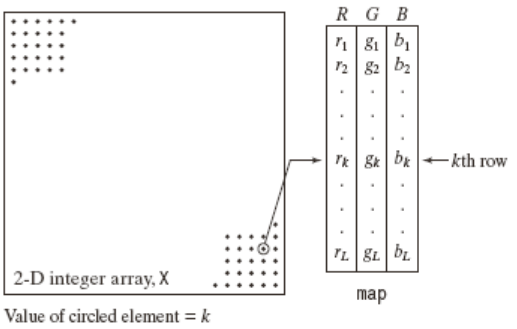


FIGURE 7.3
Elements of an indexed image. The value of an element of integer array X determines the row number in the color map. Each row contains an RGB triplet, and L is the total number of rows.

Slika 3. Elementi indeksirane slike(*Courtesy of Gonzalez, Woods, Eddins*)

Matrica map je mx3 niz klase double, koja sadrži decimalne vrijednosti u rangu [0,1]. Duljina mape odgovara broju boja koje definira.
Za prikaz indeksirane slike koriste se funkcije:
>>imshow(X,map)
ili
>>image(X)
>>colormap(map)

Primjer:
dither funkcija može se primjeniti kako na sive slike, tako i na slike u boji. Dithering je proces rutinski upotrebljavan kod printanja i u izdavačkoj industriji, a služi za vizualnu impresiju varijacija u sjeni na printanoj slici.
Učitajte sliku Fig0704(a).tif i prebacite je u indeksiranu sliku.

```
>>f = imread('Fig0704(a).tif');
>>imshow(f)
>>[X1, map1] = rgb2ind(f,8,'nodither');
>>figure,imshow(X1, map1)
>>[X2, map2] = rgb2ind(f,8,'dither');
>>figure,imshow(X2, map2)
```

Obje slike imaju samo 8 boja, što je značajna redukcija u 16-milijunskom mogućem izboru boja koji pruža uint8.

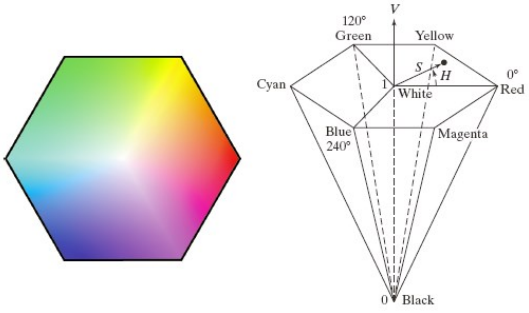
Efekti dithering funkcije bolje su ilustrirani kod slika u sivim tonovima:

Kakve slike dobivamo kao rezultat? Koja slika je ispala binarna?

Konverzija između prsotora boja

HSV - RGB:

a b
FIGURE 7.5
(a) The HSV color hexagon. (b) The HSV hexagonal cone.



Slika 4. HSV(*Courtesy of Gonzalez, Woods, Eddins*)

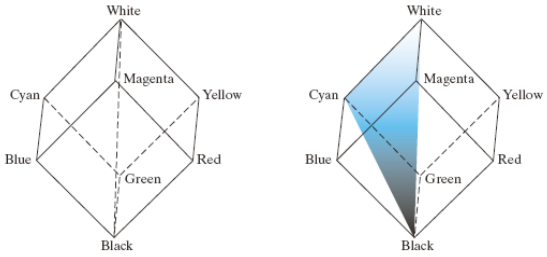
hsv_image = rgb2hsv(rgb_image)
rgb_image = hsv2rgb(hsv_image)

CMY (CMYK) – RGB:

cmy_image = imcomplement(rgb_image)
rgb_image = imcomplement(cmy_image)

HSI – RGB:

a b
FIGURE 7.6
Relationship between the RGB and HSI color models.



Slika 5. HSI(*Courtesy of Gonzalez, Woods, Eddins*)

hsi = rgb2hsi(rgb)
rgb = hsi2rgb(hsi)

Zaglađivanje slike

Rastavimo sliku na tri komponente:
fR = f(:,:,1);
fG = f(:,:,2);
fB = f(:,:,3);

Generirajmo filter:
w = randn(3);

Filtrirajmo svaku komponentu pojedinačno:

fR_filtered = imfilter(fR,w,'replicate');
fG_filtered = imfilter(fG,w,'replicate');
fB_filtered = imfilter(fB,w,'replicate');

rekonstruirajmo RGB sliku:

fc_reconstructed = cat(3,fR_filtered, fG_filtered, fB_filtered);
imshow(fc_reconstructed)

Ili kraće:

fc_filtered = imfilter(f,w,'replicate');

figure,imshow(fc_filtered)

Prikažite rezultat.

Napravimo zaglađivanje na HSI slici:

```
>> h = rgb2hsi(f);
```

Rastavimo je na komponente, H, S, I:

```
>> H = h(:,:,1);
>> S = h(:,:,2);
>> I = h(:,:,3);
```

Filtrirajmo komponentu inteziteta upotrebom istog filtera velićine 25x25 piksela. Filter uprosjećivanja trebao bi napraviti poprilično zamućenje.

```
>> w = fspecial('average',25);
>> I_filtered = imfilter(I,w,'replicate');
```

Spojimo sliku u jednu:

```
>> h = cat(3,H,S,I_filtered);
>> f = hsi2rgb(h);
>> imshow(f)
```

Koja je razlika između filtriranja u RGB i HSI prostoru boja?

Detekcija ruba u slici uz pomoć gradijenta

Učitajte sliku po izboru i nađite rub uz pomoć gradijenta, upotrebom funkcije:

```
>> [VG, A, PPG] = colorgrad(slika);
```

VG je RGB vector gradijent, A je slika kuta, a PPG je gradijenta slika formirana sumom 2D gradijentnih slika individualnih ravnina boja.
Prikažite ih.

Laboratorijska vježba 5:

2D KONVOLUCIJA I FILTRIRANJE SLIKE

Osnovna ideja konvolucije je da se jedna slika (maska) „šeta“ po drugoj slici, dok izlazne vrijednosti piksela predstavljaju težinsku sumu piksela ulazne slike, gdje su težine određene vrijednostima piksela prozorske slike, tj. filtra.
Maska nam predstavlja impulsni odziv sustava i obično je kvadratna matrica manjih dimenzija kao što su 3×3, 5×5 i 7×7. Razlog je u broju operacija potrebnih da se odredi 2D konvolucija signala i impulsnog odziva (dvostruka suma po stupcima i recima matrice). Konvolucija slike i maske računa se pozivom funkcije *conv2*. Matlab pri računanju 2D konvolucije pretpostavlja da su vrijednosti točaka izvan slike jednake nuli, tj. da je slika proširena nulama. Konvoluciju u MATLAB-u računamo na sljedeći način:

```
>> img = imread('moja_slika.ekstenzija');
>> mask = [1 0 -1; 2 0 -2; 1 0 -1]/4;
>> img = im2double(img);
>> imgConv(:,:,1) = conv2(img(:,:,1),mask);
>> imgConv(:,:,2) = conv2(img(:,:,2),mask);
>> imgConv(:,:,3) = conv2(img(:,:,3),mask);
>> subplot(1,2,1),imshow(img)
>> subplot(1,2,2),imshow(imgConv)
```

Ako je slika dimenzija $M_x \times N_x$ i maska dimenzija $M_h \times N_h$ konvolucija je dimenzija $M_x + M_h - 1 \times N_x + N_h - 1$. Obično je poželjna izlazna slika jednaka veličini ulazne, te se u tom slučaju uzima centralni dio konvolucije. Ovakav postupak unosi pogreške na rubovima slike:

```
>> img = rgb2gray(img);
>> Mask = [1 0 -1; 1 0 -1; 1 0 -1]/3;
>> imgC1 = conv2(img, Mask);              % računanje pune konvolucije
>> imgC2 = conv2(img, Mask, 'same');      % računanje centralnog dijela konvolucije
                                          % size(izlaz) = size(ulaz)
>> imgC3 = conv2(img, Mask, 'valid');     % računanje samo ispravnog centralnog dijela
                                          % konvolucije
>> subplot(1,3,1),imshow(imgC1)
>> subplot(1,3,2),imshow(imgC2)
>> subplot(1,3,3),imshow(imgC3)
```

Zadatak: Za jednu proizvoljnu sliku izračunajte linearne konvolucije sa zadanim maskama i komentirajte rezultate. Učitanim slikama prije konvolucije upotrebom funkcije *imresize()* smanjite rezoluciju na 15 % njihove ucitane velicine.

$$\frac{1}{4}\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix} \qquad \frac{1}{3}\begin{bmatrix}1 & 1 & 1\\ 0 & 0 & 0\\ -1 & -1 & -1\end{bmatrix} \qquad \begin{bmatrix}0 & -1 & 0\\ -1 & 4 & -1\\ 0 & -1 & 0\end{bmatrix} \qquad \begin{bmatrix}-1 & -1 & -1\\ -1 & 8 & -1\\ -1 & -1 & -1\end{bmatrix}$$

Za filtriranje slike u Matlabu koriste se gotove funkcije: *medfilt2*, *filter2*, *wiener2*:

```
>> I = imread('moja_slika.ekstenzija');
>> J = imnoise(I,'salt & pepper',0.02);
>> K(:,:,1) = medfilt2(J(:,:,1));
```

```
>> K(:,:,2) = medfilt2(J(:,:,2));
>> K(:,:,3) = medfilt2(J(:,:,3));
>> subplot(1,3,1),imshow(I)
>> subplot(1,3,2),imshow(J)
>> subplot(1,3,3),imshow(K)
```

Kakva je rezultirajuća slika? Objasnite što radi *medfilt2()* funkcija.

```
>> I = rgb2gray(I);
>> J = imnoise(I,'gaussian',0,0.005);
>> K = wiener2(J,[5 5]);
>> imshow(J)
>> figure, imshow(K)
```

Kakva je rezultirajuća slika? Objasnite što radi *wiener2*() funkcija, a što *filter2*() funkcija. Odaberite nekoliko maski veličine 3 x 3, za svoju sliku, te izračunajte i prikažite filtriranu sliku upotrebom *filter2()* funkcije. Komentirajte rezultate.

Laboratorijska vježba 6:

OBNAVLJANJE I REKONSTRUKCIJA SLIKA

Estimiranje parametara šuma

Učitajmo sliku **Fig0504(a).tif** koja sadrži šum i prikažimo je.

```
>> f = imread('Fig0504(a).tif');
>> imshow(f)
```

Cilj ovog primjera je estimirati tip šuma i njegove parametre. Generirajmo sliku **B** upotrebom funkcije *roipoly()*. Ta funkcija omogućava korisniku samostalno odabiranje mjesta interesa u slici (Region Of Interest - **ROI**), za koje korisnik smatra da nema objekata u pozadini te su jedini varijacije u intezitetima na tom području posljedica šuma u slici.

```
>> [B, c, r] = roipoly(f);
```

B je slika od interesa, dok su **c** i **r** vektori koordinata stupaca i redova. Pokažimo odabrani dio:

```
>> figure, imshow(B)
```

Pronađimo histogram

```
>> [h, npix] = histroi(f, c, r);
>> figure, bar(h,1)
```

Srednja vrijednost i varijanca regije ograničene maskom **B** dobivaju se na slijedeći način:

```
>> [v, unv] = statmoments(h, 2)
```

Iz slike histograma evidentno je da je šum približno Gausov.

Restauracija slike degradirane jedino šumom – Prostorno filtriranje

Učitajmo sliku bez šuma:

```
>> f = imread('Fig0505(a)[without_noise].tif');
```

Korumpirajmo sliku šumom tipa *'salt & pepper'* s vjerojatnošću 0.1:

```
>> [M, N] = size(f);
>> R = imnoise2('salt & pepper', M, N, 0.1, 0);
>> gp = f;
>> gp(R == 0) = 0;
>> imshow(gp)
```

Napravimo još jedan primjer korumpirane slike:

```
>> R = imnoise2('salt & pepper', M, N, 0, 0.1);
>> gs = f;
>> gs (R == 1) = 255;
>> figure,imshow(gs)
```

Dobar način za filtriranje šuma ***'papra'*** je upotreba kontraharmoničnog filtra. Pogledajmo rezultat:

```
>> fp = spfilt(gp, 'chmean', 3, 3, 1.5);
>> figure,imshow(fp)
```

Šum ***'soli'*** možemo odstraniti s negativnim vrijednostima:

```
>> fs = spfilt(gs, 'chmean', 3, 3, -1.5);
>> figure,imshow(fs)
```

Slični rezultati mogu se dobiti upotrebom **max** i **min** filtera:

```
>> fpmax = spfilt(gp, 'max', 3, 3);
>> figure,imshow(fpmax)

>> fsmin = spfilt(gs, 'min', 3, 3);
>> figure,imshow(fsmin)
```

Adaptivno prostorno filtriranje šuma

Gore primjenjeni filtri neovisni su o varijacijama u slici. U nekim slučajevima rezultati se mogu poboljšati adaptacijom ponašanja baziranog na karakteristikama slike u regiji slike u kojoj se filtriraju. Stoga ćemo pogledati primjer primjene adaptivnog median filtra.

Korumpirajmo sliku šumom *'salt & pepper'*:

```
>> g = imnoise(f, 'salt & pepper', .25);
>> imshow(g)
```

Pogledajmo rezultat djelovanja običnog median filtra:

```
>> f1 = medfilt2(g, [7 7], 'symmetric');
>> figure,imshow(f1)
```

Slika je razumno očišćena od šuma, ali je poprilično zamućena i iskrivljena. Stoga, usporedimo taj rezultat s rezultatom adaptivnog filtriranja:

```
>> f2 = adpmedian(g, 7);
>> figure,imshow(f2)
```

Rezultat je također razumno očišćen od šuma, ali i mnogo manje zamućen i mnogo manje iskrivljen.

Modeliranje funkcije degradacije

Generirajmo testnu sliku i prikažimo je:

```
>> f = checkerboard(8);
>> imshow(f)
```

Generirana slika je tipa *double*. Degradirajmo sliku na slijedeći način:

```
>> PSF = fspecial('motion', 7, 45);
```

```
>> gb = imfilter(f, PSF, 'circular');
>> figure,imshow(gb)
```

PSF je prostorni filter. Generirajmo Guasov šum sa srednjom vrijednošću 0 i varijancom 0.001:

```
>> noise = imnoise2('Gaussian', size(f,1), size(f,2), 0,...
sqrt(0.001));
```

Zatim, generirajmo zamućeno šumovitu sliku:

```
>> g = gb + noise;
>> figure,imshow(g)
```

Šum nije baš dobro vidljiv u ovoj slici jer je njegova maksimalna vrijednost približno 0.15, a maksimalna vrijednost slike je 1. Međutim, kad želimo restaurirati originalnu sliku i ova količina šuma u slici je velika.

Wiener filtar

Upotrijebimo istu degradiranu sliku **g** i njenu PSF. Napravimo direktno inverzno filtriranje i pogledajmo rezultat:

```
>> frest1 = deconvwnr(g, PSF);
>> imshow(frest1)
```

Rezultatom dominira efekt šuma. Primjenimo Wienerov filtar s konstantnim omjerom i prikažimo rezultat:

```
>> Sn = abs(fft2(noise)).^2;              % Spektar jačine šuma
>> nA = sum(Sn(:))/numel(noise);          % Prosječna snaga šuma
>> Sf = abs(fft2(f)).^2;                  % Spektar snage slike
>> fA = sum(Sf(:))/numel(f);              % Prosječna snaga slike
>> R = nA/fA;
>> frest2 = deconvwnr(g, PSF, R);
>> figure, imshow(frest2)
```

Ovakav pristup daje značajan napredak obzirom na direktno inverzno filtriranje. Međutim, upotrebom autokorelacijskoh funkcija mogu se dobiti još bolji rezultati:

```
>> NCorr = fftshift(real(ifft2(Sn)));
>> ICorr = fftshift(real(ifft2(Sf)));
>> frest3 = deconvwnr(g, PSF, NCorr, ICorr);
>> figure, imshow(frest3)
```

Rezultat je značajno bliži originalnoj slici, iako još uvijek u slici ima šuma. Obzirom da smo poznavali originalnu sliku i funkciju šuma mogli smo proračunati točne parametre potrebne za restauraciju.

Iterativna nelinearna restauracija slike upotrebom Lucy – Richardson algoritma

Generirajmo testnu sliku:

```
>> g = checkerboard(8);
```

```
>> imshow(g)
>> imshow(pixeldup(g,8))                    % povećava vidljivost slike za 8 puta
```

Generirajmo Gausov PSF veličine 7x7 sa standardnom devijacijom 10:

```
>> PSF = fspecial('gaussian', 7,10);
```

Zamutimo sliku **g** upotrebom PSF i dodajmo Gausov šum srednje vrijednosti nula i standardne deviacije 0.01:

```
>> SD = 0.01;
>> g = imnoise(imfilter(g,PSF), 'gaussian', 0, SD^2);
>> figure, imshow(g)
```

Upotrijebimo sada sintaksu potrebnu za izvođenje Lucy – Richardsonovog algoritma:

```
>> DAMPAR = 10*SD;
>> LIM = ceil(size(PSF, 1)/2);
>> WEIGHT = zeros(size(g));
>> WEIGHT(LIM + 1: end - LIM, LIM + 1: end - LIM) = 1;
>> NUMIT = 5;
>> f5 = deconvlucy (g, PSF, NUMIT, DAMPAR, WEIGHT);
>> figure, imshow(pixeldup(f5, 8), [])
```

DAMPAR je skalar koji specificira prag devijacije rezultirajuće slike. WEIGHT je niz iste veličine kao slika g, koji sadržava težine za svaki pojedini piksel u svrhu reflektiranja njegove kvalitete. Npr. Loš rezultirajući piksel može se na takav način isključiti iz rješenja ako mu se težina postavi na vrijednost nula.
Iako se rezultat popravio, slika je i dalje malo zamućena.

Zadatak:
Ponovite postupak Lucy – Richardson algoritma s brojem iteracija NUMIT = 10 , 20, 100. Koja je razlika u rezultatu?

Slijepa dekonvolucija

Kada PSF nije poznata, potrebno je estimirati što bolju procjenu PSFa. Metode restauracije slike koje se ne baziraju na prethodnom znanju PSFa nazivaju se algoritmi slijepe dekonvolucije.

Generirajmo testnu sliku i degradirajmo je:

```
>> g = checkerboard(8);
>> PSF = fspecial('gaussian', 7, 10);
>> imshow(pixeldup(PSF, 73), [])
>> SD = 0.01;
>> g = imnoise(imfilter(g, PSF), 'gaussian', 0, SD^2);
```

Upotrijebimo funkciju *deconvblind()* za dobivanje estimacije (procijene) funkcije PSF, sa danom samo degradiranom slikom **g**:

```
>> INITPSF = ones(size(PSF));
>> NUMIT = 5;
```

```
>> DAMPAR = 10*SD;
>> WEIGHT = zeros(size(g));
>> LIM = ceil(size(PSF, 1)/2);
>> WEIGHT(LIM + 1: end - LIM, LIM + 1: end - LIM) = 1;
>> [g5, PSF5] = deconvblind(g, INITPSF, NUMIT, DAMPAR, WEIGHT);
>> figure, imshow(pixeldup(PSF, 73), [])
```

Vrijednosti i značaj varijabli DAMPAR i WEIGHT isti je kao u ptrehodnom primjeru.

Zadatak:
Ponovite slijepu dekonvoluciju sa 10 i 20 iteracija (NUMIT). Kakav je rezultat?

Laboratorijska vježba 7:

KOMPRESIJA SLIKA

Učitajmo sliku **'Fig0205(a).tif'** i spremimo ju kao jpeg komprimiranu sliku sa degradacijom 25, uz pomoć funkcije:

imwrite(f, 'filename.jpg', 'quality', q);

gdje je **q** cijeli broj između 0 i 100. Niži **q** označava veću degradaciju u jpeg kompresiji.

```
>> x = imread('Fig0205(a).tif');
>> imwrite(x,'bubbles25.jpg','quality',25)
>> imfinfo bubbles25.jpg
```

Lažno oslikavanje u slici može se uočiti na **q** = 5. Stoga, prihvatljivo rješenje u pogledu greške je komprimiranje slike sa granicom q = 25. Uočimo sada razliku:

```
>> imwrite(x,'bubbles5.jpg','quality',5)
>> imfinfo bubbles5.jpg
```

Funkcija *imratio()* računa omjer kompresije:

```
>> r = imratio(imread('bubbles25.jpg'),'bubbles25.jpg')
>> r = imratio(imread('bubbles5.jpg'),'bubbles5.jpg')
```

Korijen greške srednje vrijednosti (rmse) računa razliku između dviju slika, a funkcija *rmse()* osim greške između dviju slika vraća i histogram razlike između te dvije slike:

```
>> y = imread('bubbles5.jpg');
>> rmse = compare(x,y,1)
```

Huffmanovo kodiranje

Huffmanovo kodiranje je kodiranje bez gubitaka, ne gube se nikakvi podaci.
Ako promatramo jednostavnu 16-bajtnu sliku veličine 4x4:

```
>> f2 = uint8([2 3 4 2; 3 2 4 4; 2 2 1 2; 1 1 2 2])
>> whos('f2')
```

Onda vidimo da je svaki piksel u **f2** 8-bitni bajt, a 16 bajtova treba za prikazati cijelu sliku. Kako nivoi sive u f2 nisu jednako vjerojatni, Huffmanovo kodiranje smanjit će količinu potrebe memorije za prezentiranje slike. Funkcija huffman() računa jedan takav kod:

```
>> c = huffman(hist(double(f2(:)), 4))
```

Kako se Huffmanov kod bazira na relativnoj frekvenciji pojavljivanja simbola koji se kodiraju, slika f2 može se dobiti kodirana slijedećom operacijom:

```
>> h1f2 = c(f2(:))'
```

Funkcija whos() može nam dati podatak koliko je bajtova potrebno za spremanje te slike:

```
>> whos('h1f2')
```

Vidimo da kodirana slika zauzima 1018 bajtova memorije, što je 60 puta više od potrebnog.

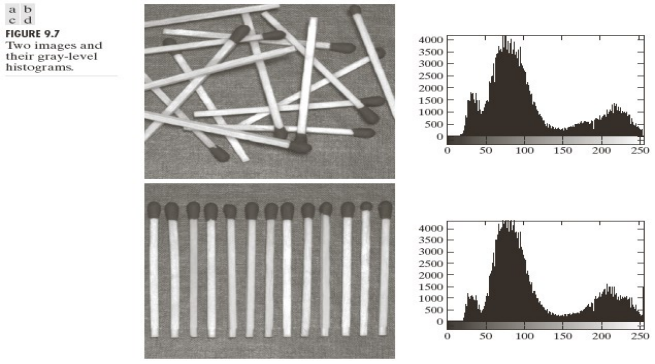
```
>> h2f2 = char(h1f2)'                % prebacujemo h1f2 u 2D niz karakternih znakova
>> whos('h2f2')
```

Da bi komprimirali f2, kod c mora biti primjenjen na bitovnoj razini. Pogledajmo to na slijedećem primjeru.

```
>> f = imread('Fig0904(a).tif');
>> c = mat2huff(f);
>> cr1 = imratio(f,c)
>> save SqueezeTracy c;
>> cr2 = imratio('Fig0904(a).tif', 'SqueezeTracy.mat')
```

Prostorna redundancija

Promotrimo slike:



Slike imaju virtulano identične histograme. Histogrami su trimodalni, što znaći da postoje tri dominantne regije vrijednosti sivih nivoa. Kako sivi nivoi slike nisu jednako vjerojatni, kodiranje na temelju dužine varijable (Huffmanovo kodiranje) može se upotrijebiti da bi se smanjila redundancija koja je posljedica binarnog kodiranja piksela.

Učitajmo slike, pogledajmo njihovu entropiju i omjer kompresije:

```
f1 = imread('Fig0907(a).tif');
c1 = mat2huff(f1);
ntrop(f1)
imratio(f1, c1)
f2 = imread('Fig0907(c).tif');
c2 = mat2huff(f2);
ntrop(f2)
imratio(f2,c2)
```

Koja slika ima veću kompresiju?

U svrhu reduciranja među pikselne redundancije, 2D pikselni niz koji upotrebljava ljudski vizualni sustav mora biti prebačen u efektivan ne-vizaulni format. Npr, razlika između susjednih piksela može se koristiti za prikaz slike. Transformacije ovoga tipa (koje odstranjuju među pikselnu redundanciju) označavaju se kao mapiranje. Nazivaju se još i reverzibilno mapiranje ako se originalni elementi slike mogu rekonstruirati iz transformiranog skupa podataka. Prediktivno kodiranje bez gubitaka eliminira među pikselnu redundanciju blisko smještenih piksela, ekstrakcijom i kodiranjem samo **novih informacija** u svakom pikselu. Ta **nova informacija** u pikselu definira se kao razlika između stvarne i

pretpostavljene vrijednosti piksela. Predskazivač generira pretpostavljenu vrijednost piksela temeljenu na nekom broju iz prijašnjih ulaza. Izlaz iz predskazivača se zatim zaokružuje prema najbližem cijelom broju i upotrebljava za izračun greške.

Promotrimo sada kodiranje slike iz gornjeg primjera:

```
>> e = mat2lpc(f2);                % implementira kodiranje temeljeno na prostornoj redundanciji
>> imshow(mat2gray(e))
>> ntrop(e)
>> c = mat2huff(e);
>> cr = imratio(f2,c)
>> [h, x] = hist(e(:) * 512, 512);          % histogram predikcije greške
>> figure; bar(x,h, 'k');
```

Primjetimo da je entropija greške predikcije znatno manja nego entropija originalne slike, što znaći da se slika s predikcijom greške može kodirati efikasnije nego originalna slika. Histogram predikcije greške ima vrh u nuli i relativno malu varijancu u usporedbi s distribucijom originalne sive slike. Ovo se reflektira u odstranjenju velike količine među pikselne redundancije.
Ako sada dekodiramo **c** i usporedimo s početnom slikom **f2**, dobit ćemo:

```
>> g = lpc2mat(huff2mat(c));          % dekodiranje
>> compare(f2,g)
```

Kodiranje nevažnih informacija

Eliminacija nevažnih informacija je bitna stoga što takva informacija sama po sebi nije bitna za ljudski vizualni sustav. Smanjuje kvantitetu te se naziva kvantizacija.

Slijedeća sekvenca naredbi kobinira poboljšanu kvantizaciju sivih nivoa, prediktivno kodiranje i Huffmanovo kodiranje za komprimiranje slike manje od četvrtine njene originalne veličine:

```
>> f = imread('Fig0910(a).tif');
>> q = quantize(f, 4, 'igs');
>> qs = double(q)/16;
>> e = mat2lpc(qs);
>> c = mat2huff(e);
>> imratio(f, c)
```

Kodirani rezultat **c** može se dekomprimirati inverznom sekvencom operacija:

```
>> ne = huff2mat(c);
>> nqs = lpc2mat(ne);
>> nq = 16 * nqs;
>> compare(q, nq)
>> compare(f, nq)
```

Primjetimo da kvadrat srednje vrijednosti greške dekomprimirane slike je otprilike sedam sivih nivoa, te da je ova greška rezultat samog koraka kvantizacije.

Prikažite rezultantne slike.

Laboratorijska vježba 8: **Morfološki operatori**

Osnovne morfološke operacije su erozija i dilatacija. Erozija sa slike briše grupe piksela koji odgovaraju zadanom uzorku, dok dilatacija maleno područje oko piksela popunjava po zadanom uzorku. Zavisno od tipa slike (binarna, gray-scale, colour), definicije dilatacije i erozije razlikuju se i moraju se razmatrati odvojeno.

Dilatacija slike

Za dilataciju slike koristimo funkciju *imdilate()*. Funkcija *imdilate* prihvaća dva osnovna argumenta:

- sliku koju treba obraditi (grayscale, binary, ili packed binary image),
- te strukturni element, koji dobije iz *strel* funkcije, ili binarnu matricu koja definira susjedstvo strukturnog elementa.

Primjer 1:

Ovaj primjer vrši dilataciju jednostavne binarne slike koja sadrži pravokutan objekt.

```
BW = zeros(9,10)
BW(4:6,4:7) = 1
```

Kako bi proširili sve strane objekta, primjer koristi matricu kvadratnog strukturnog elementa veličine 3x3.

```
SE = strel('square',3)
```

Za dilataciju slike, proslijedite sliku BW i strukturni element SE funkciji *imdilate*.

```
BW2 = imdilate(BW,SE)
```

Primjer 2:

```
A=imread('Fig1006(a).tif');
B=[0 1 0;1 1 1;0 1 0];
D=imdilate(A,B);
imshow(D);
figure, imshow(A)
```

Erozija slike

Za eroziju slike, koristimo funkciju *imerode()*. Funkcija *imerode* prihvaća dva osnovna argumenta:

- ulaznu sliku koju želimo obraditi (grayscale, binary, ili packed binary image)

- te strukturni element (koji se dobije iz *strel* funkcije, ili binarnu matricu koja definira susjedstvo strukturnog elementa).

Primjer 3:

```
A=imread('Fig1008(a).tif');
se = strel('disk',10);
E10 = imerode(A,se);
imshow(E10)

se = strel('disk',5);
E5 = imerode(A,se);
imshow(E5)

E20 = imerode(A, strel('disk', 20));
imshow(E20)
```

Primjer 4:

Sljedeći primjer vrši eroziju binarne slike circbw.tif:

1. Učitajte sliku u MATLAB radni prostor.

```
BW1 = imread('circbw.tif');
```

2. Kreirajte strukturni element.
   Sljedeći kod kreira dijagonalni strukturni element.

```
SE = strel('arbitrary',eye(5));
SE=
```

   Susjedstvo:

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

3. Pozovite funkciju *imerode*, i proslijedite joj sliku BW1 i strukturni element SE kao argumente.

```
BW2 = imerode(BW1,SE);
```

4. Primijetite dijagonalne linije na desnoj strani izlazne slike koji nastaju zbog oblika strukturnog elementa.

```
imshow(BW1)
figure, imshow(BW2)
```

Otvaranje slike

Morfološko otvaranje može se koristiti za uklanjanje malih objekata sa slike uz istodobno očuvanje oblika i veličine većih objekata na slici.

Primjer 5:

koristeći funkciju *imopen* uklonite sve linije kruga iz originalne slike, *circbw.tif*, tako što ćete kreirati izlaznu sliku koja sadržava samo pravokutne oblike mikročipova. Za morfološko otvaranje slike provedite sljedeće korake:

1. Učitajte sliku u MATLAB radnu okolinu.

```
BW1 = imread('circbw.tif');
```

2. Kreirajte strukturni element.

```
SE = strel('rectangle',[40 30]);
```

Strukturni element treba biti dovoljno velik da bi uklonio linije nakon erodiranja slike, ali ne prevelik da ukloni i pravokutnike. Treba sadržavati sve 1-ice, tako da uklanja sve osim velikih kontinuiranih površina objekta.

3. Erodirajte sliku strukturnim elementom.

```
BW2 = imerode(BW1,SE);
imshow(BW2)
```

Ovim su uklonjene linije, ali su smanjeni i pravokutnici.

4. Za obnavljanje veličine pravokutnika na originalnu veličinu, izvršite dilataciju erodirane slike koristeći isti strukturni element, SE.

```
BW3 = imdilate(BW2,SE);
imshow(BW3)
```

Primjer 6:

Učitajmo sliku sa šumom i otvorimo je:

```
f = imread('Fig1011(a).tif');
se = strel('square', 3);
fo = imopen(f, se);
```

Šum se reducira otvaranjem slike, ali se stvaraju i neželjene praznine na linijama otiska prsta. Praznine premošćujemo zatvaranjem slike:

```
foc = imclose(fo, se);
subplot(1,3,1), imshow(f);
subplot(1,3,2), imshow(fo);
subplot(1,3,3), imshow(foc);
```

Primjer 7:

Thinning – reduciranje binarnog objekta u slici na debljinu jednog piksela. U ovom primjeru želimo stanjiti konture otiska prsta da bi bile odgovarajuće debljine za obradu, analizu i usporedbu. Svako pozivanje funkcije bwmorph uklanja jedan ili dva piksela od debljine objekta:

```
f = imread('Fig1011(c).tif');
g1 = bwmorph(f, 'thin', 1);
g2 = bwmorph(f, 'thin', 2);
subplot(1,3,1),imshow(f)
subplot(1,3,2),imshow(g1)
subplot(1,3,3),imshow(g2)
```

Postavlja se pitanje: koliko takne želimo da budu konture? Funkcija *bwmorph* dozvoljava kao parametar i INF (beskonačnu) vrijednost. U tom slučaju, kada je postavljen parametar INF u pozivu funkcije *bwmorph,* funkcija se ponavlja sve dok se slika ne prestane mijenjati. Ovo se naziva ponavanje operacije do postizanja stabilnosti.

```
ginf = bwmorph(f, 'thin', Inf);
subplot(2,2,1),imshow(f)
subplot(2,2,2),imshow(g1)
subplot(2,2,3),imshow(g2)
subplot(2,2,4),imshow(ginf)
```

Skeletonizacija – drugi način za reduciranje binarnog objekta u slici na debljinu jednog piksela.

```
fs = bwmorph(f, 'skel', Inf);
imshow(f); figure, imshow(fs)
```

Skeletonizacija, kao i *thinning* često proizvode kratke stršeće ogranke , koje nazivamo parazitnim komponentama. Proces odstranjivanja ovih ogranaka naziva se obrezivanje. U tu svrhu koristimo funkciju *endpoints()*, koja iterativno identificira i odstranjuje ogranke.

Slijedeći kod postprocesuira skeleton sliku **fs** kroz pet iteracija *endpoint* obrezivanja:

```
for k = 1:5
fs = fs & ~endpoints(fs);
end
```

Slični rezultati mogu se dobiti i pozivom funkcije bwmorph sa 'spur' opcijom:

```
fs = bwmorph(fs, 'spur', 5);
```

Upotrebom parametra Inf umjersto 5, bwmorph bi reducirao sliku do skoro jedne točke.

Zadatak

1. Upišite i komentirajte rezultate dobivene izvršavanjem sljedećeg koda:

```
BW1 = imread('circbw.tif');
BW2 = bwmorph(BW1,'skel',Inf);
imshow(BW1)
figure, imshow(BW2)
```

2. Upišite i komentirajte rezultate dobivene izvršavanjem sljedećeg koda:

```
BW1 = imread('circles.png');
figure, imshow(BW1)
BW2 = bwmorph(BW1,'remove');
BW3 = bwmorph(BW1,'skel',Inf);
figure, imshow(BW2)
figure, imshow(BW3)
```

Laboratorijska vježba 8:

Klasifikacija objekata u slici

Prepoznavanje objekta

Prepoznavanje objekta u slici svodi se na prepoznavanje rubova i regija. Za prepoznavanje nekakvog uzorka u slici potrebi su kvantitativni deskriptori kao što su dužina, površina, tekstura, odnosno strukturni elementi poput nizova. Što se tiče samog prepoznavanja I identifikacije objekta u slici uvodi se pojam "učenja" iz primjera uzoraka.

Uzorak možemo shvatiti kao aranžman deskriptora, odnosno značajki koje su svojstvene određenom skupu *i.e.* uzorci koji dijele skup istih značajki pripadaju istoj skupini. Automatsko prepoznavanje uzoraka je računarska tehnika dodjeljivanja uzoraka određenoj klasi sa što manjom ljudskom intervencijom. Najčešći kvantitativni descriptor je u svojoj suštini vektor, a strukturni je niz. Bazni koncept u prepoznavanju, a posebno u aplikacijama teorijskog odlučivanja je ideja o podudaranju uzoraka koja je temeljena na mjerama udaljenosi između vektora uzoraka.

Računanje mjera udaljenosti u matlabu

Standardne *for* i *while* petlje zamjenjuju se računanjem vektoriziranih udaljenosti.

Euclidova distanca između dva *n* dimenzionalna vektora **x** i **y** definira se kao skalar:

$$D(x,y) = \|x-y\| = \|y-x\| = [(x_1-y_1)^2 + (x_2-y_2)^2 + \cdots + (x_n-y_n)^2]^{\frac{1}{2}}$$

Ovaj izraz u MatLabu se računa kao druga norma udaljenosti između dva vektora:

D=norm(x,y)

Najčešće je potrebno izračunati skup Euclideovih udaljenosti između vektora **y** i svakog pojedinog vekora iz populacije **p** *n* dimenzionalnih vektora, složenih u retke $p \times n$ matrice **X.** Da bi se dimenzije složile, **y** mora biti dimenzija $1 \times n$. Stoga se udaljenost između **y** i svakog retka matrice **X** računa kao:

D=sqrt(sum(abs(x-repmat(y, p, 1)).^2,2));

Gdje je D*(i)* Euclideova udaljenost između **y** i svakog *i*-tog retka od **X**.

Funkciju *repmat* upotrebljavamo za dupliciranje vector retka **y** *p* puta i na takav način stvaramo $p \times n$ matricu koja će odgovarati veličini matrice **X**.

Poslijednji broj 2 u kodu označava da se suma mora izvršiti po drugoj dimenziji *i.e.* da sumira elemente preko horizontalne dimenzije.

repmat naredba može se zamijeniti novijom MatLab funkcijom *bsxfun*, koja istu radnju obavlja sa manje memorije, odnosno brže:

C = bsxfun(fun, A, B)

Ova funkcija primjenjuje operaciju element po element na nizove **A** i **B**, definiranu sa **fun**, koja je pak jedna od ugrađenih funkcija ili M funkcija:

TABLE 13.1 Built-in functions for function `bsxfun`.

Function	Explanation	Function	Explanation	Function	Explanation
`@plus`	Plus	`@min`	Minimum	`@lt`	Less than
`@minus`	Minus	`@rem`	Remainder after division	`@le`	Less than or equal to
`@times`	Array multiply	`@mod`	Modulus after division	`@gt`	Greater than
`@rdivide`	Right array divide	`@atan2`	4-quadrant arctangent	`@ge`	Greater than or equal to
`@ldivide`	Left array divide	`@hypot`	Sq. root of sum of squares	`@and`	Logical AND
`@power`	Array power	`@eq`	Equal	`@or`	Logical OR
`@max`	Maximum	`@ne`	Not equal	`@xor`	Logical exclusive OR

Primjer:

```
x=[1 2; 3 4; 5 6]
y=[1 3]
bsxfun(@minus, x,y)
```

Podudaranje preko korelacije

Uz zadanu sliku *f(x,y)* problem korelacije je pronaći sva mjesta u slici koja odgovaraju maski $\omega(x, y)$, koja je manja od zadane slike. To se postiže upotrebom funkcije

g == normcorr2(template, f)

Primjer:

Pronaći oko (centar) uragana u slici.

```
f = imread('Fig1301(a).tif');
w = imread('Fig1301(b).tif');

g = abs(normxcorr2(w, f));
imshow(g)

% pronađimo sve maksimalne vrijednosti
gT = g ==max(g(:));    %  gT je logički niz

% otkrijmo koliko postoji vrhova
idx = find(gT ==1);
numel(idx)

% jednu točku se teško može vidjeti, stoga je povečajmo
gT =imdilate(gT, ones(7));

figure, imshow(gT)
```

Značajka od interesa je doznati lokaciju največe podudarnosti, odnosno lokaciju največe vrijednosti u slici korelacije. Lokaciju vrhunca pronalazimo slijedećim kodom:

```
[r, c] = ind2sub(size(f), idx)
```

Prepoznavanje temeljeno na bayesovom klasifikatoru

Bayesovo prepoznavanje se često koristi za automatiziranje klasifikacije regija u multispektralnim slikama. Slijedi primjer s četiri slike (R, G, B, IR). Cilj ovog primjera je upotreba Bayesovog (statističkog) klasifikatora za podjelu piksela u tri klase: voda, urbano područje i vegetacija.

Slike kao i maske učitavamo slijedećim kodom (maske su prethodno generirane *roipoly()* funkcijom):

```
f1 = imread('Fig1302(a).tif');
f2 = imread('Fig1302(b).tif');
f3 = imread('Fig1302(c).tif');
f4 = imread('Fig1302(d).tif');

B1 = imread('Fig1302(e)[mask_B1].tif');
B2 = imread('Fig1302(e)[mask_B2].tif');
B3 = imread('Fig1302(e)[mask_B3].tif');
```

Četiri slike se lijepe zajedno preko treće dimenzije, te dobivamo jedan stog slika:

```
stack = cat(3, f1, f2, f3, f4);
```

T1, T2 i T3 su primjeri za treniranje procjene matrice kovarijance i vektora srednje vrijednosti:

```
[X1, R1] = imstack2vectors(stack, B1);
[X2, R2] = imstack2vectors(stack, B2);
[X3, R3] = imstack2vectors(stack, B3);
T1 = X1(1:2:end, :);
T2 = X2(1:2:end, :);
T3 = X3(1:2:end, :);
```

Matrica kovarijance i vektor srednjih vrijednosti:

```
[C1, m1] = covmatrix(T1);
[C2, m2] = covmatrix(T2);
[C3, m3] = covmatrix(T3);
CA = cat(3, C1, C2, C3);
MA = cat(2, m1, m2, m3);
```

Performansa klasifikatora sa uzorcima za treniranje:

```
dT1 = bayesgauss(T1, CA, MA);
dT2 = bayesgauss(T2, CA, MA);
dT3 = bayesgauss(T3, CA, MA);
```

Rezultati klasifikacije uzoraka za treniranje:

```
class1_to1 = numel(find(dT1==1));
class1_to2 = numel(find(dT1==2));
class1_to3 = numel(find(dT1==3));

class2_to1 = numel(find(dT2==1));
class2_to2 = numel(find(dT2==2));
class2_to3 = numel(find(dT2==3));
```

```
class3_to1 = numel(find(dT3==1));
class3_to2 = numel(find(dT3==2));
class3_to3 = numel(find(dT3==3));

I1 = X1(2:2:end, :);
I2 = X2(2:2:end, :);
I3 = X3(2:2:end, :);

image2 = false(size(f2));
d2 = bayesgauss(X2, CA, MA);
idx2 = find(d2 ==2);
image2(R2(idx2)) = 1;

image1 = false(size(f1));
d1 = bayesgauss(X1, CA, MA);
idx1 = find(d1 ==2);
image1(R1(idx1)) = 1;

image3 = false(size(f3));
d3 = bayesgauss(X3, CA, MA);
idx3 = find(d3 ==2);
image3(R3(idx3)) = 1;

compositeImage = image1 | image2 | image3;
imshow(compositeImage)
```

Upotrebom vektora srednjih vrijednosti i matrica kovarijanci dobivenih iz seta za treniranje, klasifikacija SVIH piksela u slici u jednu od tri kategorije se radi na način:

```
B = ones(size(f1));
X = imstack2vectors(stack, B);
dAll = bayesgauss(X, CA, MA);
image_class1 = reshape(dAll == 1, 512, 512);
image_class2 = reshape(dAll == 2, 512, 512);
image_class3 = reshape(dAll == 3, 512, 512);
figure, imshow(image_class1)
figure, imshow(image_class2)
figure, imshow(image_class3)
```

Bijeli pikseli klasificirani su kao voda, dok oni što nisu voda prikazani su kao crni. Bayesov klasifikator je odlično razvrstao piksele. Na drugoj slici bijeli pikseli predstavljaju urbano područje, što se odlično očituje kod prikaza urbanih značajki kao što su mostovi i autoceste. Poslijednja slika predstavlja vegetaciju. Centralno područje u slici pokazuje malu gustoću bijelih piksela u samom centru grada (vegetacije), dok prema perifieriji gustoća vegetacije raste, a gustoća urbanog područja opada.

BIBLIOGRAFIJA

- **Gonzalez, Woods**: *Digital Image Processing, Second Edition*, 2002
- **Gonzales, Woods, Eddins**: *Digital Image Processing Using Matlab*, 2005
- **Sven Lončarić**: *Digitalna obrada i analiza slike, predavanja*, FER Zagreb
- **R. Hartley, A. Zisserman**: *Multiple View Geometry in Computer Vision*, Second Edition, Cambridge University Press 2003.
- http://www.robots.ox.ac.uk/~vgg/research/
- **Hrvoje Dujmić:** *Multimedijski sustavi II dio, FESB, listopad 2010. (Interna skripta)*
- **David Capel**: *Image mosaicing and Super resolution*, PhD thesis, Springer Science & Business Media, Jan 19, 2004, distinguished disertations.
- **Barbara Džaja**: *Vjerojatnosni model super-rezolucijske slike u boji*, 2013, doktorska disertacija.