

Upute za laboratorijske vježbe iz digitalne obradbe i analize slike

Hrvoje Kalinić, Tomislav Petković, Sven Lončarić

15. lipnja 2010.

Sadržaj

1. Uvod	3
2. Vježba 1 - Digitalan zapis slike	4
2.1. Uvod	4
2.2. Priprema radnog okruženja	4
2.3. Učitavanje, prikaz i snimanje slike	5
2.4. Računanje sa slikama	9
2.5. Prikaz boja i transformacije boja	12
3. Vježba 2 - Operacije na slici	15
3.1. Unarne operacije na slici	15
3.2. Binarne operacije na slici	16
3.3. Digitalna angiografija	17
3.4. Gama korekcija	18
3.5. Linearna konvolucija	19
4. Vježba 3 - Otipkavanje i kvantizacija	21
4.1. Kvantizacija	21
4.2. Otipkavanje	22
4.3. Pikselizacija	23
4.4. Alias-efekt	24
4.5. Moarški efekt	26
5. Vježba 4 - Frekvencijske transformacije	28
5.1. Diskretna Fourieova transformacija	28
5.2. DFT i geometrijske transformacije slike	30
5.3. Filtriranje slike	31
5.4. Diskretna kosinusna transformacija	33
5.5. DCT i kompresija slike	35
6. Vježba 5 - Poboljšanje slike	37
6.1. Histogram prvog reda	37
6.2. Izjednačavanje histograma	37
6.3. Modeliranje histograma	38
6.4. Usrednjavanje i median filter	38
6.5. Uklanjanje neoštine	40

7. Vježba 6 - Obnavljanje slike	43
7.1. Uvod	43
7.2. Obnavljanje slike	44
7.3. Modeliranje degradacije slike kao FIR filtra	44
7.4. Inverzni filter	46
7.5. Pseudoinverzni filter	47
7.6. Wienerov filter	48
8. Vježba 7 - Pronalaženje značajki slike	51
8.1. Prostorne značajke	51
8.2. Amplitudne značajke slike	51
8.3. Značajke histograma prvog reda	52
8.4. Histogram drugog reda	52
8.5. Detekcija rubova	53
8.6. Sobelov i Prewittov operator	53
8.7. Kompas operatori za detekciju ruba	55
8.8. Laplaceov operator	56
8.9. Značajke teksture	57
9. Vježba 8 - Segmentacija slike	59
9.1. Uvod	59
9.2. Amplitudna segmentacija	59
9.3. Ručno odabiranje praga	59
9.4. Automatsko odabiranje praga	61
9.5. Određivanje rubova	61
9.6. Segmentacija tekstura	62
10. Vježba 9 - Registracija slike	64
10.1. Uvod	64
10.2. Registracija slike	64
11. Dodatci	67

Poglavlje 1.

Uvod

Svrha laboratorijskih vježbi je da studenti kroz praktične zadatke i neposredno rješavanje konkretnih problema iz područja obrade i analize slike steknu dodatna znanja o primjeni teorije i usvoje vještine za rješavanje realnih problema. Ove upute za laboratorijske vježbe su nastale u sklopu predmeta Digitalna obradba i analiza slike s namjerom da omoguće samostalan rad studentima i posluže kao vodič za rješavanje osnovnih tipičnih problema u digitalnoj obradi slike. Sve vježbe se izvode na računalu, a možete koristiti Octave, odnosno MATLAB te njegov paket za obradbu slike (Image Processing Toolbox).

Bazu slika, koje se obično koriste za evaluaciju algoritama digitalne obradbe slike možete pregledati na nekoj od slijedećih adresa:

1. <http://sipi.usc.edu/services/database/>
2. <http://peipa.essex.ac.uk/ipa/pix/mias/>
3. <http://staff.science.uva.nl/~mark/ALOT/>
4. <http://staff.science.uva.nl/~aloi/>

a mali uzorak slika za testiranje svojih algoritama možete pronaći i na stranici predmeta. Neke od slika iz ovih baza koristiti ćemo i za primjere i zadatke na laboratorijskim vježbama.

Eventualne pogreške ili nejasnoće u vezi laboratorijskih vježbi javite asistentu elektroničkom poštom na hrvoje.kalinic@fer.hr.

Poglavlje 2.

Vježba 1 - Digitalan zapis slike

2.1. Uvod

U ovoj vježbi pozabavit ćemo se učitavanjem i spremanjem slike, osnovnim funkcijama za prikazivanje i matičnim operatorima koje možemo primijeniti nad digitalnom slikom. Počet ćemo sa slikama koje sadrže samo sive tonove (sive slike), te pokazati kako taj zapis slike možemo proširiti na slike u boji. Posebno ćemo se osvrnuti na RGB kao osnovni prostor boja i prikaz digitalne slike u njemu, odnosno na indeksirane slike i njihov zapis u računalu.

U računalu, siva 2D slika je reprezentirana kao $M \times N$ matrica. RGB slika će biti reprezentirana kao tri takve matrice odnosno matrica veličine $M \times N \times 3$. Jednako tako slike u drugim prostorima boja ili drugim spektrima mogu biti reprezentirane kao višedimenzionalne matrice. Stoga se za obradbu i analizu slike mogu koristiti sve uobičajene funkcije za rad s matricama (matični operatori) uz funkcije namjenjene upravo za rad sa slikama.

2.2. Priprema radnog okruženja

Sa stranica predmeta dohvatite datoteke (funkcije i slike) potrebne za vježbu i kopirajte ih u radni direktorij (npr. `/home/ime/radni_direktorij`). Ukoliko ne želite da vam se funkcije koje pišete za vježbu mješaju s funkcijama koje ste sami napisali možete ih kopirati u bilo koji drugi direktorij (npr. `/home/ime/radni_direktorij/funkcije`). U tom slučaju prije pokretanja naredbe za koje su one potrebne morate ih uključiti u putanju, što činite naredbom:

2.2.1. Primjer

```
>> addpath('/home/ime/radni_direktorij/funkcije')
```

Na isti način možete i uključiti i direktorij s slikama, odnosno bazu slika s nekih drugih internet stranica. Za navigaciju kroz direktorije možete koristiti standardne UNIX naredbe (`ls`, `cd` ...).

Upute za korištenje funkcije možete dobiti kroz terminal upisujući:

2.2.2. Primjer

```
>> help ime_naredbe
```

Uz to MATLAB podržava i korisnički vrlo pristupačan "HelpDesk" koji možete pozvati naredbom:

2.2.3. Primjer

```
>> helpdesk
```

2.3. Učitavanje, prikaz i snimanje slike

Da bi mogli vršiti operacije na slikama najprije je potrebno učitati sliku. To vršimo naredbom *imread()*:

2.3.1. Primjer

```
>> [img, map] = imread('salona.png'); % učitavamo sliku
>> whos
  Name          Size          Bytes  Class      Attributes
  img           300x613        183900  uint8
  map           256x3          6144   double
```

U ovom primjeru učitano sliku u boji *canoe.tif*¹ predstavljamo dvodimenzionalnom matricom *img* u kojoj su spremljene vrijednosti indeksa elemenata u paleti *map* koja ima 256 boja. Učitano sliku možemo pogledati kao i svaku ostalu varijablu unutar glavnog prozora (terminala) zadavajući ime. Time samo ispisujemo numeričke vrijednosti elemenata matrice *img*, što nije pravi vizualni prikaz slike. Za vizualni prikaz slike koristimo naredbe *imshow()*, *image()* i *imagesc()*:

2.3.2. Primjer

```
>> img           % ispisuje matricu img

img =

  Columns 1 through 12

  156   156   ...
```

¹TI(F)F je kratica za Tag(ged) Image (File) Format. Taj format zapisa predviđen je isključivo za pohranu raster grafike, a između ostalog podržava i proizvoljan broj bitova po točki. Detaljnije informacije mogu se pronaći na <ftp://ftp.sgi.com/graphics/tiff>.

```
>> image(img);      % otvara novi grafički prozor te u njemu
    % prikazuje sliku, ali s krivim bojama zbog
    % krive palete (početna paleta je jet-paleta)
>> colormap(map);   % mijenja početnu paletu u ispravnu paletu
```

Kod pozivanja naredbe *image()* program pretpostavlja standardnu paletu boja. Stoga je gotovo uvijek potrebno promijeniti paletu naredbom *colormap()*. Odabranu paletu možemo pogledati na prikazanoj slici zadavanjem naredbe *colorbar* (rezultat naredbi je prikazan na slici 2.1.):

2.3.3. Primjer

```
>> [img, map] = imread('tombak.png'); % učitavamo sliku
>> image(img)           % prikazujemo sliku
>> colormap(map)       % odabiremo paletu
>> colorbar            % prikazujemo paletu
```

Primijetite kako ste isti efekt mogli postići naredbom *imagesc()*, odnosno naredbom *imshow*:

2.3.4. Primjer

```
>> imshow('tombak.png'); % prikazujemo sliku
>> colorbar % ako želimo prikazati paletu
```



Slika 2.1.: Prikaz rezultata operacija navedenih u Primjeru 2.3.3.

Paletu boja je potrebno učitati samo ukoliko je riječ o slikama i boji koje se sastoje od jedne dvodimenzionalne matrice. Ovakve slike se nazivaju indeksiranim slikama. Slike u boji se u računalu najčešće pohranjuju na način da svaku točku slike reprezentiramo s tri vrijednosti intenziteta koji odgovaraju crvenoj ($\lambda = 700nm$) zelenoj ($\lambda = 546.1nm$) i plavoj ($\lambda = 435.8nm$) boji.

Takvu sliku reprezentiramo jednom trodimenzionalnom matricom ($M \times N \times 3$), a pri prikazivanju nije potrebno odabrati paletu jer je boja definiran u ove tri "osnovne" komponente.

Na ovom primjeru je korisno primijetiti da je ishodište slike u gornjem lijevom kutu, što odudara od standardne reprezentacije koordinatnog sustava (poput Kartezijevog). Ovu stvar treba imati na umu jednako kao i činjenicu da je svaka slika zadana s (y, x) koordinatom, umjesto (x, y) , jer će nam biti korisna za izbjegavanje nekih tipičnih i učestalih pogrešaka prilikom digitalne obradbe slike.

2.3.5. Primjer

```
>> [img, map] = imread('kljakovic1.png'); % učitavamo RGB sliku
>> whos
  Name          Size          Bytes  Class  Attributes

  img           446x400x3        535200  uint8
  map           0x0                0      double

>> image(img)                % prikazujemo sliku
```

Sliku u boji jednostavno je rastaviti na komponente:

2.3.6. Primjer

```
>> tocka=img(30,40,:);      % točka s koordinatama (30,40)
>> crvena=img(:,:,1);      % crvena boja je prva komponenta
>> zelena=img(:,:,2);      % zelena boja je druga komponenta
>> plava=img(:,:,3);       % plava boja je treća komponenta
>> whos
  Name          Size          Bytes  Class  Attributes

  crvena       325x487        158275  uint8
  img          325x487x3        474825  uint8
  map          0x0                0      double
  plava        325x487        158275  uint8
  tocka        1x1x3            3      uint8
  zelena       325x487        158275  uint8
```

Ako pokušamo istodobno prikazati crvenu, zelenu i plavu komponentu primijetili bi da naredba *image()* uvijek crta u isti prozor. Da bi istodobno promatrali sve komponente moramo otvoriti tri različita prozora za prikaz. To se postiže naredbom *figure()* koja otvara novi grafički prozor:

2.3.7. Primjer

```
>> figure      % otvaramo novi prozor
>> imshow(crvena) % u prozoru prikazujemo crvenu komponentu
>> figure()    % otvaramo novi prozor
>> imshow(zelena) % u prozoru prikazujemo zelenu komponentu
>> figure; imshow(plava)
                    % u novom prozoru prikazujemo plavu komponentu
```

Svaka od slika predstavlja intenzitet svjetlost određene valne duljine ($\lambda = 700nm, 546.1nm$ ili $435.8nm$), dok smo ih mi prikazali kao sive slike. Kako prikazati slike s odgovarajućom paletom boja (crvenom, zelenom i plavom, koje odgovaraju valnim duljinama svjetlosti koje pojedine komponente prenose), bit će opisano u poglavlju 2.5..

2.3.8. Zadatci

1. Ponovite prethodni primjer s naredbama *imagesc()* i *image()* da biste uočili razliku između naredbi za prikaz slike.

Naredbe *image()* i *imagesc()* služe za prikaz matrice kao slike i matricu uvijek prikazuju kao sliku u boji. To rade iz razloga što ljudsko oko bolje razlikuje boje nego nijanse sive, čime su detalji u slici bolje vidljivi.² Razlika među njima je što naredba *image()* pretpostavlja da je slika u $[0, 255)$, što znači da vrijednosti van tog raspona prikazuje kao minimalnu ili maksimalnu vrijednost, tako npr. broj 500 ne razlikuje od broja 255, a -40 ne razlikuje od 0, te ih prikazuje istim intenzitetom u slici. S druge strane naredba *imagesc()* vrijednosti matrice koje mogu biti u bilo kojem rasponu $[A, B)$ preslikava na raspon $[0, 255)$, što nije uvijek poželjno jer mijenja kontrast slike. Naredba *imshow()* prikazuje sliku onako kako je ona zapisana u računu. Ako njome prikazujemo matricu, naredba se prema njoj ponaša kao da je slika, tj. pretpostavlja da je u rasponu $[0, 1)$, ako je tipa *double* odnosno $[0, 255)$, ako je tipa *uint8*.

Naredba *figure()* bez argumenata otvara novi prozor. Zadamo li joj kao argument cijeli broj naredba otvara novi prozor ako prozor s tim brojem ne postoji, a inače čini aktivnim prozor s tim brojem. Sve naredbe za crtanje koriste trenutno aktivni prozor:

2.3.9. Primjer

```
>>(gcf)

ans =
```

²Načinima na koji možemo promijeniti paletu boja koju slika koristi, govorit ćemo u poglavlju 2.5.

```
>> imagesc(plava)      % u prozoru 4 prikazujemo plavu komponentu
>> figure(3)          % sada je prozor 3 aktivan
```

Bilo koju dvodimenzionalnu matricu možemo snimiti kao sliku, no pri tome treba paziti na ograničenja standardnih formata računalnog zapisa slike. Kod učitavanja slike ste primijetili da su vrijednosti intenziteta tipa *uint8* (cijeli broj zapisan s 8 bita), dok je matrica obično tipa *double*. U takvim slučajevima je matricu koju želimo pohraniti kao sliku potrebno pretvoriti u tip *uint8*, npr. skaliranjem i zaokruživanjem vrijednosti. Jednom kada imamo matricu tipa *uint8* koristimo naredbu *imwrite()*:

2.3.10. Primjer

```
>> imwrite(plava, 'plava.jpg'); % snimamo plavu komponentu
                                % u datoteku plava.jpg
>> img = (rand(90,90)-0.5)*70;   % napravimo novu sliku (šum)
>> img = uint8(round((img+35)*255/100));
                                % transformacija intenziteta
>> imwrite(img, 'nered.jpg');   % snimamo matricu
```

Kako će izgledati slika zapisana naredbom *imwrite()* možete pogledati naredbom *imshow()*. Za detaljnije upute o korištenju naredbi *imread()*, *imshow()* i *imwrite()* pogledajte ugrađenu pomoć.

2.3.11. Zadatci

1. Pozicionirajte se u svoj radni direktorij te učitajte neke od slika, prikažite ih na gore opisani način te spremite u različitim formatima.
2. 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 u intervalu $[-100, 300]$. Prikažite ih korištenjem funkcija *image()* i *imagesc()*, te ih snimite korištenjem funkcije *imwrite*, sa i bez pretvorbe u *uint8* tip. Snimljene slike pregledajte u nekom pregledniku slika. Komentirajte razlike.

2.4. Računanje sa slikama

Jednom kada učitamo slike i pohranimo ih kao matrice, nad njima možemo koristiti svaku ugrađenu funkciju koja zna rukovati s matricama. Neke funkcije rukuju samo s vektorima, odnosno daju željene rezultate samo ako je ulaz funkcije vektor. U tom slučaju moramo zati kako matricu pretvoriti u vektor. Najjednostavniji način za to je da sve stupce slijedno dodajemo na

kraj prethodnog, pretvarajući matricu u vektor stupac. To radimo naredbom za referenciranje svih vrijednosti matrice, dvotočkom (:). Vektor pretvaramo u matricu naredbom *reshape()*:

2.4.1. Primjer

```
>> a = [1, 2; 3,4]
```

```
a =
```

```
    1    2
    3    4
```

```
>> a(:)
```

```
ans =
```

```
    1
    3
    2
    4
```

```
>> reshape(a(:),2,2)
```

```
ans =
```

```
    1    2
    3    4
```

```
>>
```

Ukoliko matrica sadrži *uint8* tip podataka, često je moramo pretvoriti u tip *double*, da bismo mogli koristiti neke od funkcija za rad sa slikama. To činimo naredbom *double()*:

2.4.2. Primjer

```
>> img = imread('salona.png');      % učitavamo sliku salona.png
>> max(img(:))      % provjerimo maksimalnu vrijednost intenziteta
```

```
ans =
```

```
    255
```

```
>> whos
```

Name	Size	Bytes	Class	Attributes
ans	1x1	1	uint8	
img	300x613	183900	uint8	% tip je uint8

```

>> A = 3*img; % transformacija intenziteta
>> max(A(:)) % provjerimo maksimalnu vrijednost intenziteta

ans = % transformacija intenziteta nije valjana nad
      % tipom uint8 jer dovodi do preljeva
      255
>> A = double(img); % pretvaramo je u double tip
>> A = 3*A; % operacija je sada valjana
>> max(A(:)) % provjera

ans =

      765

```

Želimo li računati s više slika ili kombinirati slike npr. zbrajanjem, obje matrice koje sadrže slike moraju biti istih dimenzija. Promjenu dimenzija (skaliranje) slike vršimo naredbom *imresize()*.

2.4.3. Primjer

```

>> [img1, map] = imread('medalja_kamenita_vrata.png');
>> [img2, map] = imread('medalja_dubrovnik.png');
>> whos

```

Name	Size	Bytes	Class	Attributes
img1	600x600	360000	uint8	% prva slika je 600x600
img2	545x548	298660	uint8	% druga slika je 545x548
map	0x0	0	double	

```

>> img2=imresize(img2,[600 600],'bilinear');
>> size(img2) % sada je i prva slika 600x600

ans =

      600      600

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

```

Funkcija *imresize()* podržava različite metode interpolacije, a mi smo odabrali linearnu. Za ostale metode interpolacije pogledajte ugrađenu pomoć.

Primijetite da smo rezultat prikazali koristeći naredbu *imagesc()*. Naime, gotovo uvijek nakon niza operacija nad matricama vrijednosti više neće biti cjelobrojne u intervalu od 0 do 255, a vrijednosti intenziteta za računalni prikaz slika su uobičajeno 8-bitne. Stoga bi korištenjem naredbe *image* koja očekuje 8-bitne podatke dobili prikaz matrice koji ne iskorištava dostupnu paletu u potpunosti (sve prevelike vrijednosti postaju 255, a premale 0). Najčešće je potrebno skalirati matricu prije korištenja naredbe *image* ili, alternativno, koristiti naredbu *imagesc()* koja odmah linearno skalira matricu tako da najveća vrijednost postaje 255, a najmanja 0.

2.4.4. Zadatci

1. Odaberite dvije slike različitih dimenzija, oduzmite ih te prikažite rezultat. Isprobajte više različitih metoda interpolacije³ pri mijenjanju dimenzija i komentirajte razlike.

2.5. Prikaz boja i transformacije boja

U prethodnom dijelu vježbe već smo pokazali da je za indeksirane slike bitna paleta boja. U ovom dijelu vježbe bit će prikazane različite palete boja koje se mogu koristiti. Paleta koja se sastoji od m boja je matrica veličine $m \times 3$, gdje je svaka od m boja opisana s tri vrijednosti intenziteta. Te vrijednosti intenziteta opisuju udio crvene, zelene i plave boje, s time da 0 odgovara minimalnom, a 1 maksimalnom intenzitetu. Na taj način se može definirati bilo koja paleta. Standardne palete boja se mogu dobiti pozivanjem istoimenih funkcija *hsv*, *gray*, *hot*, *cool*, *bone*, *copper*, *pink*, *jet*, *lines*...

2.5.1. Primjer

```
>> [img,map]=imread('tombak.png');      % učitajmo sliku
>> max(img(:))

ans =

    255          % najveća vrijednost u slici je 255

>> image(img)          % prikažemo sliku
>> colormap(gray(256)) % odaberemo paletu
```

Naredbom *gray(256)* kreirali smo sivu paletu koja ima 256 nijansi sive. Umjesto korištenja navedenih paleta možemo proizvoljno definirati vlastitu paletu te svakoj vrijednosti elementa slike pridružiti neku boju. Ako boju

³Primijetite da je naredbom *imresize()* nemoguće ispravno interpolirati slike s paletom. Takve slike pretvorite u sive prije interpolacije. Alternativno, možete takve slike prikazati u nekom od prikladnih prostora boja te tada izvršiti interpolaciju.

pridružujemo s ciljem isticanja nekih elemenata slike postupak se naziva pseudokoloriranje. Većina algoritma za digitalnu obradbu slike osmišljena je za rad sa slikama u kojima je svaka točka predstavljena samo s vrijednošću intenziteta. Zato će i tokom ovih laboratorijskih vježbi većina algoritama biti će primijenjena na sive slike. Slike u boji pretvaramo u sive slike korištenjem naredbe `rgb2gray()` ako je slika zadana u RGB prostoru boja:

2.5.2. Primjer

```
>> [img,map]=imread('prof_baltazar.png');
>> image(img) % prikažimo sliku u boji
>> intenzitet=rgb2gray(img); % pretvorimo je u sivu sliku
>> figure, image(intenzitet) % nacrtajmo i nju te usporedimo
% dvije slike
>> colormap(gray)
```

Osim RGB zapisa možemo naći i druge zapise slika u boji, npr. za HSV zapis koristimo `hsv2rgb()` funkciju. Općenito, ako imamo indeksiranu sliku s poznatom paletom koristimo `ind2gray()`.

2.5.3. Zadatci

1. Učitajte sliku *numbers.512.tiff* koja se nalazi u USC-SIPI baza slika. Isprobajte na njoj različite palete boja.

Osim standardnih paleta boja možemo koristiti i vlastite palete boja, osmišljene za neku posebnu namjenu. Prisjetimo se primjera 2.3.7., te razmotrimo kako bismo mogli prikazati komponente slike s odgovarajućom paletom boja (crvenom, zelenom i plavom), umjesto da ih sve gledamo kao sive slike. To možemo učiniti na slijedeći način:

2.5.4. Primjer

```
>> [img, map] = imread('kljakovic1.png');
>> load rgb_paleta % učitajmo potrebne palete boja
>> figure; image(img(:,:,1))
>> figure; image(img(:,:,2))
>> figure; image(img(:,:,3)) % prikažimo komponente slike
>> figure(1)
>> colormap(crvena_paleta)
>> figure(2)
>> colormap(zelena_paleta)
>> figure(3)
>> colormap(plava_paleta)
```

Ukoliko sve tri komponente slike želimo prikazati u jednom prozoru, poslužiti ćemo se slijedećim trikom:

2.5.5. Primjer

```
>> img1 = im2double(img);    % pretvaramo sliku u tip double
>> img2 = img1/max(img1(:));
           % želimo da maksimalna vrijednost slike bude 1
>> figure; imshow([img2(:,:,1)/3, ... % crtamo sve 3 komponente
                   (1+img2(:,:,2))/3, (2+img2(:,:,3))/3 ])
>> colormap(paleta)         % postavljamo željenu paletu boja
```



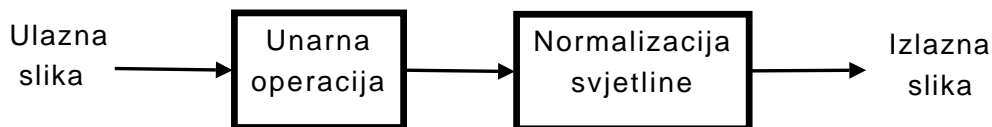
Slika 2.2.: Prikaz rezultata operacija navedenih u Primjeru 2.5.5.

Poglavlje 3.

Vježba 2 - Operacije na slici

3.1. Unarne operacije na slici

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. Blok shema unarne operacije prikazana je slikom 3.1..



Slika 3.1.: Blok shema unarne operacije.

Unarna operacija je bilo koja matematička funkcija koja mijenja vrijednosti točaka¹ u slici, te kodomena te funkcije ne mora biti jednaka domeni. Iz tog razloga, 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.

3.1.1. Primjer

```
>> [img,map]=imread('medalja_kamenita_vrata.png');  
>> img=double(img); % većina funkcija traži double tip  
>> imgU=sqrt(img); % izvršimo unarnu operaciju  
>> max(imgU(:))
```

ans =

```
15.9687 % najveći element nije više 255
```

¹koristit ćemo također i sinonime piksel i piknjica

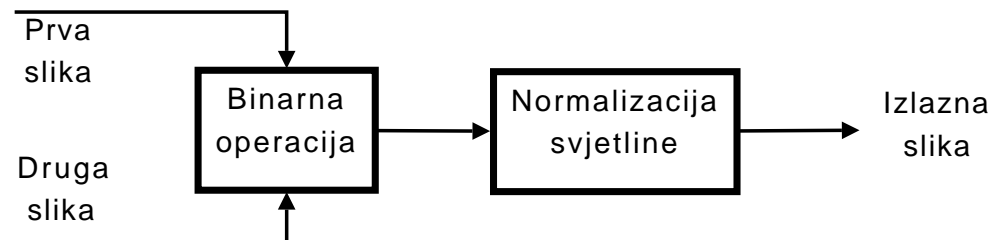

```
>> imagesc(imgU)    % odmah skaliramo i prikazujemo sliku
>> colormap(gray)  % ne zaboravimo promijeniti paletu
```

3.1.2. Zadatci

1. Usporedite naredbu *imagesc* s naredbama *image* i *imshow*. Što je potrebno napraviti da bi rezultati naredbi bili jednaki?
2. Kako biste to napravili matematički (napišite jednadžbe)?
3. Učitajte sliku *uskoci1.png* (pretvorite je 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.
4. Ako unarnu operaciju označimo s U_x ($x = 1, 2, 3$), a normiranje svjetline s N napišite izraz za prijenosnu funkciju sustava zadanog slikom 3.1. u obliku kompozicije funkcija $H_x = N \circ U_x$.
5. Prikažite prijenosne funkcije sustava za svaku odabranu unarnu operaciju.

3.2. Binarne operacije na slici

Kod binarnih operacija na ulazu imamo par slika nad kojima vršimo neku matematičku operaciju. Blok shema binarne operacije prikazana je na slici 3.2..



Slika 3.2.: Blok shema binarne operacije.

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. Posebno 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.

3.2.1. Primjer

```
>> [img1, map] = imread('medalja_kamenita_vrata.png');
>> [img2, map] = imread('medalja_dubrovnik.png');
>> whos
  Name      Size      Bytes  Class      Attributes

  img1     600x600    360000  uint8      % prva slika je 600x600
  img2     545x548    298660  uint8      % druga slika je 545x548
  map      0x0        0       double

>> img2=imresize(img2,[600 600],'bilinear');
>> size(img2)      % sada je i prva slika 600x600

ans =

    600    600

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

3.2.2. Zadatci

1. Iz USC-SIPI baze slika 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.

3.3. Digitalna angiografija

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.

3.3.1. Zadatci

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 prikazite rezultat. Što ste dobili?

3.4. 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 određivanje točnih vrijednosti gama faktora potrebna je kalibracija. Gama korekcija dakle odgovara matematičkoj funkciji potenciranja, za koju smo potenciju označili slovom gama (γ):

$$E_{iz} = E_{ul}^{\gamma}, \quad (3.1)$$

gdje E označava, napon, odnosno intenzitet prikazan na zaslonu.

Za računanje gama korekcije koristi se funkcija *imadjust()*.

3.4.1. 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
```

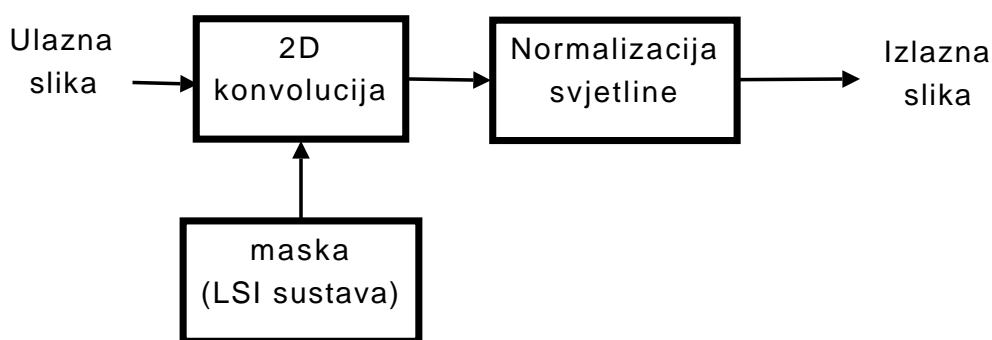
Ukoliko je slika u RGB prostoru boja, sva tri kanala možemo mijenjati istovremeno naredbom *imadjust()* gdje za posljednji argument unosimo vrijednost gama korekcije. Ukoliko želimo mijenjati gama korekciju svakog kanala zasebno za posljednji argument unosit ćemo vektor redak s tri komponente, od kojih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

3.4.2. Zadatci

1. Iz USC-SIPI baze slika odaberite nekoliko slika i isprobajte različite vrijednosti faktora gamma.
2. Učitajte sliku *psgamma.tif* koja sadrži testni uzorak za određivanje gama korekcije. Uzorak se sastoji od četiri pruge u bijeloj, plavoj, zelenoj i crvenoj boji. Svaka od pruga ima popunjeni pravokutnih u sredini te uzorak sa strane. Sliku je potrebno prikazati na crnom ekranu u sredini uz normalno osvjetljenje. Tada se podešava gama korekcija za svaku od boja na taj način da s veće udaljenosti nije moguće razlikovati popunjeni pravokutnih od odgovarajuće pozadine. Jednom kada su određene vrijednosti gama korekcije za crvenu, zelenu i plavu boju četvrti bijeli pravokutnik trebao bi biti stopljen s pozadinom. Pokušajte odrediti iznose korekcije za vaš ekran.
3. Je li gama korekcija unarna ili binarna operacija na slici?

3.5. Linearna konvolucija

Svi dosadašnji primjeri predstavljaju bezmemorijske sustave jer izlazi sustava ovise samo o trenutnom stanju (trenutnom pikselu, a ne njegovoj okolini). Slijedeći primjer će prikazati kako možemo sliku obraditi memorijskim sustavom. U digitalnoj obradbi slike zanimljiv nam je pojam linearnog prostorno invarijantnog sustava². Ovdje vremenska invarijantnost sustava nema osobitu važnost ako sliku kao signal opisujemo samo kao vrijednosti intenziteta za određene prostorne koordinate, odnosno ako vrijeme nije varijabla signala. Odziv LSI sustava, jednako kao i odziv LTI sustava, možemo odrediti linearnom konvolucijom.



Slika 3.3.: Blok shema linearne konvolucije.

Maska nam predstavlja impulsni odziv FIR 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 retcima matrice). Primjer računanja konvolucije dan je u nastavku.

3.5.1. Primjer

```
>> [img,map]=imread('medalja_kamenita_vrata.png');
>> msk=[1 0 -1      % definiramo masku
2 0 -2
1 0 -1]/4;
>> imgC=conv2(img,msk); % računamo konvoluciju
>> imshow(imgC) % prikazujemo rezultat
```

Konvolucija slike i maske računa se pozivom funkcije *conv2()*. MATLAB pri računanju 2D konvolucije pretpostavlja da su vrijednosti točaka izvan slike jednake nuli, tj. slika je proširena s nulama. Ako je slika dimenzija $M_x \times N_x$ i maska dimenzija $M_h \times N_h$ konvolucija je dimenzija $(M_x + M_h - 1) \times (N_x + N_h - 1)$.

² eng. LSI - Linear Space Invariant

3.5.2. Primjer

```
>> [img,map]=imread('medalja_dubrovnik.png'); % učitavamo sliku
>> msk=[1 0 -1 % definiramo masku
1 0 -1
1 0 -1]/3;
>> imgC1=conv2(img,msk); % računamo punu konvoluciju
>> imgC2=conv2(img,msk,'same'); % računamo centralni dio
% konvolucije tako da je izlaz istih
% dimenzija kao ulaz
>> imgC1=conv2(img,msk,'valid'); % računamo samo ispravni
% centralni dio konvolucije
```

Primijetite da, kako nemamo informacija o vrijednosti točaka izvan slike koje su potrebne za računanje linearne konvolucije, ovakav postupak unosi pogreške u rubovima slike. Dakle, pouzdan je samo onaj dio rezultata u kojem računamo središnji dio konvolucije veličine $(M_x - M_h + 1) \times (N_x - N_h + 1)$

3.5.3. Zadatci

1. Odaberite nekoliko slika i izračunajte linearne konvolucije s maskama:

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

2. Prikažite i komentirajte rezultate.
3. Odredite masku za usrednjavanje dimenzije 4×4 i za jednu od odabranih slika iz prethodnog zadatka izračunajte linearnu konvoluciju.
4. Koju ste masku odabrali?
5. Je li za odabranu masku bila potrebna normalizacija svjetline? Zašto?
6. Je li linearna konvolucija unarna ili binarna operacija na slici?

Poglavlje 4.

Vježba 3 - Otipkavanje i kvantizacija

4.1. Kvantizacija

Matematički, kvantizaciju možemo definirati kao preslikavanje kodomene signala iz prostora \mathbb{R} u prostor \mathbb{N} .

Kvantizacija je proces u kojem se signal mijenja na način da se vrijednosti signala "zaokružuju" na točno određene (diskretne) vrijednosti (razine kvantizacije). Razlika između dvije susjedne razine kvantizacije naziva se korakom kvantizacije ili kvantom¹. Ideja kvantizacije je ukratko predstavljena slikom 4.1.. Veza kvanta i broja bitova potrebnih za kvantizaciju dana je s:

$$q = \frac{A}{2^B - 1}, \quad (4.1)$$

gdje je q kvant, a B broj bitova, a A raspon signala.



Slika 4.1.: Kontinuirani signal prije i nakon kvantizacije.

4.1.1. Zadatci

1. Provjerite što radi funkcija `quant()`. Kako iskoristiti ovu naredbu da jednoliko kvantizira sliku u N razina?

¹od lat. quant - koliko

2. Napišite funkciju za kvantizaciju slike koja cijeli dinamički opseg slike jednoliko kvantizira u N razina, gdje je N zadan (ulazni parametar funkcije). Tu funkciju koristite u ostatku vježbe.
3. Odaberite nekoliko različitih crno-bijelih fotografija (npr. *salona.png* ili *knjiga_ssa.png*) ili RGB slika koje prethodno pretvorite u crno-bijele naredbom *rgb2gray()*. Istovremeno prikazite originalnu sliku te tu istu sliku kvantiziranu na različiti broj bitova (1-8).
4. Kod kojeg broja bitova se počinje primjećivati razlika?
5. Što se dobiva kvantizacijom slike na 1 bit?
6. Što je kvantizacijski šum? Koliku maksimalnu vrijednost može poprimiti za pojedine primjere?

4.2. Otipkavanje

Matematički, otipkavanje možemo definirati kao preslikavanje domene signala iz prostora \mathbb{R} u prostor \mathbb{N} .

Otipkavanje ili uzorkovanje signala je proces u kojem se od kontinuiranog signala dobiva diskretni signal, na način da se uzimaju vrijednosti (uzorci) originalnog signala samo na pojedinim mjestima (ili određenim vremenskim razmacima). Ukoliko su uzorci uzimani u pravilnim razmacima otipkavanje se naziva pravilnim ili homogenim. Prikaz uzorkovanja slike dan je na slici 4.2.. Odabirući frekvenciju otipkavanja (definiramo udaljenost između uzoraka koje želimo pohraniti) u okomitom i vodoravnom smjeru definirali smo količinu informacija koju ćemo pohraniti. Uzorkovana informacija predstavlja jednu točku digitalizirane slike.

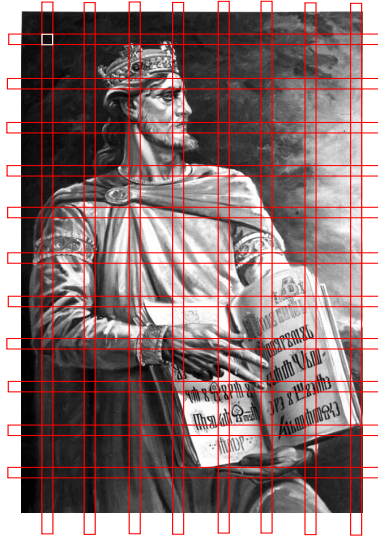
Kako je u digitalnom računalu nemoguće pohraniti kontinuirane signale radit ćemo s gusto otipkanim signalima, te njih otipkavati nekom manjom frekvencijom da bismo demonstrirali efekte otipkavanja. Ovaj proces naziva se podotipkavanje² jer uzima podskup od postojećih uzoraka signala. Ukoliko želimo podotipkati sliku to jednostavno radimo uzimajući svako N -tu piknjicu. U primjeru je dan jednostavni dio koda kojim možemo izvršiti decimaciju³.

4.2.1. Primjer

```
>> [img,map]=imread('klis2.png');
>> klisM=img(1:10:end,1:10:end,:);
```

²eng. subsampling

³za $N = 10$ ovaj proces se naziva i desetkovanje odnosno decimacija, od lat. decimus - desetina; primijetite da u engleskoj literaturi ovaj izraz (*decimation*) nema isto značenje kao i u hrvatskom jer označava filtriranje niskopropusnim filtrom te podotipkavanje za bilo koji faktor N .



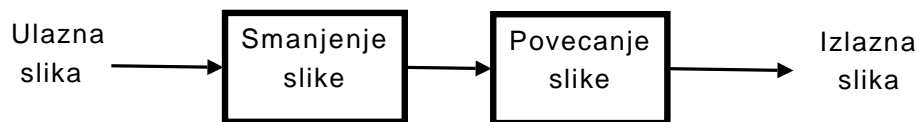
Slika 4.2.: Uzorkovanjem uzimamo samo onaj dio slike koji se nalazi na preklapanju vodoravne i okomite pruge (označen kvadratićem). Otipkavanje demonstriramo na slici "Kralj Tomislav" - Kristiana Krekovića.

4.2.2. Zadatci

1. Podotipkajte sliku *asinh1.tiff* (vidi dodatke za vježbe na stranici predmeta) s faktorom 1 do 10.
2. Prikažite i komentirajte rezultate. Što se dogodilo? Zašto?

4.3. Pikselizacija

U ovom djelu vježbe demonstriran je takozvani 'efekt šahovske ploče' koji nastaje uslijed smanjivanja rezolucije. Ovaj efekt se ponekad koristi za skrivanje identiteta osobe na slici ili cenzuriranja nekih informacija i tada se naziva pikselizacija. Vrši se na način prikazan shemom na slici 4.3..



Slika 4.3.: Blok shema eksperimenta

Smanjivanje slike za odabrani faktor te povećavanje za isti faktor ponavljanjem točaka ima efekt smanjivanja rezolucije slike. Naredbu koju možemo koristiti za smanjivanje ili povećanje slike je *imresize()*. Primjer korištenja naredbe za smanjenje, a zatim povećanje slike za 5 puta dan je u nastavku:

4.3.1. Primjer

```
>> [img,map]=imread('klis2.png');  
>> klisM=imresize(img,1/5,'nearest',0);  
>> klisV=imresize(klisM,5,'nearest',0);
```

Ukoliko želite izvršiti cenzuru objekta koji zauzima površinu od 60×25 piksela, a počinje s točkom (20, 32), odabrati ga možete kao što je prikazano u sljedećem primjeru. Pikselizaciju vršite kao i ranije.

4.3.2. Primjer

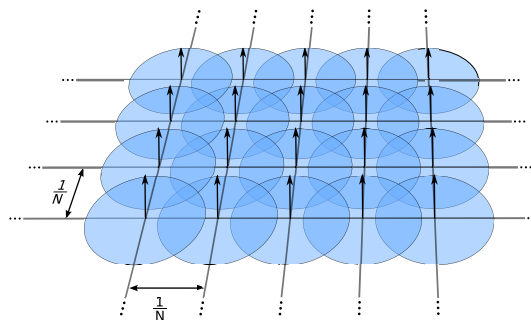
```
>> dio=img(20:79,32:56);
```

4.3.3. Zadatci

1. Odaberite nekoliko različitih fotografija. Na njima odaberite objekt koji želite cenzurirati. Koristeći opisan postupak, cenzurirajte objekt, na način da ga prekrijete njegovom pikseliziranom verzijom.
2. Čemu služi 'nearest' parametar u naredbi *imresize*?

4.4. Alias-efekt

Prema Shannonovom teoremu otipkavanja, frekvencija otipkavanja mora biti dvostruko veća od najveće frekvencije koja se pojavljuje u signalu, da bi se signal mogao pravilno rekonstruirati. Ukoliko frekvencija otipkavanja nije dovoljno velika, događa se da dva signala različitih frekvencija više ne razlikujemo (jedan drugome su alias⁴). Tu pojavu nazivamo preslikavanje spektra ili alias-efekt. Taj efekt je grafički prikazan na slici 4.4., gdje je prikazana frekvencijska domena signala i preklapanje dijela spektra zbog nedovoljno velike frekvencije otipkavanja. Blok shema ovog eksperimenta prikazana je na slici 4.5..



Slika 4.4.: Ilustracija preslikavanja spektra u frekvencijskoj domeni.

⁴lat. alias - (neki) drugi (neki sljedeći), drugačiji, inače

Preslikavanje spektra dovodi do interferencije dvaju alias signala, što rezultira fenomenom zvanim moarški⁵ efekt. O ovom fenomenu bit će više riječi u sljedećem poglavlju.



Slika 4.5.: Blok shema eksperimenta

Da bi se izbjeglo preslikavanje spektra i interferencija signala ukoliko želimo smanjiti rezoluciju slike, ne ćemo samo otipkati sliku već ćemo prije toga odrezati više frekvencije u signalu (one koje bi se mogle preslikati). To radimo na način sliku filtriramo niskopropusnim filtrom. Najjednostavniji primjer niskopropusnog filtra je operacija prostornog usrednjavanja.

4.4.1. Primjer

```
>> [img,map]=imread('uzorak.tif'); % učitavamo sliku
>> H=fspecial('average',3); % računamo odziv traženog filtra
>> imgL=filter2(H,img); % primjenjujemo filter
```

Može nam se učiniti da smo nepotrebno izgubili dio detalja usrednjavajući sliku, no detalje bismo ionako izgubili podotipkavanjem signala. Usrednjavajući sliku, zamutili smo sve piksele na takav način da se dio informacije susjednih piksela preslikao na onaj piksel u sredini filtra za usrednjavanje. Stoga ovaj postupak možemo i promatrati kao spremanje maksimalne količine informacije u što manji broj piksela. Nažalost, ovaj postupak nije reverzibilan.

4.4.2. Zadatci

1. Zamutite, te podotipkajte sliku *asinh1.tiff* s faktorom 1 do 10. Usporedite rezultate s rezultatima zadatka 4.2.2..
2. Podotipkajte slike *testpat1.tif* i *testpat2.tif* za četiri puta.
3. Što se dogodilo sa višim, a što sa nižim frekvencijama u slici?
4. Ponovite eksperiment uz usrednjavanje slike. Usporedite rezultate.
5. Kako rezultati ovise o veličini filtra za usrednjavanje?
6. Učitajte sliku *uzorak.tif*. Podotipkajte je za tri puta samo po x osi.
7. Učitajte sliku *uzorak.tif*. Podotipkajte je za dva puta samo po y osi.

⁵od franc. moiré - vrsta tkanine

8. Što se dogodilo u prvom, a što u drugom slučaju? Zašto?
9. Smanjite rezoluciju slika *testpat1.tif* i *uzorak.tif* za četiri puta koristeći naredbu *imresize()*. Razlikuju li se ovi rezultati od prethodnih? Zašto?
10. Možemo li ovom naredbom izbjeći postupak prefiltriranja slike? Zašto?
11. Demonstrirajte alias-efekt na nekoj slici koja sadrži prirodnu scenu.

4.5. Moarški efekt

Moarški⁶ efekt je otkriven mnogo prije nastanka digitalnih računala, kao fenomen koji nastaje prilikom preklapanja različitih uzoraka (npr. tkanine). Moarški efekt u osnovi nastaje interferencijom različitih uzoraka (ili valova). Ako signale promatramo kao valove, možemo reći da moarški efekt nastaje i prilikom preklapanja spektra i pojavljivanja aliasa nekog signala. Jednostavan prikaz Moarškog efekta možete vidjeti na animaciji na stranici

<http://ipg.zesoi.fer.hr/~hrvoje/>

U prikazanom slučaju čovjek doživljava brojke i slova koja se ne nalaze na slici zbog vlastite percepcije, a koji je u Berlinskoj školi Gestalt⁷ psihologije opisan kao zakon neprekinutosti⁸. Ova vrsta optičke iluzije je znana i pod nazivom materijalizacija⁹. Bitno je primijetiti da se isti efekt (doživljavanja objekta koji ne postoji) može postići i s računalom, kao alias-efekt.

4.5.1. Zadatci

1. Odaberite sliku *AB2.tiff* i podotipkajte je s faktorom 39 po y-osi. Prikažite originalnu i otipkanu verziju slike. Komentirajte rezultate.
2. Što se dogodi ukoliko je podotipkate s nekim drugim faktorom po y-osi? Pokušajte s nekim slučajno odabranim faktorom, te faktorima 37 i 40.

Još jedan primjer moarškog efekta do kojeg dolazi zbog aliasa možete vidjeti u slijedećem zadatku.

⁶eng. *moiré effect*, dolazi od francuske riječi *moiré* koja označava vrstu svile s valovitim uzorcima, nastalu tkanjem uglačanih niti svile. Etimološki korijen riječi *moiré* može se pratiti do arapske riječi *mukhayyar*, koja je označavala vrstu kašmira (u doslovnom značenju *odabrani*). Ipak, sumnja se da je arapska riječ *mukhayyar* u vezi s latinskom riječi *marmoreus* (što znači mramoran, umjetnički). Iz tog razloga predloženo je da se ovaj efekt naziva i mramornim efektom.

⁷njem. die Gestalt - oblik, građa

⁸njem. fortgesetzt

⁹eng. reification

4.5.2. Zadatci

1. Odaberite sliku *hsokol.png* (ili *rolete.png*) i podotipkajte je s faktorom 5. Prikažite obje slike i objasnite rezultat.

Do moarškog efekta može doći i prilikom skeniranja slika. Zbog preklapanja frekvencijskog spektra dolazi do aliasa i javljaju se interferencijski uzorci. Oni su najbolje vidljivi na nekim uzorcima kao što je je bilo demonstrirano u zadatku 4.2.2.. Kako ne možemo demonstrirati skeniranje, ponovno ćemo ga simulirati podotipkavanjem slika velike rezolucije. Slike *testpat_circ.tiff* i *asinh2.tiff* su skenirane sa 300dpi¹⁰. Simulirajte što bi se dogodilo da su skenirane nekom manjom rezolucijom (200, 100 ili 50 dpi).

4.5.3. Zadatci

1. Demonstrirajte moarški (alias-) efekt na uzorcima *testpat_circ.tiff* i *asinh2.tiff*, za nekoliko različitih faktora podotipkavanja.

¹⁰eng. dots per inch

Poglavlje 5.

Vježba 4 - Frekvencijske transformacije

5.1. Diskretna Fourierova transformacija

Jednodimenzionalna diskretna Fourierova transformacija (DFT) dana je parom jednadžbom 5.1, a njen inverz jednadžbom 5.2.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \quad k = 0, \dots, N-1 \quad (5.1)$$

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N} kn} \quad n = 0, \dots, N-1 \quad (5.2)$$

Jednadžbe 5.1 i 5.2 možemo zapisati i u matricnom obliku:

$$X = \mathbf{W}x \quad (5.3)$$

$$x = \mathbf{W}^{-1}X \quad (5.4)$$

ako definiramo matricu \mathbf{F} kao:

$$\mathbf{W} = \begin{bmatrix} \omega_N^{0 \cdot 0} & \omega_N^{0 \cdot 1} & \dots & \omega_N^{0 \cdot (N-1)} \\ \omega_N^{1 \cdot 0} & \omega_N^{1 \cdot 1} & \dots & \omega_N^{1 \cdot (N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_N^{(N-1) \cdot 0} & \omega_N^{(N-1) \cdot 1} & \dots & \omega_N^{(N-1) \cdot (N-1)} \end{bmatrix} \quad (5.5)$$

gdje je $\omega_N = e^{-2\pi i/N}$.

Dvodimenzionalna DFT je dana izrazom:

$$X(k, l) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} x(m, n) e^{-\frac{2\pi i}{N} kn} e^{-\frac{2\pi i}{M} lm}, \quad k \in [0, N-1], l \in [0, M-1] \quad (5.6)$$

Lako se pokaže da je to separabilna transformacija:

$$X(k, l) = \sum_{n=0}^{N-1} e^{-\frac{2\pi i}{N} kn} \sum_{m=0}^{M-1} e^{-\frac{2\pi i}{M} km} x(m, n) \quad (5.7)$$

iz čega je vidljivo da se može zapisati kao dvije jednodimenzionalne DFT. Kako se DFT najčešće računa korištenjem algoritma za brzu Fourierovu transformaciju (FFT), tako i 2D DFT obično računamo prvo 1D FFT po redcima te 1D FFT po stupcima. Stoga je složenost 2D FFT algoritma $O(N^2 \log_2 N)$. 2D FFT računamo korištenjem funkcije `fft2` kao što je navedeno u sljedećem primjeru.

5.1.1. Primjer

```
>> img = imread('klis2.png');           % učitamo sliku
>> imgFT=fft2(img);                     % primijenimo fft2 funkciju
>> imgFT=imgFT/prod(size(imgFT));      % da bi dobili stvarne
                                         % koeficijente DFT-a moramo rezultat skalirati
>> imgFT=fftshift(imgFT);              % centriramo spektar
>> whos
```

Name	Size	Bytes	Class	Attributes
img	500x419	209500	uint8	
imgFT	500x419	3352000	double	complex

Funkcije `fft()` i `fft2()` računaju spektar čiji centar nije u sredini dobivene matrice, već se nalazi u gornjem lijevom kutu. To je napravljeno zbog konvencije da se ishodište nalazi u gornjem lijevom kutu, što je već ranije objašnjeno u Poglavlju 2. i prikazano na slici 11.1.. Ukoliko rezultat želimo prikazati tako da je ishodište smješteno u središtu slike, potrebno je primijeniti funkciju `fftshift()`. Ukoliko nakon pomaka ishodišta još i skaliramo dobivenu diskretnu Fourierovu transformaciju s brojem elemenata ulazne matrice (dijeljenje spektra s `prod(size(imgFT))`), dobit ćemo izgled spektra kako ga uobičajeno prikazujemo u domeni transformacije. Važno je primijetiti da funkcije `ifft()` i `ifft2()` koje računaju inverznu transformaciju očekuju spektar koji nije pomaknut i skaliran!

Spektar koji smo dobili u prethodnom primjeru i koji smo zapisali u varijabli `imgFT` je kompleksan. Spektar obično prikazujemo tako da odvojeno prikažemo amplitudu i fazu definirane kao:

$$A_k = |X_k| = \sqrt{\mathbf{Re}(X_k)^2 + \mathbf{Im}(X_k)^2} \quad (5.8)$$

$$\phi_k = \arg(X_k) = \arctg^*(\mathbf{Im}(X_k), \mathbf{Re}(X_k)) \quad (5.9)$$

gdje je ϕ_k argument kompleksne varijable X_k koji se računa kao arkus tangens gdje se predznaci realnog i imaginarnog dijela kompleksne varijable koriste da bi se točno odredio kut vektora (kvadrant u kojem se vektor nalazi). Stoga koristimo zapis \arctg^* kao kraticu za:

$$\arctg^*(y, x) = \begin{cases} \arctg(\frac{y}{x}) & x > 0 \\ \pi + \arctg(\frac{y}{x}) & y \geq 0, x < 0 \\ -\pi + \arctg(\frac{y}{x}) & y < 0, x < 0 \\ \frac{\pi}{2} & y > 0, x = 0 \\ -\frac{\pi}{2} & y < 0, x = 0 \\ 0 & y = 0, x = 0 \end{cases}$$

Budući da su razlike u iznosu koeficijenata velike te dosežu i do nekoliko redova veličine, amplitudu uobičajeno prikazujemo u decibelima (odnosno logaritmiramo je radi kompresije dinamičkog opsega). Sve operacije vršimo s tek nekoliko linija koda:

5.1.2. Primjer

```
>> amplituda=20*log10(abs(imgFT)); % računamo amplitudu u dB
>> faza=angle(imgFT);           % računamo fazu
>> figure, imagesc(amplituda)
>> figure, imagesc(faza)
```

PRIMJEDBA: Ukoliko transformirate neke od slika kojoj dimenzije nisu potencija broja dva koristi se običan algoritam računanja diskretne Fourire-ove transformacije. Ako su dimenzije potencije broja dva koristi se izuzetno brzi *radix-2* algoritam. Ponekad se slika nadopuni s nulama dok dimenzije ne dosegnu potencije broja dva da bi se mogla primijeniti *radix-2* metoda.

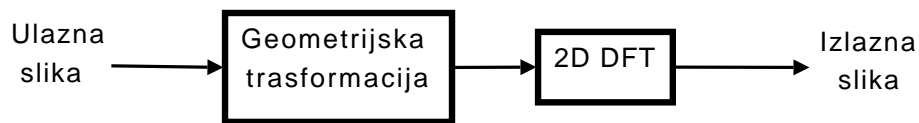
5.1.3. Zadatci

1. Odaberite jednu sliku koja sadrži prirodnu scenu (npr *kllis2.png*) i jednu sliku teksture (npr. *misal_1483.png* ili *uzorak.tif*). Prikažite amplitudu i fazu spektra. U kojem se dijelu spektra nalazi većina energije?

5.2. DFT i geometrijske transformacije slike

U ovom dijelu vježbe ispitat ćemo kako se mijenja spektar slike, ako sliku deformiramo nekim osnovnim geometrijskim transformacijama poput skaliranja, rotacije ili translacije. Blok shema eksperimenta prikazana je na slici 5.1..

Veličinu slike mijenjamo naredbom *imresize()*. Ako sliku želimo zakrenuti za kut ϕ tada koristimo naredbu *imrotate()*. Translaciju slike ćemo izvesti na način da definiramo dvije velike matrice popunjene nulama u kojima sliku postavljamo na različita mjesta. Na taj način smo efektivno translatali sliku unutar matrice za pomak (dx, dy) , gdje dx i dy definiraju pomak



Slika 5.1.: Blok shema eksperimenta

(odnosno razmak) po x i y osi između centara dviju slika. Primjer korištenja navedenih naredbi dan je u nastavku.

5.2.1. Primjer

```

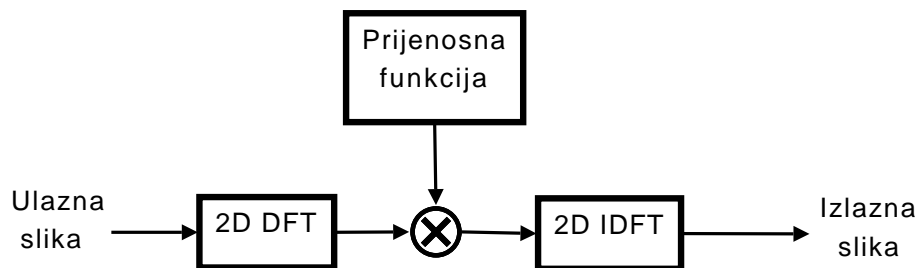
>> [img,map]=imread('misal_1483.png');          % učitamo sliku
>> imgV=imresize(img,2);                        % povećamo sliku dva puta
>> imgM=imresize(img,.25);                    % smanjujemo sliku četiri puta
>> imgR=imrotate(img,66,'crop');              % rotiramo sliku oko
                                                % centra za kut 66°
>> imgV1=uint8(zeros(1024));                  % stvaramo prvu veliku sliku
                                                % popunjenu nulama
>> imgV1(50:50+536,72:72+357)=img;           % od koordinata (50,72)
                                                % stavljamo matricu img (veličine 537x358)
>> imgV2=uint8(zeros(1024));
>> imgV2(460:460+536,830:830+357)=img;
  
```

5.2.2. Zadaci

1. Odaberite neku sliku. Izračunajte i prikažite amplitudne spektre originalne i skalirane slike. Po čemu se razlikuju?
2. Odaberite neku sliku. Izračunajte i prikažite amplitudne spektre originalne i rotirane slike. Po čemu se razlikuju?
3. Odaberite jednu manju sliku. Izračunajte i prikažite amplitudne i fazne spektre originalne i translirane slike. Po čemu se razlikuju?
4. Možete li iz faznog spektra zaključiti gdje se nalazi slika?

5.3. Filtriranje slike

Pod filtriranjem slike obično podrazumijevamo propuštanje slike kroz neki LSI sustav. Takvu operaciju možemo računati preko 2D konvolucije (kako je prikazano u poglavlju 3.5.) ili u domeni transformacije. Filtriranje u domeni transformacije izvodi se množenjem Fourierove transformacije slike s Fourierovom transformacijom impulsnog odziva filtra.



Slika 5.2.: Filtriranje slike u frekvencijskoj domeni

Prijenosna funkcija LSI sustava je transformacija impulsnog odziva sustava. Za konstrukciju prijenosne funkcije idealnog filtra koristite naredbu *frqflt()*. Filtriranje u frekvencijskoj domeni je jednostavno množenje. Pri tome pripazite kada je potrebno koristiti naredbu *fftshift()*:

5.3.1. Primjer

```

>> help frqflt
FRQFLT - Konstrukcija idealnog filtra u frekvencijskoj domeni
poziv funkcije: filt = frqflt(dim, x0, y0, r)
filt = frqflt(dim, x0, y0, r)
dim - dimenzija filtra
x0, y0 - ishodište filtra
r - koliko okolnih frekvencija čuvamo (radius)
Izlaz funkcije je slika filt koja na mjestu frekvencija koje
čuvamo ima 1, a na ostalima 0. Složenije filtre možemo lako
konstruirati logičkim operacijama (npr. not(filt1) ili
filt1&filt2 i sl.)

>> filter=frqflt([512 512], 512/2, 512/2, 256*0.5);
      % konstruiramo NP filter s pojasom propuštanja do
      % 0.5, ako je 1 na polovici frekvencije otipkavanja
>> imagesc(filter); % crtamo odziv filtra
>> [img,map]=imread('testpat1.tif'); % učitamo sliku
>> imgFT=fft2(img); % odredimo DFT
>> imgNP=imgFT.*fftshift(filter); % filtriramo sliku, centar
      % spektra filtra moramo premjestiti
      % koristeći fftshift
>> imgNP=ifft2(imgNP); % odredimo inverznu DFT
  
```

Primijetite da je rezultat filtracije kompleksna matrica. Stoga je potrebno primijeniti funkciju *real()* (ili *abs()*) na rezultat dobiven nakon inverzne transformacije. Ta operacija je potrebna jer su numeričke pogreške

pri određivanju odziva filtra te pri računanju inverzne transformacije uzrokovala pojavljivanje kompleksnih vrijednosti u rekonstruiranoj slici. Prilikom računanja diskretne Fourierove transformacije pretpostavljamo periodičnost signala, stoga prikazani postupak množenja u DFT domeni odgovara upravo kružnoj (cirkularnoj) konvoluciji koju u 1D definiramo na slijedeći način:

$$x_N[n] \circledast h[n] = \sum_{m=-\infty}^{\infty} h[m] \cdot x_N[n - m] \quad (5.10)$$

$$= \sum_{m=-\infty}^{\infty} \left(h[m] \cdot \sum_{k=-\infty}^{\infty} x[n - m - kN] \right) \quad (5.11)$$

primijetimo kako je definicija kružne konvolucije jednaka jednoj periodične konvolucije. U 2D ova definicija poprima oblik:

$$y[m, n] = \sum_{i=0}^{M_x-1} \sum_{j=0}^{N_x-1} x[i, j] h[(m - i) \% M_x, (n - j) \% N_x] \quad (5.12)$$

gdje $\%$ označava operator modulo dijeljenja, a M_x i N_x su dimenzije slike $x[i, j]$ uz $M_x \geq M_h$ i $N_x \geq N_h$. Ako želimo pomoću diskretne Fourierove transformacije izračunati linearnu konvoluciju potrebno je proširiti signale $x[i, j]$ i $h[i, j]$ s nulama tako da dimenzije oba signala budu barem $M_x + M_h - 1 \times N_x + N_h - 1$. Za tako proširene signale množenje u domeni diskretne Fourierove transformacije odgovara linearnoj konvoluciji signala.

Uobičajeno se signal proširuje s nulama do prve veće potencije broja dva zbog ubrzanja postupka (korištenje FFT algoritma). Sada je nakon množenja signala u domeni transformacije rezultat inverzne transformacije dimenzija većih od rezultata linearne konvolucije (veći od $M_x + M_h - 1 \times N_x + N_h - 1$). Rezultat linearne konvolucije koji tražimo se nalazi unutar izračunate inverzne Fourierove transformacije te je samo potrebno odbaciti višak koji ne nosi informaciju. Točan položaj rezultata će ovisiti o načinu na koji smo signal $h[i, j]$ proširili s nulama.

5.3.2. Zadatci

1. Iz USC-SIPI baze slika odaberite jednu fotografiju. Ispitajte djelovanje svakog od tipova¹ filtara na sliku. Kakav je učinak za pojedini tip filtra? Filtriranje obavite izravnim množenjem u frekvencijskoj domeni.

5.4. Diskretna kosinusna transformacija

Diskretna Fourierova transformacija nije prikladna za razne primjene upravo jer je po svojoj naravi kompleksna. Umjesto kompleksne diskretne Fourierove

¹niskopropusni, pojasnopropusni, visokopropusni, te pojasna brana

transformacije najčešće se koristi realna diskretna kosinusna transformacija (DCT). Kako je DCT realna transformacija ne postoji fazni dio spektra, već samo amplitudni dio. Iako postoji više verzija DCT (DCT-I – DCT-VIII) kada govorimo o DCT obično podrazumijevamo DCT-II. U 1D prostoru ona je dana izrazom:

$$X[k] = \alpha(k) \sum_{n=0}^{N-1} x[n] \cos \frac{(2n+1)k\pi}{2N} \quad (5.13)$$

gdje je $\alpha(k)$ normalizacijski koeficijent koji poprima vrijednosti:

$$\alpha(k) = \begin{cases} \sqrt{\frac{1}{N}}, & k = 0 \\ \sqrt{\frac{2}{N}}, & 1 \leq k < N \end{cases} \quad (5.14)$$

Izraze 5.13 i 5.14 možemo zapisati jednim izrazom u obliku:

$$X[k] = \sqrt{\frac{2 - \delta[k]}{N}} \sum_{n=0}^{N-1} x[n] \cos \frac{(2n+1)k\pi}{2N} \quad (5.15)$$

Ako želimo zapisati DCT u 2D, tada koristimo izraz:

$$X_{k_1, k_2} = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x_{n_1, n_2} \cos \left[\frac{\pi}{N_1} \left(n_1 + \frac{1}{2} \right) k_1 \right] \cos \left[\frac{\pi}{N_2} \left(n_2 + \frac{1}{2} \right) k_2 \right] \quad (5.16)$$

Normalizirani oblik se dobije kao i u 1D slučaju, dodavajući normalizacijski koeficijent $\alpha(k_1, k_2)$.

5.4.1. Primjer

```
>> img=imread('klis2.png'); % učitavamo sliku
>> imgCT=dct2(img); % računamo DCT
>> img0=idct2(img); % računamo inverznu DCT
>> whos % više nemamo problema s kompleksnim vrijednostima
Name Size Bytes Class Attributes
img 500x419 209500 uint8
imgCT 500x419 1676000 double
img0 500x419 1676000 double
```

5.4.2. Zadatci

1. Odaberite jednu sliku koja sadrži prirodnu scenu i neku sliku s teksturom. Primijenite na njih diskretnu kosinusnu transformaciju. Prikažite rezultate.
2. Gdje se u spektru nalazi istosmjerna komponenta?

3. Kod koje od dvije prikazane transformacije (DFT i DCT) je više energije raspoređeno u niskofrekvencijski dio spektra? Objasnite zašto.

5.5. DCT i kompresija slike

Kao što smo primijetili u prethodnom poglavlju većina energije je sadržana u niskofrekvencijskim komponentama spektra. To uvijek vrijedi ukoliko je riječ o prirodnim slikama². Upravo to svojstvo DCT možemo iskoristiti za kompresiju slike koja se sastoji od DCT transformacije, restrikcije baze i kvantizacije koeficijenata. Transformacija ne vrši na cijeloj slici, već se slika rastavlja u blokove veličine 8×8 ili 16×16 . Tada se na svaki pojedini blok primjenjuje DCT te se vrši restrikcija baze i kvantizacija koeficijenata. Na primjer, osnovni JPEG algoritam dijeli sliku na blokove veličine 8×8 . Za svaki blok se tada računa diskretna kosinusna transformacija te se kvantiziraju koeficijenti. Kvantizirani koeficijenti se tada redom kodiraju Huffmanovim algoritmom. U MATLAB-u postoji nekoliko naredbi koje omogućuju rad na blokovima slike. To su *blkproc()*, *colfilt()* i *nlfilt()*. Mi ćemo koristiti naredbu *blkproc()*. Da odredimo DCT transformaciju na blokovima veličine 8×8 u MATLAB-u radimo sljedeće:

5.5.1. Primjer

```
>> [img,map]=imread('salona.png'); % učitavamo sliku
>> imgT=blkproc(img,[8 8],'dct2'); % računamo DCT u blokovima
                                     % veličine 8x8
>> img0=blkproc(imgT,[8 8],'idct2'); % računamo inverznu DCT u
                                     % blokovima veličine 8x8
```

Sada bi samo na svaki blok izračunate transformacije trebalo primijeniti restrikciju baze. Ako nam je matrica koja sadrži masku za restrikciju baze spremljena u varijabli M, restrikciju vršimo na sljedeći način:

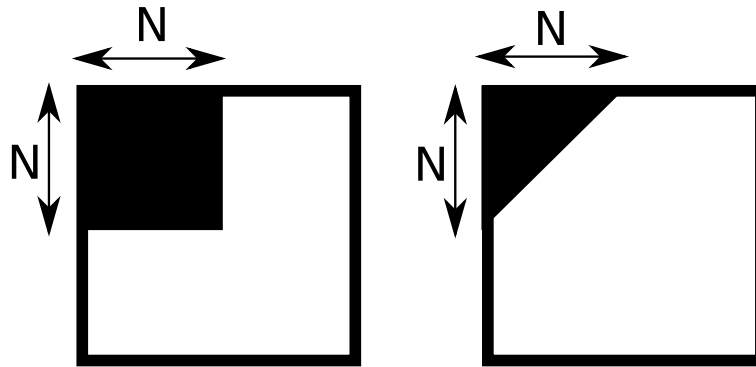
5.5.2. Primjer

```
>> M=ones(8); % M je maska za restrikciju
>> imgR=blkproc(imgT,[8 8],'x.*P1',M);
                                     % svaki blok množimo s maskom
```

5.5.3. Zadatci

1. Napišite funkciju koja vraća matricu 8×8 koju ćemo koristiti kao masku za restrikciju baze. Ulazni parametri funkcije neka budu oblik maske i veličina maske. Primjeri maski trokutastog i kvadratnog oblika su dani na slici 5.3., gdje tamni dio označava jedinice, a svijetli nule, dok N označava veličinu maske.

²prirodna slika - sadržaj slike je scena iz stvarnog svijeta, a ne umjetno generirana slika



Slika 5.3.: Primjer maski koje ćemo koristiti za restrikciju baze

2. Iz USC-SIPI baze slika odaberite jednu fotografiju. Uz veličinu bloka 8×8 odredite DCT te primijenite restrikciju baze. Prikažite originalnu i rekonstruiranu sliku.
3. Odredite srednje kvadratno odstupanje između originalne slike i slike rekonstruirane iz spektra kojemu je smanjena baza.
4. Isprobajte različite maske za restrikciju baze i usporedite rezultate (vizualno i računajući srednje kvadratno odstupanje od originalne slike). Što zaključujete?
5. Odredite srednje kvadratno odstupanje za slučaj kada uz restrikciju baze vršimo i kvantizaciju koeficijenata.

Poglavlje 6.

Vježba 5 - Poboljšanje slike

6.1. Histogram prvog reda

Za sive slike histogram prvog reda predstavlja relativnu frekvenciju pojave različitih nivoa sivog u slici. Histogram prvog reda računamo i pokazujemo naredbom *imhist()*.

6.1.1. Primjer

```
>> [img,map]=imread('detalj.png'); % učitavamo indeksiranu sliku
>> imhist(img,map)           % prikazujemo histogram uz
                             % odgovarajuću paletu boja
>> [img,map]=imread('salona.png'); % učitavamo sivu sliku
>> imhist(img)               % paleta sada nije potrebna
```

6.1.2. Zadatci

1. Odaberite nekoliko slika i prikažite njihove histograme prvog reda. Što se nalazi na kojoj osi?

6.2. Izjednačavanje histograma

Izjednačavanje histograma je operacija pri kojoj se histograma slike mijenja tako da broj točaka bude približno jednak za sve svjetline. Histogram slike izjednačava se pozivom funkcije *histeq()*:

6.2.1. Primjer

```
>> img=imread('uskoci1.png'); % učitamo sliku
>> img=rgb2gray(img,map);     % pretvaramo sliku u sivu
>> imgEQ=histeq(img);         % izjednačavamo histogram
>> figure, imshow(img)
>> figure, imshow(imgEQ)
```

Za više detalja o funkciji *histeq()* pogledajte ugrađenu pomoć.

6.2.2. Zadatci

1. Učitajte slike *uskoci1.png* i *salona.png*. Prikažite histograme prvog reda tih slike prije i nakon izjednačavanja histograma.
2. Vidite li više detalja prije ili nakon izjednačavanja?

6.3. Modeliranje histograma

Ako je F neka funkcija kumulativne distribucije i F^{-1} njezina inverzna distribucija te ako je U skup brojeva jednolike distribucije na intervalu $[0,1]$ (u ovom slučaju skup vrijednosti točaka slike), onda skup brojeva $F^{-1}(U)$ ima distribuciju F . Ovu činjenicu možemo koristiti za modeliranje histograma:

6.3.1. Primjer

```
>> [img,map]=imread('salona.png'); % učitamo sliku
>> img=ind2gray(img,map);          % pretvaramo sliku u sivu
>> imgEQ=histeq(img);              % izjednačavamo histogram
>> imgMEQ=imscale(img,[0 1]);      % sada slika imgMEQ ima
                                   % približno jednoliku distribuciju na [0 1]
>> imgMEQ=norminv(imgMEQ,0,10);    % slika sada ima normalnu
                                   % razdiobu s očekivanjem 0 i devijacijom 10
```

Za modeliranje histograma se može izravno koristiti i funkcija *histeq()*. Za detalje pogledajte ugrađenu pomoć.

6.3.2. Zadatci

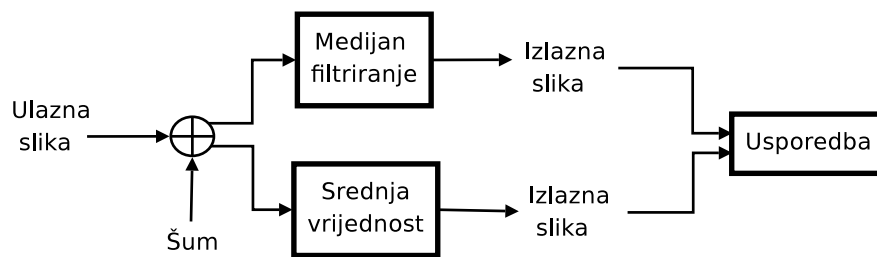
1. Odaberite neku sliku. Modelirajte joj histogram tako da dobijete željenu funkciju distribucije.
2. Učitajte sliku *auto.tif* te modelirajte histogram¹ tako da broj na registarskim tablicama auta sa slike postane čitljiv.

6.4. Usrednjavanje i median filtar

U ovom dijelu vježbe usporediti ćemo statistički median filtar i obični filtar za usrednjavanje u primjeni uklanjanja šuma. Blok shema eksperimenta prikazana je na slici.

Aditivni šum slici dodajemo pozivom funkcije *imnoise()*:

¹Pogledajte u kojem intervalu brojeva se nalaze vrijednosti točaka registarske tablice te odredite preslikavanje koje rasteže taj interval na veći. Opisani postupak se može promatrati ili kao modeliranje histograma ili kao konstrukcija nelinearnog preslikavanja koje poboljšava sliku.



Slika 6.1.: Blok shema eksperimenta

6.4.1. Primjer

```

>> [img,map]=imread('medalja_dubrovnik.png'); % učitamo sliku
>> imgGN=imnoise(img,'gaussian');           % dodajemo bijeli šum
>> imgSP=imnoise(img,'salt & pepper'); % dodajemo impulsni šum
  
```

Median filtar je statistički filtar koji za svaki ulazni blok zadane veličine na izlazu daje median tog bloka. Postoje i drugi statistički filtri koji računaju određenu statističku veličinu za pojedini ulazni blok. Median filtriranje se vrši funkcijom *medfilt2()*. Usrednjavanje (koje također možemo promatrati kao statistički filtar) ćemo realizirati konvolucijom.

6.4.2. Primjer

```

>> imgMF=medfilt2(imgSP,[5 5]); % filtriramo šum s median
                                % filtrom i blokom veličine 5x5
>> maska=ones(5); % kreiramo masku željenih dimenzija (5x5)
>> maska=maska/sum2(maska); % usrednjavamo masku
>> maska % filter je određen s maskom u kojoj
                                % su svi elementi jednaki
  
```

maska =

```

0.0400    0.0400    0.0400    0.0400    0.0400
0.0400    0.0400    0.0400    0.0400    0.0400
0.0400    0.0400    0.0400    0.0400    0.0400
0.0400    0.0400    0.0400    0.0400    0.0400
0.0400    0.0400    0.0400    0.0400    0.0400
  
```

```

>> imgSF=conv2(imgSP,maska,'same');
  
```

6.4.3. Zadatci

1. Odaberite sliku te joj dodajte Gaussov i impulsni šum².

²eng. salt'n'pepper noise

2. Usporedite učinak oba filtra na uklanjanje Gaussovog i na uklanjanje impulsnog šuma. Komentirajte rezultate. Koji filter je primjereniji za uklanjanje pojedinih vrsta šuma?

6.5. Uklanjanje neoštine

Nekada se za uklanjanje neoštine koristio postupak u kojem se od dobivene slike oduzima zamućena slika³, odnosno neoština se uklanja prema izrazu

$$I_{hb}(x, y) = A \cdot I(x, y) - I_{nf}(x, y), \quad (6.1)$$

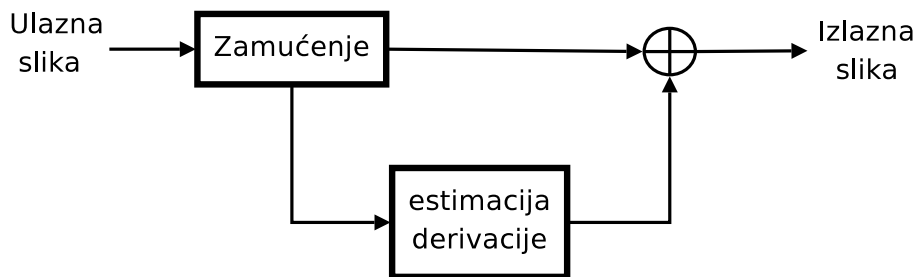
gdje je A pozitivan broj veći od jedan, a I_{nf} označava zamućenu originalnu sliku, odnosno I provučenu kroz niskopropusni filter. Uporaba ove metode za prividno pojačavanje oštine slika je prvi put zabilježena u Njemačkoj 30-ih godina 20-og stoljeća. Ovaj postupak možemo promatrati i obratno, tj. umjesto da slici oduzimamo zamućenu sliku, da joj dodamo sliku s pojačanim rubovima. Ako se slika I sastoji od niskih i visokih frekvencija ($I = I_{nf} + I_{vf}$) tada prethodnu jednadžbu možemo raspisati u:

$$I_{hb}(x, y) = A \cdot I(x, y) - I_{nf}(x, y) = \quad (6.2)$$

$$= A \cdot I_{vf}(x, y) + A \cdot I_{nf}(x, y) - I_{nf}(x, y) = \quad (6.3)$$

$$= (A - 1) \cdot I_{nf}(x, y) + A \cdot I_{vf}(x, y) \quad (6.4)$$

iz čega je vidljivo kako možemo povećati oštrinu zamućene slike ako joj dodamo visoke frekvencije. Ovaj postupak se naziva i *high-boost* filtriranje, te smo zbog toga izoštreanu sliku (s istaknutim detaljima) označili kao $I_{hb}(x, y)$. Najveći problem ove metode je kako doznati visoke frekvencije slike. U ovoj vježbi to ćemo učiniti dodavanjem estimirane derivacije da bismo dobili sliku s istaknutim visokofrekvencijskim komponentama. Blok shema osnovnog eksperimenta prikazana je na slici 6.2. gdje ulaznu sliku zamućujemo te na njoj estimiramo gradijent i formiramo izlaznu sliku koju možemo promatrati kao rekonstrukciju ulazne slike.



Slika 6.2.: Blok shema eksperimenta

³Ovaj postupak je poznat pod nazivom *unsharp masking*

Slika se zamućuje konvolucijom s maskom za usrednjavanje odgovarajuće veličine, npr. 3×3 , da bismo dobili zamućenu sliku.

6.5.1. Primjer

```
>> [img,map]=imread('blood1.tif'); % učitavamo sliku
>> maska=ones(3);
>> maska=maska/sum(maska(:)); % odziv sustava za usrednjavanje
>> imgZ=conv2(img,maska,'same'); % usrednjavamo sliku
```

Derivaciju u točki možemo procijeniti izrazom:

$$\Delta(x) = \frac{f(x+h) - f(x)}{h}, \quad (6.5)$$

ako govorimo o 1D slučaju. U 2D slučaju derivacija će imati svoj smjer i iznos, a procjenjujemo je nekim operatorom. Prvu derivaciju, odnosno gradijent, procijeniti ćemo npr. Soebelovim operatorom, a drugu derivaciju Laplaceovim operatorom. Pri tome koristimo funkciju *fspecial()* za kreiranje maske koja aproksimira Laplaceov operator:

6.5.2. Primjer

```
>> lap=fspecial('laplacian',0.2) % maska koja aproksimira
    % Laplaceov operator
ans =
    0.1667    0.6667    0.1667
    0.6667   -3.3333    0.6667
    0.1667    0.6667    0.1667
```

6.5.3. Zadatci

1. Odaberite jednu sliku i provedite postupak prikazan blok shemom na slici 6.2.. Prikažite razliku između slika. Koliko ste zadovoljni izoštravanjem (rekonstrukcijom)?
2. Učitajte sliku *medalja_dubrovnik.png* te na njoj uklonite neoštrinu koristeći oduzimajući joj njenu zamućenu verziju (prema jednadžbi 6.2). Što smo dobili ovim postupkom?
3. Učitajte sliku *medalja_dubrovnik.png* te na njoj uklonite neoštrinu dodavanjem procjene prve derivacije Soebelovim operatorom (prema jednadžbi 6.4). Što smo dobili dodavanjem gradijenta?
4. Iz USC-SIPI baze slika odaberite sliku *5.1.09.tiff* i pokušajte ukloniti neoštrinu dodavanjem prve (gradijent) i dodavanjem druge derivacije

(Laplaceov operator). Koji postupak daje bolje rezultate? NAPOMENA: Pri dodavanju estimirane druge derivacije estimaciju pribrajate ako je centralni koeficijent Laplaceove maske pozitivan, a oduzimate ako je centralni koeficijent Laplaceove maske negativan.

5. Kakve rezultate dobijete ukoliko kao Laplaceove maske koristite maske:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \text{ i } \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

NAPOMENA: Pokušajte s dijelom koda:

```
>> lap1=fspecial('laplacian',0)
>> lap2=3*fspecial('laplacian',0.5)
```

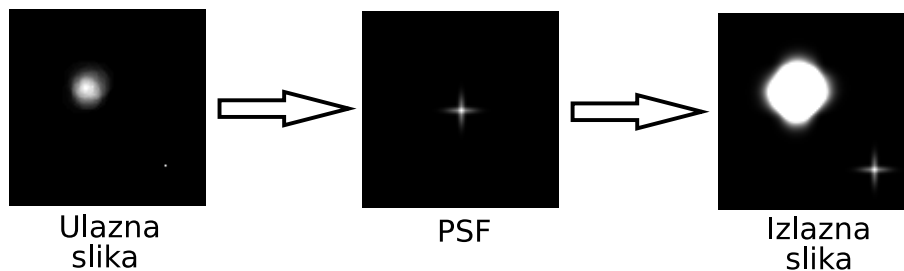
6. Isprobajte obje metode (dodavanje estimacije prve i druge derivacije) na još nekoliko slika. Koja je bitna razlika između dodavanja prve i druge derivacije slici? Koja metoda daje bolje rezultate?

Poglavlje 7.

Vježba 6 - Obnavljanje slike

7.1. Uvod

Kada radimo sa sustavima za dohvatanje slike (npr. teleskopi, fotoaparati ili razna medicinska oprema) susrest ćemo se s pojmovima kao što funkcija razmazivanja točke¹ (ponekad i kernel degradacije²) i optička prijenosna funkcija³). pretpostavimo da promatramo dio noćnog neba, koji izgleda kao Ulazna slika na slici 7.1.. pretpostavimo da je naš teleskop (sustav za dohvatanje slike) u potpunosti opisan svojom prijenosnom funkcijom, te da nema nikakav šum⁴ te da nije prisutan nikakav drugi šum (poput atmosferskog). Ukoliko nam je poznata funkciju razmazivanja točke tada dohvatanje slike možemo prikazati slikom 7.1.. Primijetite na koji način sustav mijenja sliku, odnosno kako PSF ima identičan oblik kao i odziv sustava na točkasti izvor svjetla (dolje desno na slici 7.1.). Upravo po tome je funkcija razmazivanja točke dobila ime.



Slika 7.1.: Sustav za dohvatanje slike

Kako točkasti izvor svjetla možemo matematički opisati kao delta funkciju jasno je da u terminologiji signala i sustava, PSF zapravo predstavlja impulsni odziv sustava (h), a OTF prijenosnu funkciju sustava (H). Poznato vam je da odziv sustava dobivamo konvolucijom slike s PSF u prostornoj domeni, odnosno množenjem s OTF u frekvencijskoj domeni. Ukoliko želimo doznati

¹eng. Point Spread Function (PSF)

²eng. blurr kernel

³eng. Optical Transfer Function (OTF)

⁴eng. sensor noise

kako izgleda originalna slika (bez distorzije koju je u sliku unio sustav za dohvata slike), radimo dekonvoluciju, odnosno definiramo inverzan filtar koji poništava distorziju sustava. Ako je sustav kroz koji je slika propuštena linearni prostorno invarijantan sustav čiju funkciju razmazivanja točke poznajemo, tada možemo definirati inverzan filtar kao:

$$H_i(k, l) = \frac{1}{H(k, l)} \quad (7.1)$$

dok je pseudoinverzni filtar definiran s:

$$H_{pi}(k, l) = \begin{cases} \frac{1}{H(k, l)} & \forall H(k, l) > K \\ 0 & \forall H(k, l) \leq K \end{cases} \quad (7.2)$$

gdje je K proizvoljno odabrana konstanta. (Ponekad se prijenosna funkcija pseudoinverzno filtra dodatno ograničava množenjem s prijenosnom funkcijom nekog niskopropusnog filtra.)

7.1.1. Zadaci

1. Napišite funkciju koja na temelju zadane funkcije razmazivanja točke (modelirane kao impulsni odziva FIR filtra) iz degradirane slike računa originalnu sliku korištenjem inverznog i pseudoinverzno filtra. Pri tome pripazite na razliku između linearne i cirkularne konvolucije kako je objašnjeno u Poglavlju 5..

7.2. Obnavljanje slike

Obnavljanje slike⁵ ima za svrhu dobivanje slike što sličnije originalnoj slici i to samo na temelju poznavanja degradirane slike i načina odnosno modela degradacije. Za razliku od problema poboljšanja slike kojeg je teško strogo matematički formulirati⁶, problem obnavljanja slike je dobro definiran. Degradiranje slike obično modeliramo kao složeni sustav koji se sastoji od linearnog memorijskog sustava i bezmemorijske nelinearnosti. Osim toga realni sustavi uvijek unose šum koji se najčešće modelira kao aditivni šum. Uz zanemarenu bezmemorijsku nelinearnost⁷ problem degradacije i obnavljanja slike se može predstaviti slijedećom blok shemom 7.2..

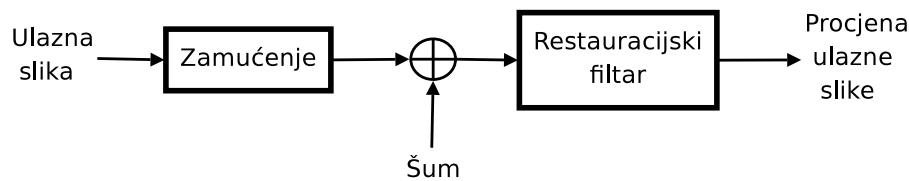
7.3. Modeliranje degradacije slike kao FIR filtra

Linearni sustav koji predstavlja funkciju degradacije ćemo modelirati kao FIR filtar koji će biti predstavljen impulsnim odzivom. Za konstruiranje različitih odziva koristit ćemo funkciju *fspecial()*.

⁵eng. image restoration

⁶eng. ill-posed/conditioned problem, u Hadamardovom smislu to su problemi koji a) nemaju rješenje, b) ono nije jednoznačno ili c) ne ovisi kontinuirano o skupu podataka.

⁷Kako nelinearnost modeliramo kao bezmemorijski sustav, određivanje postupka restauracije svodi se na problem određivanja inverzne funkcije.



Slika 7.2.: Blok shema eksperimenta. Zamućenje će biti aproksimalno linearnim sustavom, a šum senzora aditivnim šumom (ostale vrste šuma su zanezarene). Restauracijskim filtrom pokušavamo dobiti izvornu sliku. Imajmo na umu da je to tek procjena, čija točnost ovisi o pretpostavljenom modelu.

7.3.1. Primjer

```

>> h=fspecial('gaussian',4,1); % Gaussovo zamućenje
>> mesh(h) % prikaz odziva u vremenskoj domeni
>> freqz2(h) % frekvencijska karakteristika u 3D
  
```

Kako sve degradacije modeliramo kao linearni prostorno nepromjenjiv sustav, rezultat degradacije računamo linearnom konvolucijom. Ako želite primijeniti više različitih degradacija (npr. atmosfersko zamućenje i pomak kamere) morate računati više uzastopnih konvolucija. Odziv takvog složenog sustava možete odrediti računajući linearnu konvoluciju odziva pojedinih sustava. U sljedećem primjeru slabo atmosfersko zamućenje je modelirano filtrom Gaussovog oblika, a jednoliki pomak kamere u horizontalnom i vertikalnom smjeru je simuliran jednakim pomakom od 20 točaka po x i y osi zajedno:

7.3.2. Primjer

```

>> h1=eye(20)/20; % uniforman pomak od 20 točaka
>> h2=fspecial('gaussian',4,1); % Gaussovo zamućenje
>> h=conv2(h1,h2); % ukupno zamućenje je konvolucija
% dva impulsa odziva
>> mesh(h) % odziv u vremenskoj domeni
>> freqz2(h) % frekvencijska karakteristika
>> img=imread('moon.tif'); % učitavamo sliku
>> imgZ=conv2(img,h); % primijenjujemo zamućenje
>> imagesc(imgZ); % prikazujemo degradiranu sliku
>> colormap(gray) % prikazujemo sliku u sivim tonovima
  
```

7.3.3. Zadaci

1. Odredite PSF i OTF za karakteristična zamućenja, npr. slabo atmosfersko zamućenje i vertikalne i horizontalne (zajedno) pomake za nekoliko točaka.

2. Odredite i skicirajte PSF i OTF za kombinaciju zamućenja, i to za slabo atmosfersko zamućenje i vertikalni te horizontalni pomak od desetak točaka.
3. Primijenite dobivenu PSF na sliku *salona.png* (ili npr. *2.2.01.tiff* i *2.1.08.tiff* iz USC-SIPI baze). Odabranu PSF koristite kroz sve ostale zadatke.
4. Primijenite dobivenu PSF na sliku *klls1.jpg* na način da svaki kanal zasebno zamutite istim zamućenjem. Prikažite konačnu RGB sliku. Objasnite porijeklo crnog okvira na slici. O čemu će ovisiti njegova debljina?

7.4. Inverzni filter

Inverzni filter je određen izrazom

$$H_i(k, l) = \frac{1}{H(k, l)}.$$

Kako dani izraz uključuje računanje inverza, dobiveni filter može biti nestabilan. Osim nestabilnosti za većinu primjena dobiveni filter nije pretjerano upotrebljiv zbog velike osjetljivosti na šum. Zbog neizbježne kvantizacije slike na konačan broj diskretnih razina čak i u slučaju zanemarivog aditivnog šuma uvijek je prisutan šum kvantizacije. Ako se izvorna slika nalazi u varijabli *img*, a obnovljena u varijabli *imgO*, srednje kvadratno odstupanje određujemo na slijedeći način:

7.4.1. Primjer

```
>> img=im2double(img); % slike pretvaramo u double
>> img0=im2double(img0); % i skaliramo na raspon [0,1]
>> mean2((img-img0).^2) % računamo srednje kvadratno odstupanje
```

Ponekad nas osim srednjeg kvadratnog odstupanja zanima i omjer snaga signal/šum ili općenito signal/signal:

7.4.2. Primjer

```
>> imgP=abs(fft2(img)).^2; % spektar snage izvorne slike
>> imgPsr=mean2(imgP) % srednja snaga izvorne slike
```

```
imgPsr =
```

```
1.1467e+005
```

```
>> imgOP=abs(fft2(img0)).^2; % spektar snage obnovljene slike
>> imgOPsr=mean2(imgOP) % srednja snaga obnovljene slike
```

```

imgOPsr =

    1.2677e+005

>> imgPsr/imgOPsr % odnos izvorna/obnovljena slika
    % primijetite da ste ga također mogli
    % izračunati u dB
ans =

    0.9045

```

Ponekad je teško vizualizirati ponašanje filtra uspoređujući samo slike prije i nakon filtriranja. U tom slučaju se najčešće odabiru pojedine linije slike, bilo redci bilo stupci, koje se prikazuju kao jednodimenzionalni signal:

7.4.3. Primjer

```

>> size(imgZ) % određujemo veličinu slike

ans =

    534    534

>> plot(imgZ(128,:)) % prikazujemo 128 redak zamućene slike
>> plot(img0(128,:)) % prikazujemo 128 redak obnovljene slike

```

7.4.4. Zadatci

1. Za odabranu PSF degradacije prikažite rezultat inverznog filtriranja za slučaj degradacije s i bez aditivnog šuma za nekoliko slika.
2. Odredite srednje kvadratno odstupanje obnovljene slike od izvorne slike u oba slučaja.
3. Isprobajte inverzno filtriranje kao u prethodnom zadatku, ali uz kvantizaciju razina na 8 bita (vrijednosti od 0 do 255). Prikažite rezultate.
4. Odredite srednje kvadratno odstupanje obnovljene slike od izvorne slike u oba slučaja. Kako kvantizacija utječe na rezultat filtriranja?

7.5. Pseudoinverzni filter

Pseudoinverzni filter je stabilizirani inverzni filter. Ponekad se prijenosna funkcija pseudoinverznog filtra dodatno ograničava množenjem s prijenosnom funkcijom nekog niskopropusnog filtra, odnosno ograničavanjem odziva samo na niže frekvencije.

7.5.1. Zadatci

1. Za odabranu PSF odredite rezultat pseudoinverznog filtriranja za slučaj degradacije s i bez aditivnog šuma za nekoliko slika.
2. Odredite srednje kvadratno odstupanje obnovljene slike od izvorne slike.
3. Usporedite rezultate s rezultatima inverznog filtriranja.
4. Isprobajte pseudoinverzno filtriranje kao u prethodnom zadatku, ali uz kvantizaciju razina na 8 bita (vrijednosti od 0 do 255).
5. Odredite srednje kvadratno odstupanje obnovljene slike od izvorne slike.
6. Usporedite rezultate s rezultatima inverznog filtriranja.
7. Kako kvantizacija utječe na rezultat filtriranja?

7.6. Wienerov filtar

Uz pretpostavku stacionarnosti u širem smislu slučajnog procesa koji predstavlja model slike te uz prisutan nekorelirani aditivni šum srednje vrijednosti nula Wienerov filtar je određen izrazom

$$H_w[k, l] = \frac{H^*[k, l]}{H^*[k, l]H[k, l] + \frac{S_{NN}[k, l]}{S_{XX}[k, l]}} \quad (7.3)$$

gdje je $H[k, l]$ optička prijenosna funkcija, dok su $S_{NN}[k, l]$ i $S_{XX}[k, l]$ gustoće spektra snage signala i šuma. Zvezdica (*) označava kompleksnu konjugaciju. Izraz se može zapisati i u slijedećim obliku:

$$H_w[k, l] = \frac{1}{H(f)} \frac{|H[k, l]|^2 S_{XX}[k, l]}{|H[k, l]|^2 S_{XX}[k, l] + S_{XX}[k, l]} \quad (7.4)$$

Odnosno:

$$H_w[k, l] = \frac{1}{H(f)} \frac{|H[k, l]|^2}{|H[k, l]|^2 + \frac{1}{SNR[k, l]}} \quad (7.5)$$

Gdje sa $SNR[k, l]$ označavamo funkciju odnosa signala i šuma⁸. Izraz 7.5 nam je ujedno i zgodniji za daljnje razmatranje jer pokazuje kako Wienerov filtar možemo prikazati kaskadom inverznog filtra i filtra čija prijenosna karakteristika ovisi o odnosu signal šum na toj frekvenciji. Tako će frekvencije koje nemaju šuma ($SNR = \infty$) imati prijenosnu karakteristiku jednaku jedan (prenosit će se bez atenuacije), dok će sve ostale na kojima šum postoji atenuirati s nekim faktorom u intervalu $[0, 1)$.

Problem pri određivanju prijenosne funkcije Wienerovog filtra je upravo estimacija gustoće spektara snage (odnosno autokorelacijskih funkcija signala i

⁸eng. SNR - signal-to-noise-ratio

šuma) te optičke prijenosne funkcije jer najčešće raspolažemo samo s degradiranom slikom. Stoga ćemo u prvoj aproksimaciji pretpostaviti konstantni omjer gustoća spektara snage K , te je prijenosna funkcija Wienerovog filtra

$$H_w[k, l] = \frac{1}{H(f)} \frac{|H[k, l]|^2}{|H[k, l]|^2 + K} \quad (7.6)$$

Pri tome se parametar K može procijeniti iz degradirane slike. Primijetite da za male vrijednosti parametra K Wienerov filtar postaje blizak inverznom filtru.

Primjer uklanjanja zamućenja uz aditivni bijeli šum uz korištenje aproksimacije Wienerovog filtra:

7.6.1. Primjer

```
>> img=imread('kljakovic2.png'); % učitavamo fotografiju
>> img=rgb2gray(img); % slika mora biti siva...
>> img=img2double(img); % ... tipa double
>> imgZ=conv2(img,h); % zamućenje
>> imgZ=imgZ/imgZ(:); % skaliranje zbog dodavanja šuma
>> imgZN=imnoise(imgZ,'gaussian',0,.001); % dodajemo šum
>> imshow(imgZN)
>> img0=imwiener(imgZN,h,0.013); % Wienerov filtar
>> imshow(img0) % obnovljena slika
```

7.6.2. Zadaci

1. Koristeći se kodom kojim ste implementirali pseudoinverzni filtar, napišite funkciju koja implementira Wienerov filtar u obliku danom jednadžbom 7.6. U slijedećim zadacima pretpostavljat ćemo da ne poznajemo vrijednosti gustoće spektra snage slike i šuma, odnosno da odnos spektara modeliramo samo konstantom K .
2. Odredite rezultat Wienerovog filtriranja za slučaj aditivnog Gaussovog šuma srednje vrijednosti nula. Kako biste procijenili K u prvom koraku?
3. Za ranije odabranu PSF (zadatak 7.3.3.) odredite rezultat Wienerovog filtriranja za slučaj degradacije bez aditivnog šuma.
4. Prikažite rezultat Wienerovog filtriranja za slučaj degradacije uz postojanje aditivnog šuma.
5. U sva tri slučaja odredite srednje kvadratno odstupanje obnovljene slike od izvorne slike.
6. Usporedite rezultate filtracije s rezultatima inverznog i pseudoinverzong filtriranja.
7. Koliki je odnos signal/šum prije i nakon filtracije?

Za procjenu autokorelacijske funkcije napisana je za potrebe vježbi funkcija *akf()*. Bez zadanih parametara funkcija računa prostornu autokorelacijsku funkciju do polovine dimenzija slike zbog izbjegavanja rubnih efekata. Za realne slike može se pretpostaviti da postoji koreliranost točaka koje su udaljene do 30-tak pomaka, tako da nije potrebno računati cijelu autokorelacijsku funkciju:

7.6.3. Primjer

```
>> help akf % proučimo funkciju akf

>> [img,map]=imread('kljakovic3.png'); % učitavamo sliku
>> img = rgb2gray(img); % pretvorimo je u crno-bijelu
>> rxx=akf(img,30); % računamo AKF za prvih 30 pomaka po x i y
>> imagesc(rxx) % prikazujemo izračunatu AKF
```

7.6.4. Zadatci

1. Degradirajte sliku dodavanjem aditivnog Gaussovog šuma srednje vrijednosti nula.
2. Izračunajte i prikažite autokorelacijsku funkciju R_{XX} estimirane iz degradirane slike⁹.
3. Izračunajte i prikažite autokorelacijsku funkciju šuma (R_{NN}).
4. Odredite rezultat Wienerovog filtriranja, uz poznajete autokorelacijske funkcije signala i šuma (dakle, uz poznate gustoće spektra snage šuma i slike).
5. Usporedite rezultat s autokorelacijskom funkciju R_{XX} izračunatom ne izvornoj slici.
6. Ponovite zadatke za sliku degradiranu s ranije definiranim PSF (zadatak 7.3.3.) i aditivni Gaussov šum. Usporedite i komentirajte rezultate.
7. Sliku degradiranu s ranije definiranim PSF (zadatak 7.3.3.) i aditivni Gaussov šum pokušajte filtrirati funkcijom *wiener2()*. Jeste li dobili iste rezultate?

⁹Estimacija autokorelacijske funkcije slike (alternativno, estimacija gustoće spektra snage) izravno iz degradirane slike računanjem autokorelacije nije potpuno ispravna. Takav postupak se obično koristi pri iterativnom Wienerovom filtriranju i predstavlja prvi korak filtracije. Bolja estimacija autokorelacijske funkcije se tada računa iz filtrirane slike te se postupak ponavlja.

Poglavlje 8.

Vježba 7 - Pronalaženje značajki slike

8.1. Prostorne značajke

Značajke možemo razvrstati u odnosu na to u kojem ih prostoru tražimo, odnosno u kojem prostoru signal egzistira. Tako tipično možemo imati prostorne, vremenske (temporalne), frekvencijske¹ itd. značajke. Kako pod pojmom "slika" najčešće podrazumijevamo sliku u 2D prostoru, u najvećem broju aplikacija potrebno je izdvojiti upravo prostorne značajke slike. Na temelju takvih značajki u područjima analize slike i računalnog vida² računalo pokušava analizirati, odnosno opisati i interpretirati sadržaj slike. Zbog toga možemo reći da je pronalaženje značajki prva karika u računalnoj analizi i interpretaciji slike.

8.2. Amplitudne značajke slike

Amplitudne značajke slike za svaku točku slike određujemo kao neke parametre izračunate iz vrijednosti intenziteta točaka u okolini promatrane točke. U MATLAB-u postoji nekoliko funkcija koje primjenjuju zadanu operaciju na definiranu okolinu za svaku točku slike (*blkproc*, *colfilt*, *nlfilt*). Mi ćemo koristiti funkciju *colfilt*, no za to se trebamo upoznati s pokazivačima na funkciju, odnosno kako ih koristiti u MATLAB-u.

8.2.1. Primjer

```
>> img = imread('texture.tif'); % učitajmo sliku
>> fun = @(x) mean(x); % pointer funkcije (function handle)
>> img2 = colfilt(img,[32 32],'sliding',fun);
% procesiramo sliku
>> figure; imagesc(img2); colormap(gray) % prikaz slike
```

¹eng. spatial, temporal, frequency

²eng. machine vision

Kako smo definirali pokazivač *fun* na funkciju *mean()*, možemo definirati na bilo koju funkciju koju sami napišemo.

8.2.2. Zadatci

1. Odaberite neku sliku manjih dimenzija, do 256×256 (možete i smanjiti neku veću sliku). Za odabranu sliku odredite nekoliko amplitudnih značajki (funkcijama *max()*, *min()*, *mean()*, *std()*).
2. Kako rezultati ovise o veličini odabranog bloka?

8.3. Značajke histograma prvog reda

Vrijednosti amplituda unutar promatrane okoline neke točke možemo interpretirati kao rezultate slučajnog eksperimenta. Tada nam histogram prvog reda određen za promatranu okolinu predstavlja procjenu funkcije gustoće vjerojatnosti. Na temelju te procjene možemo odrediti neke statističke značajke kao što su momenti i centralni momenti funkcije gustoće vjerojatnosti³. Osim momenata zanimljive značajke su nam entropija i energija histograma. Za ovo će nam biti potrebna funkcije računanja histograma, čiji primjer je dan Primjerom 8.3.1.. Kako funkcije *hist()* prima niz odnosno vektor podataka u jednoj naredbi smo sliku pretvorili u niz i predali je kao parametar funkciji.

8.3.1. Primjer

```
>> img = imread('texture.tif'); % učitajmo sliku
>> hist(im2double(img(:)),30); % računa i prikazuje histogram
    % slike; grupiranje vrijednosti histograma u 30 polja
```

8.3.2. Zadatci

1. Napišite funkcije za računanje momenata, entropije i energije⁴, tako da ih možete pozvati funkcijom *colfilt()*.
2. Odaberite neku sliku manjih dimenzija, do 256×256 (možete i smanjiti neku veću sliku). Za odabranu sliku odredite značajke histograma prvog reda (entropije i energije) na prozoru veličine 5×5 koristeći funkciju *colfilt()*. Prikažite i komentirajte rezultate.

8.4. Histogram drugog reda

Možemo pretpostaviti da su vrijednosti intenziteta za svaku točku ishod nekog slučajnog eksperimenta. Uz takvu pretpostavku histogram prvog reda predstavlja procjena funkcije gustoće vjerojatnosti. Umjesto samo jedne točke

³eng. probability density function - PDF

⁴Formule su dane u predavanjima.

možemo na sličan način promatrati bilo koji par točaka slike čija je međusobna pozicija određena nekom relacijom. U tom slučaju naš pretpostavljeni slučajni eksperiment postaje dvodimenzionalan, a procjenu odnosno estimaciju funkcije gustoće vjerojatnosti dobivamo iz histograma drugog reda.

8.4.1. Primjer

```
>> [img,map]=imread('testpat1.tif'); % učitavamo sliku
>> img = im2double(img); % pretvaramo sliku u double format
>> img1 = img(4:end,3:end); % režemo slike tako da pomak
>> img2 = img(1:end-3,1:end-2); % između njih bude (2,3)
>> hist3([img1(:), img2(:)])
      % računa i prikazuje histogram drugog reda
>> img1 = img(2:end,6:end); % režemo slike tako da pomak
>> img2 = img(1:end-1:1:end-5); % između njih bude (5,1)
>> hist3([img1(:), img2(:)])
      % računa i prikazuje histogram drugog reda
```

Primijetite da su histogrami različiti za različite pomake.

8.4.2. Zadaci

1. Učitajte sliku *4.2.04.tiff* iz USC-SIPI baze, ili npr. sliku *kljakovic3.png*. Odredite histograme drugog reda za pomake [11] i [55].
2. Zašto su rezultati grupirani oko dijagonale?
3. Zašto su u drugom slučaju raspršeniji nego u prvom?
4. Učitajte sliku *tombak.png* te odredite histograme drugog reda za nekoliko raznih pomaka. Prikažite i komentirajte rezultate.

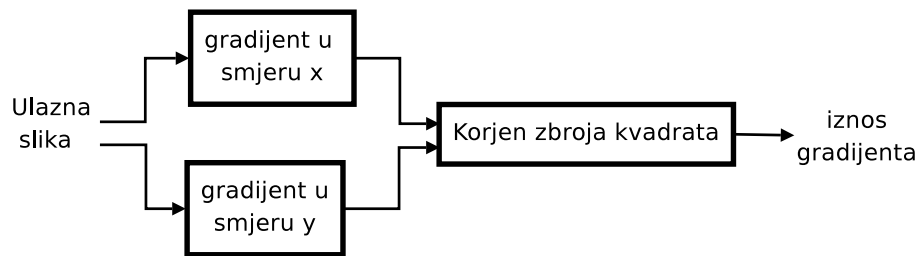
8.5. Detekcija rubova

Detekcija rubova je važna u analizi slika jer rubovi najčešće određuju granice objekata na slici. Pronalaženje tih granica je korisno za segmentaciju, registraciju i identifikaciju objekata na slici. Kako su rubovi mjesta naglih promjena u vrijednosti amplitude točaka slike, u detekciju rubova najčešće se koriste metode za procjenu gradijenta.

8.6. Sobelov i Prewittov operator

Kod gradijentnih metoda obično procjenjujemo iznos gradijenta u dva smjera, x i y . Ta dva estimirana vektora nam tada određuju smjer i iznos gradijenta

u promatranoj točki. Obično nas zanima iznos gradijenta kojeg računamo kao korijen zbroja kvadrata kako je prikazano na slici⁵.



Slika 8.1.: Računanje gradijenta

Maske za Previttov i Sobelov operator za detekciju vodoravnih rubova možemo jednostavno dobiti pozivom funkcije *fspecial()*:

8.6.1. Primjer

```
>> sy=fspecial('sobel') % sobelova maska za vodoravne rubove
```

```
sy =
```

```
    1     2     1
    0     0     0
   -1    -2    -1
```

```
>> sx=rot90(sy) % rotiramo je da dobijemo masku za okomite
% rubove
```

```
sx =
```

```
    1     0    -1
    2     0    -2
    1     0    -1
```

Kada znamo maske za vodoravni i okomiti smjer računamo konvolucije slike s tim maskama, odnosno određujemo estimacije gradijenta za x i y smjer:

8.6.2. Primjer

```
>> [img,map]=imread('uskoci1.png'); % učitamo sliku
```

⁵Ovdje je prikazan matematički ispravan postupak gdje računamo korijen iz zbroja kvadrata. Kako je određivanje gradijenta jedna od učestalih operacija pri obradbi slike zbog smanjenja kompleksnosti i ubrzavanja postupka često se korijen iz zbroja kvadrata aproksimira jednostavnijim izrazom, npr. zbrojem apsolutnih vrijednosti. Time se naravno unosi mala pogreška, no uz znatno ubrzanje postupka.

```

>> img = rgb2gray(im2double(img)); % pretvaramo u sivu
                                     % sliku tipa double
>> gx=conv2(img,sx,'same');          % gradijent u x smjeru
>> gy=conv2(img,sy,'same');          % gradijent u y smjeru
>> g=sqrt(gx.*gx+gy.*gy);           % iznos gradijenta
>> imagesc(g)                       % crtamo ga

```

8.6.3. Zadatci

1. Učitajte sliku *knjiga_ssa.png*. Procijenite iznos gradijenta koristeći Sobelov i Prewittov operator⁶.
2. Prikažite i komentirajte rezultate.

8.7. Kompas operatori za detekciju ruba

Jednostavne gradijentne metode estimiraju gradijent u dva smjera. Kompas operatori su zamišljeni tako da detektiraju rubove koji se nalaze pod određenim kutom. Primjer maski veličine 3×3 za kutove od 0° , 45° te 90° :

8.7.1. Primjer

```

>> h0=[1 1 1; 0 0 0; -1 -1 -1] % maska za kut 0°

```

h0 =

```

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

```

```

>> h45=[1 1 0; 1 0 -1; 0 -1 -1] % maska za kut 45°

```

h45 =

```

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

```

```

>> h90=[1 0 -1; 1 0 -1; 1 0 -1] % maska za kut 90°

```

⁶ Primijetite da Sobelove i Prewittove maske imaju samo cjelobrojne koeficijente. I ostale slične maske, npr. Abdou piramidalna maska, imaju cjelobrojne koeficijente, prvenstveno zbog mogućnosti korištenja cjelobrojne aritmetike koja donosi značajno ubrzanje postupka određivanja gradijenta. Operatori za estimaciju gradijenta određeni takvim maskama su cjelobrojne aproksimacije i nisu izotropni, odnosno ne dobivamo jednaku vrijednost gradijenta za horizontalne, vertikalne i dijagonalne rubove. Primjer izotropne maske je Frei-Chen maska, no ona nije izvediva u cjelobrojnoj aritmetici.

h90 =

```
1    0   -1
1    0   -1
1    0   -1
```

Konvolucija slike s odabranim kompas operatorima ističe rubove u odabranom smjeru.

8.7.2. Zadatci

1. Učitajte sliku *knjiga_ssa.png*. Primijenite na nju nekoliko kompas operatora za različite smjerove.
2. Što se događa sa rubovima koji nisu pod kutom za koji je predviđen pojedini operator?
3. Frekvencijsku karakteristiku gradijentnog kompas operatora moguće je odrediti koristeći funkciju *fft2()* kao što smo radili u Poglavlju 5.. Odredite frekvencijske karakteristike za kompas operatore.
4. Zašto su te karakteristike toliko male razlučivosti?

8.8. Laplaceov operator

Gradijentne maske daju najbolje rezultate za oštre rubove koji odgovaraju nagloj promjeni vrijednosti intenziteta točaka. Kada rubovi postaju blaži bolje rezultate daju metode koje procjenjuju druge derivacije. Za to se često koristi Laplaceov operator. Maske za aproksimaciju Laplaceovog operatora veličine 3×3 također se mogu generirati pozivom funkcije *fspecial()*:

8.8.1. Primjer

```
>> lap=fspecial('laplacian',0) % maska koja aproksimira Laplaceov
% operator
```

lap =

```
0    1    0
1   -4    1
0    1    0
```

8.8.2. Zadatci

1. Učitajte sliku *knjiga_ssa.png*. Primijenite na nju nekoliko aproksimacija Laplaceovog operatora (različiti drugi parametar funkcije *fspecial()*). Komentirajte rezultate.

8.9. Značajke teksture

Teksture se javljaju na gotovo svim slikama koje svakodnevno susrećemo. Iako imamo jasnu intuitivnu predodžbu teksture i usprkos njihovoj važnosti ne postoji općeprihvaćena formalna definicija teksture⁷. Kao značajke teksture u ovoj vježbi koristiti će se značajke histograma drugoga reda⁸.

Ovaj put ćemo značajke računati na blokovima odabrane veličine u ulaznoj slici pomoću funkcije *nlfilter()*.

8.9.1. Primjer

```
>> [img,map]=imread('testpat1.tif'); % učitavamo sliku
>> img = im2double(img); % pretvaramo sliku u double format
>> p = [2,2]; % definiramo pomak
>> okolina = [15, 15]; % definiramo okolinu
>> fun = @(x) inertia(x,p); % pointer funkcije (function handle)
>> img2 = nlfilter(img,okolina,fun); % procesiramo sliku
>> figure; imagesc(img2); colormap(gray) % prikaz slike
```

Primijetite da je za određivanje histograma drugog reda potrebno odabrati neki vektor pomaka $p = [D_y, D_x]$ koji nam određuje međusobni odnos dviju točaka koje promatramo. To nam predstavlja problem jer već i za male okoline ($okolina = [M, N]$) broj mogućih izbora postaje prevelik. Pri odabiru veličine okoline te vektora pomaka $[D_y, D_x]$ imajte u vidu da vektor $[D_y, D_x]$ mora biti manji od okoline. Da računanje značajki ne bi potrajalo dulje vrijeme odaberite slike manjih dimenzija i manje veličine prozora.

8.9.2. Zadatci

1. Učitajte sliku *texture.tif*. Na toj slici nalazi se pet različitih područja, i to tri različite teksture u gornjem dijelu slike, te dva jednobojna područja u donjem dijelu slike. Odredite značajke (*absolute()*, *inertia()*, *covariance()*, *energy()*, *entropy()* - dane istoimenim funkcijama) za nekoliko različitih pomaka (npr. pomaci $[2, 2]$, $[4, 4]$ i $[3, 5]$) uz veličinu bloka od $[12, 12]$.
2. Prikažite dobivene rezultate i komentirajte ih.
3. Odaberite par slika različitih tekstura (izrežite dio manjih dimenzija iz veće slike). Odaberite jednu od značajki te je odredite za svaku od tih

⁷IEEE Standard Glossary of Image Processing and Pattern Recognition daje dosta široku definiciju teksture koja opet nije opće prihvaćena: "Texture is an attribute representing the spatial arrangement of the gray levels of the pixels in a region."

⁸U engleskom jeziku često susrećemo i izraz "co-occurrence matrix". Primijetite da on označava histogram drugog reda u kojem se dvije slike razlikuju za međusoban pomak, dok općenito, histogram drugog reda nije tako usko definiran, odnosno, možemo ga računati za bilo koje dvije slike.

tekstura. Koristite slike teksture iz baze slika na stranicama predmeta ili iz USC-SIPI baze.

4. Razlikuju li se značajke dovoljno za automatsku klasifikaciju tih tekstura?
5. Koja je značajka najbolja za klasifikaciju odabranih tekstura?

Poglavlje 9.

Vježba 8 - Segmentacija slike

9.1. Uvod

1. Kako funkcionira algoritam k-srednjih vrijednosti (funkcija *kmeans()*).
2. Što radi funkcija *reshape()*?
3. Napišite funkciju koja kao značajku tekstone računa energiju signala bez istosmjerne komponente¹. Prisjetite se kako ste u prošloj vježbi napisali funkcije za računanje momenata i entropije (Zadatak 8.3.2.).

9.2. Amplitudna segmentacija

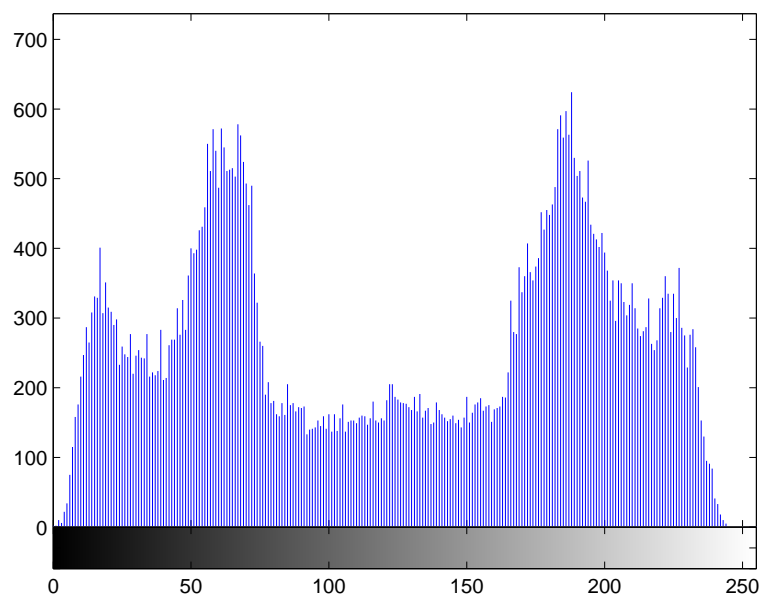
Amplitudna segmentacija je najjednostavnija metoda segmentacije slike. Korisna je kad amplitudne značajke dovoljno precizno definiraju regije scene. Najčešće se za amplitudnu segmentaciju koristi histogram prvog reda.

9.3. Ručno odabiranje praga

U ovom dijelu vježbe potrebno je na osnovu histograma slike odrediti prag segmentacije. Ukoliko na slici postoji više različitih regija takvih da je svjetlina točaka unutar regije otprilike jednaka, očekujemo bimodalan ili multimodalan histogram. Jedan takav histogram prikazan je na slici 9.1.. Na tom histogramu vidimo nekoliko skupina, te bi u slučaju segmentacije na dvije regije za prag odabrali vrijednost oko 130.

Za amplitudnu segmentaciju koristimo funkciju *grayscale()* koja ulazne slike segmentira s zadanim pragovima. Pragovi se navode u jednom vektoru i moraju biti između 0 i 1. Ako izlaz funkcije *grayscale()* nismo pridružili nekoj varijabli, funkcija odmah crta rezultat segmentacije. Postupak je opisan u slijedećem primjeru:

¹Uputa: Za svaki blok odredite 2D Fourierovu transformaciju te na mjesto istosmjerne komponente upišite nulu. Sada iz tako modificiranog spektra odredite energiju signala (Parsevalova relacija).



Slika 9.1.: Računanje gradijenta

9.3.1. Primjer

```
>> [img,map]=imread('testpat1.tif'); % učitavamo sliku
>> imhist(img) % prikazujemo histogram
>> grayslice(im2double(img),130/255)
    % segmentiramo sliku pragom koji smo odabrali
```

Ako histogram interpretiramo kao višemodalni, istom naredbom sliku segmentiramo u tri područja definirajući dva praga:

```
>> grayslice(im2double(img),[45 130]/255)
    % segmentiramo sliku s pragovima 45 i 130
```

9.3.2. Zadatci

1. Učitajte sliku *blood1.tif*. Na toj slici prikazana su krvna zrnca. Kao rezultat segmentacije potrebno je dobiti pojedina krvna zrnca uz što manje šuma. Na tako dobivenoj binarnoj slici krvna zrnca su objekti kružnog oblika sa šupljinom u sredini. Prikažite histogram te odredite prag. Dobivate li segmentirana krvna zrnca?
2. Za sliku *blood1.tif* prikažite rezultat segmentacije s odabranim pragom te rezultate koji se dobiju za prag 0.2 veći i 0.2 manji od optimalnog². Opišite rezultate.

²Kao optimalan prag obično biramo minimum između dva maksimuma bimodalnog histograma. Ovaj kriterij nekada nazivamo i kriterij srednjeg minimuma.

3. Ukoliko histogram slike *blood1.tif* interpretiramo kao trimodalni gdje bi se nalazio drugi prag? Što dobivamo u ovom slučaju kao rezultat segmentacije?

9.4. Automatsko odabiranje praga

U nastavku vježbe biti će prikazana metoda K-srednjih vrijednosti koja služi za automatsko određivanje praga iz histograma. Za automatsko odabiranje praga koristit ćemo funkciju *kmeans()*:

9.4.1. Primjer

```
>> [img,map]=imread('testpat1.tif'); % učitavamo sliku
>> [idx, c] = kmeans(im2double(img(:)), 3);
                                % određujemo segmentaciju
>> figure; imagesc(reshape(idx,size(img)));
                                % prikazujemo segmentaciju
```

Primijetite kako se parametar *c*, koji predstavlja centre razreda u koje grupiramo intenzitete slike, mijenja opetovanim pozivanjem funkcije. Ako želimo da nam funkcija K-srednjih vrijednosti uvijek vraća jednak rezultat pozvat ćemo je na slijedeći način:

```
>> [idx, c] = kmeans(im2double(img(:)),3, 'start', ...
                    [0.5:3]'/3, 'onlinephase', 'off');
```

Sada parametar *c* ostaje nepromijenjen opetovanim pozivanjem funkcije, jer smo početne odabire centara postavili da budu fiksni ($\frac{1}{6}, \frac{1}{2}, \frac{5}{6}$). Posljednja dva parametra služe za ubrzanje algoritma (nauštrb preciznosti). Ovaj poziv ćemo koristiti u nastavku vježbe.

9.4.2. Zadatci

1. Učitajte sliku *blood1.tif*. Usporediti vrijednosti dobivene metodom K-srednjih vrijednosti sa vrijednostima dobivenim ručnim odabirom.

9.5. Određivanje rubova

Za određivanje rubova koristit ćemo funkciju *edge()*. Podržane je nekoliko različitih metoda određivanja rubova. Funkcija kao rezultat vraća binarnu sliku veličine ulazne slike na kojoj su označeni detektirani rubovi. Podržane metode su *'sobel'*, *'prewitt'*, *'roberts'*, *'zerocross'*, *'log'* i *'canny'*. Za metode koje se oslanjaju na procjenu prve derivacije estimacija derivacije se računa za svaku točku te se na tako dobivenoj slici vrši amplitudna segmentacija. Prag za amplitudnu segmentaciju moguće je zadati unaprijed. Za metode koje se oslanjaju na procjenu druge derivacije detektira se prolaz kroz nulu. I za te

metode je moguće zadati prag, no prag tada određuje strminu ruba. Naredbu pozivamo na sljedeći način:

9.5.1. Primjer

```
>> [img,map]=imread('knjiga_ssa.png'); % učitavamo sliku
>> rub=edge(img,'sobel');           % primjenimo Sobelovu metodu
>> imshow(rub)                     % prikažimo detektirane rubove
>> rub=edge(img,'sobel',0.02);     % zadali smo osjetljivost 0,02
>> imshow(rub)
>> rub=edge(img,'log',[ ],4);      % primjenjujemo LoG metodu
                                   % uz devijaciju 4
>> imshow(rub)
```

9.5.2. Zadaci

1. Učitajte sliku *uskoci1.png*. Primijenite na sliku funkciju *edge* te prikažite rezultate za Sobelov operator uz razne vrijednosti praga. Za koju vrijednost se dobije najbolji rezultat?
2. Na slici *uskoci1.png* primijenite funkciju *edge()* uz automatski odabir praga. Razlikuje li se ta vrijednost od vrijednosti koju ste dobili u prethodnom zadatku? Za koji slučaj dobivamo bolji rezultat?
3. Na slici *uskoci1.png* primijenite *'zerocross'* ili *'log'* metodu. Dodajte šum slici pomoću naredbe *imnoise()* te usporedite rezultate *'zerocross'* ili *'log'* metode s rezultatima za Sobelov operator.
4. Na slici *uskoci1.png* primijenite Cannyjevu metodu. Usporedite rezultate s rezultatima detekcije za ostale metode.

9.6. Segmentacija tekstura

Teksture ćemo segmentirati na temelju nekih odabranih značajki. Najprije ćemo za svaku točku slike izračunati odabranu značajku na odabranoj okolini. Izračunate značajke možemo promatrati kao sliku na kojoj se može izvršiti amplitudna segmentacija. Značajke teksture smo računali u prošloj vježbi koristeći funkciju *colfilt()*. Sada ćemo jednostavno rezultat te funkcije prosljediti funkciji *kmeans()* za automatsku amplitudnu segmentaciju:

9.6.1. Primjer

```
>> [img,map]=imread('texture.tif');
>> p = [2,3]; % definiramo pomak
>> fun = @(x) inertia(x,p); % pointer funkcije (function handle)
>> znacajke = nlfilt(img,[10 10],fun); % procesiramo sliku
>> figure; imagesc(znacajke); colormap(gray)
```

```
>> [idx, c] = kmeans(im2double(znacajke(:)), 5, 'start', ...  
    [0.5:5]'/5, 'onlinephase', 'off');  
>> figure; imagesc(reshape(idx,size(img)));  
    % prikazujemo segmentaciju
```

9.6.2. Zadatci

1. Učitajte sliku *texture.tif*. Prikažite je i opišite teksture na slici. Koliko ih ima?
2. Odaberite te odredite neke značajke teksture na okolini veličine 12×12 . Prikažite dobivene značajke te procijenite koje su odgovarajuće za segmentiranje promatranih tekstura. Za odabrane značajke primijenite metodu K srednjih vrijednosti te komentirajte rezultate.
3. Koristeći funkciju koju ste napisali za pripremu izračunajte energiju spektra bez istosmjerne komponente uz okolinu veličine 12×12 . Je li ova značajka dobra za segmentiranje tekstura sa slike? Segmentirajte matricu značajki korištenjem algoritma K-srednjih vrijednosti te komentirajte rezultate.

Poglavlje 10.

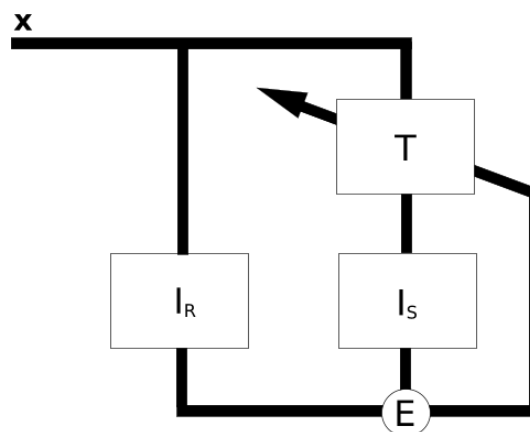
Vježba 9 - Registracija slike

10.1. Uvod

1. Proučite kod funkcija `corr2()`, `mesh()`, `rectangle()`, `imrotate()`.
2. Što radi naredba `hold on`?

10.2. Registracija slike

Registracija slike sastoji se od transformacije slike, mjere sličnosti kojom se određuje kvaliteta registracije i funkcijom optimizacije kojom se maksimizira mjera sličnosti. Ta tri osnovna elementa registracije slike prikazana su na slici 10.1.. Registriramo ulaznu¹ sliku (I_S) na referentu² sliku (I_R) tako što primijenjujemo transformaciju \mathcal{T} na prostor u kojem se slika I_S nalazi. Mjera sličnosti je neka mjera energije ili pogreške (koju sami definiramo) i označena je sa E , dok je optimizacijski algoritam prikazan povratnom vezom.



Slika 10.1.: Registracija slike

¹eng. source

²eng. reference

Geometrijska transformacija se vrši na prostoru u kojem slika prebiva, a ne na intenzitetima slike, zbog toga matematički pišemo transformaciju u obliku:

$$I_T(\mathbf{x}) = I_S(\mathcal{T}(\mathbf{x})) \quad (10.1)$$

gdje nam \mathbf{x} označava vektor (x, y) , ako je slika u 2D prostoru.

Radi jednostavnosti, optimizaciju u ovoj vježbi ne ćemo vršiti već ćemo problem riješiti potpunim pretraživanjem prostora, te traženjem maksimuma. Kao mjeru sličnosti ćemo koristiti korelaciju u obliku:

$$r = \frac{\sum_{m=1}^M \sum_{n=1}^N (X_{mn} - \bar{X})(Y_{mn} - \bar{Y})}{\sqrt{\sum_{m=1}^M \sum_{n=1}^N (X_{mn} - \bar{X})^2} \sqrt{\sum_{m=1}^M \sum_{n=1}^N (Y_{mn} - \bar{Y})^2}}, \quad (10.2)$$

gdje, M i N dimenzije područja na kojem se slike X i Y preklapaju, a \bar{Y} i \bar{X} označavaju njihove prosječne vrijednosti X i Y , računate kao:

$$\bar{X} = \frac{1}{NM} \sum_{m=1}^M \sum_{n=1}^N Y_{mn} \quad (10.3)$$

U ovoj vježbi koristit ćemo registraciju da bismo pronašli objekt u slici, da bismo pratili objekt u nizu slika (filmu) te da bismo odredili orijentaciju objekta na slici.

10.2.1. Zadatci

1. Zadane su dvije slike S_1 i S_2 . Slika S_1 prikazuje objekt, a slika S_2 prikazuje objekt s okolinom. Potrebno je napisati funkciju koja računa korelaciju između tih dviju slika za bilo koji pomak x i y slike S_1 . Radi jednostavnosti računajte korelaciju samo na mjestima gdje se slike u potpunosti preklapaju. Koristite funkciju `corr2()`.
2. Koristeći funkciju napisanu u prethodnom zadatku napišite funkciju koja računa korelaciju za sve položaje preklapanja slika S_1 i S_2 . Ulaz funkcije su slike S_1 i S_2 , a izlaz funkcije je matrica R čiji element $r_{i,j}$ sadrži vrijednosti korelacije slika S_1 pomaknute za (i, j) i slike S_2 .
3. Izračunajte matricu R za sliku `auto1.tiff` i sliku `slika1.tiff`. Rezultat prikažite funkcijom `mesh()`.
4. Gdje³ se nalazi maksimum (m_i, m_j) matrice R ?
5. Prikažite sliku S_2 te na njoj iscrtajte pravokutnik dimenzija `size(S1)`, na položaju (m_i, m_j) . Koristite naredbu `rectangle()`.
6. Izračunajte matricu R za sliku `auto1.tiff` i slike: `slika2.tiff`, `slika3.tiff` i `slika4.tiff`. Prikažite detekciju objekta u sceni. Interpretirajte rezultate.

³Na kojem položaju (x, y)

7. Napišite funkciju koja računa korelaciju između dvije slike, od kojih je jedna slika rotirana za kut α . Koristite naredbe `corr2()` i `imrotate()`. Ulazi funkcije su S_1 , S_2 i α , a izlaz funkcije je korelacija među slikama.
8. Izračunajte korelaciju između slika `auto1.tiff` i `auto2.tiff` za pomak $\alpha = 0 - 359^\circ$. Prikažite funkciju ovisnosti korelacije o kutu α . Za koliki α je korelacija maksimalana? Zašto?

Poglavlje 11.

Dodatci

11.0.2. Popis korištenih slika



Slika 11.1.: Na slici je prikazana spomen-medalja na Bleiburg, rad akademskog kipara Damira Mataušića. Slika i sadržaj na slici vlasništvo Hrvatskog novčarskog zavoda. Korišteno uz dozvolu Hrvatskog novčarskog zavoda.



Slika 11.2.: Slika prikazuje profesora Baltazara, najpoznatiji hrvatski crtali lik. Idejni začetnik lika profesora Baltazara je Zlatko Grgić iz Zagrebačke škole crtanog filma. Slika preuzeta s www.croatianhistory.net.



Slika 11.3.: Na slici je prikazana medalja Kamenitih vrata, rad akademskog kipara Stjepana Divkovića. Slika i sadržaj na slici vlasništvo Hrvatskog novčarskog zavoda. Korišteno uz dozvolu Hrvatskog novčarskog zavoda.



Slika 11.4.: Na slici je prikazana medalja s likom grada Dubrovnika, rad akademskog kipara Stjepana Divkovića. Slika i sadržaj na slici vlasništvo Hrvatskog novčarskog zavoda. Korišteno uz dozvolu Hrvatskog novčarskog zavoda.



Slika 11.5.: Kralj Tomislav - rad Kristiana Krekovića. Fotografija prikazuje jedan od 63 portreta Hrvatskih knezova i kraljeva. Originali portretâ nažalost nisu sačuvani. Slika preuzeta s www.croatianhistory.net.



Slika 11.6.: Stranica iz Misala po zakonu rimskoga dvora. Misal po zakonu rimskoga dvora je prva hrvatska tiskana knjiga, otisnuta svega 28 godina nakon dovršetka Gutenbergove Biblije. Tiskana je na glagoljici i predstavlja prvi misal u Europi koji nije tiskan latiničnim slovima. Očuvano je 11 primjeraka knjige. Slika preuzeta s www.croatianhistory.net.



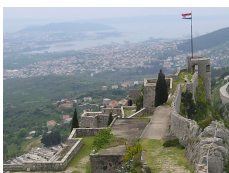
Slika 11.7.: Detalj sa slike "Sušačke dizalice" - Joze Kljakovića. Zanimljivo je primijetiti da se isti motiv pojavljuje već ranije na slici "Hrvatska privreda nakon ujedinjenja" koja je danas u vlasništvu hrvatske gospodarske komore. Slika preuzeta iz Monografije o Jozi Kljakoviću u izdanju Galerije "Klovićevi dvori". Korišteno uz dozvolu.



Slika 11.8.: Naslovna stranica knjige "Smrt Smail-age Čengića" - Ivana Mažuranića, prvog bana pučanina. Knjiga piše o ustanku crnogorskih hajduka protiv turske vlasti. Zanimljivo je primijetiti da se i Kljakovićev trag nalazi u Zaljevu hrvatskih svetaca, na slikama u crkvi sv. Eustahija. Slika preuzeta iz Monografije o Jozi Kljakoviću u izdanju Galerije "Klovićevi dvori". Korišteno uz dozvolu.



Slika 11.9.: Amfiteatar antičke Salone (današnji Solin). Slika je obiteljsko vlasništvo autora.



Slika 11.10.: Pogled na Split s vrha Kliške tvrđave. Slika je vlasništvo autora.



Slika 11.11.: Djelić mozaika "Krunjenje Zvonimira" - Joze Kljakovića. Mozaik je vlasništvo Hrvatskog zavoda svetog Jeronima u Rimu. Slika preuzeta iz Monografije o Jozi Kljakoviću u izdanju Galerije "Klovićevi dvori". Korišteno uz dozvolu.



Slika 11.12.: Središnji od tri mozaika na pročelju Hrvatskog zavoda svetog Jeronima u Rimu, nazvan "Krist knez mira" - Joze Kljakovića. Slika preuzeta iz Monografije o Jozi Kljakoviću u izdanju Galerije "Klovićevi dvori". Korišteno uz dozvolu.



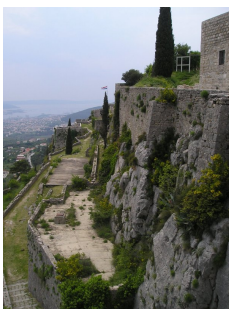
Slika 11.13.: Slika prikazuje djelić mozaika "Pokrštenje Hrvata" - Joze Kljakovića. Mozaik je vlasništvo Hrvatskog zavoda svetog Jeronima u Rimu. Slika preuzeta iz Monografije o Jozi Kljakoviću u izdanju Galerije "Klovićevi dvori". Korišteno uz dozvolu.



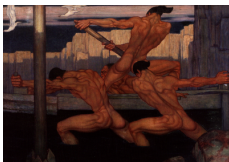
Slika 11.14.: Detalj dvorane Hrvatskog sokola. Hrvatski sokol je od 1874. centar bavljenja športom (prvenstveno gimnastike) u Zagrebu, a zanimljivost je da je upravo u zgradi Hrvatskog sokola prikazan prvi film u Zagrebu i to 1896. godine. Slika je vlasništvo autora.



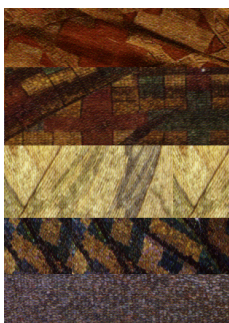
Slika 11.15.: Detalj Kliške tvrđave. U 9. stoljeću Klis je bio sjedište hrvatskih knezova i kraljeva Trpimirovića, odnosno hrvatskim glavnim gradom. Slika je vlasništvo autora.



Slika 11.16.: Detalj Kliške tvrđave. Kliška tvrđava predstavlja jednu od najistaknutijih utvrda u Hrvatskoj, bila je sjedište Hrvatskih knezova i kraljeva. Ipak, najpoznatija po Petru Kružiću i uskocima koji su se desetljećima odupirali pokušajima turskog osvajanja tvrđave. Slika je vlasništvo autora.



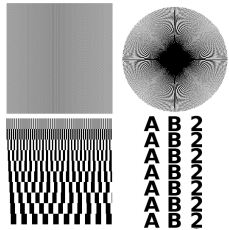
Slika 11.17.: Nakon pada Klisa 1537. godine, većina Uskoka seli u Senj, gdje postaju poznati po svom gusarenju Mletačkih brodova. Nakon mira između Austrije i Venecije, sele u okolicu Karlovca gdje s ranije doseljenim Uskocima nastanjenim u Žumberačkom gorju počinju činiti jezgru Vojne krajine. Slika preuzeta iz Monografije o Jozi Kljakoviću u izdanju Galerije "Klovićevi dvori". Korišteno uz dozvolu.



Slika 11.18.: Različite teksture uzete iz mozaikâ "Krunjenje Zvonimira", "Krist knez mira", "pokrštenje Hrvata" te freske iz kapele sv. Fabijana i sv. Sebastijana nazvane "Trijumf slavenskog bogoslužja (Splitski sabor)" Joze Kljakovića. Detalji preuzeti iz Monografije o Jozi Kljakoviću u izdanju Galerije "Klovićevi dvori". Korišteno uz dozvolu.



Slika 11.19.: Primjer ponavljajućih uzoraka u stvarnom svijetu. Slika je vlasništvo autora.



Slika 11.20.: Testni uzorak. Iscrtan Octave kodom ili Inkscape-om. Vlasništvo autora.

11.0.3. Napomene o pravopisu

U ovom djelu slijedili smo Babić-Finka-Moguš Pravopis Hrvatskog jezika u izdanju Školske Knjige. Važno je napomenuti:

- Piše se *obradba*, a ne *obrada*. Jednako kao i riječi *prosidba*, *kosidba*, *selidba*, odnosno *rašćlanba*. Kao riječ koja se često griješi, navedena je na 295. str. Pravopisa.
- Čestica *ne* se piše rastavljeno od glagola. Zato se piše *ne ću*, a ne *neću*. To je navedeno pod pravilom 305.
- Prema pravilu 201 ispravno je *zadatci* i *dodatci*, a ne *zadaci* ili *dodaci*, jer se *t* ne gubi ispred *c*, osim u riječima *otac*, *sudac*, *svetac*.

Sve ostale primjedbe slobodno pošaljite asistentu elektroničkom poštom na hrvoje.kalinic@fer.hr.

11.0.4. Zahvale

Ovim putem želimo zahvaliti:

- Borisu Zaninoviću i Kristini Višnić iz Hrvatskog novčarskog zavoda, na dobroj volji i ustupanju fotografija
- Petri Senjanović iz Galerije "Klovićevi Dvori" na dozvoli korištenja fotografija objavljenih u Monografiji Joze Kljakovića
- Prof. dr. sc. Darku Žubriniću, na dozvoli korištenja fotografija iz njegove zbirke sa stranice www.croatianhistory.net

Također, zahvaljujemo svim studentima koji su svojim pitanjima i komentarima omogućili da ove upute za laboratorijske vježbe budu jasnije i kvalitetnije.