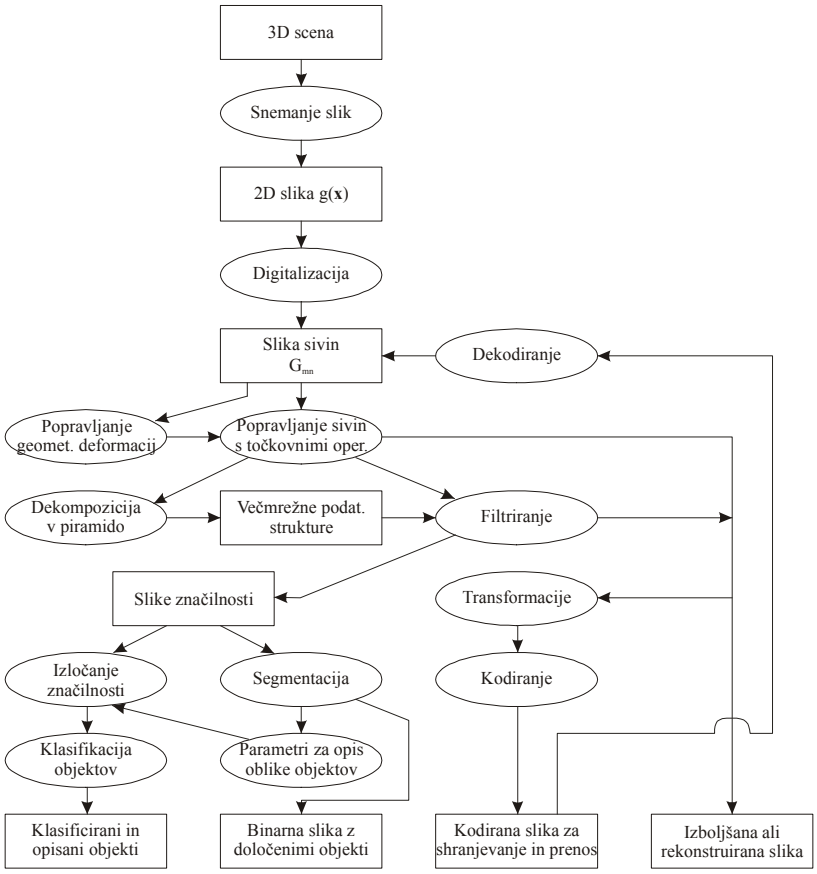
5 DIGITALNA OBDELAVA SLIK

dr. Božidar Potočnik
Fakulteta za elektrotehniko, računalništvo in informatiko
Smetanova 17, 2000 Maribor
e-mail: bozo.potocnik@uni-mb.si
tel. +386 2 220 7484

Obdelava slik z računalnikom že dolga leta ni več samo domena vojaških sistemov, ampak jo najdemo že na mnogih civilnih področjih. Digitalna obdelava slik se veliko uporablja v meteorologiji kot pomoč za napovedovanje vremena, v kartografiji, pri analizi slik, ki jih pošiljajo sateliti iz vesolja, geologiji in v raznih vejah medicine, v zadnjem času pa tudi na športnem področju, predvsem kot pripomoček za optimiziranje športnikovih gibov in reakcij.

Metode in postopki za digitalno obdelavo ene slike so razviti že vrsto let. V zadnjih letih se pojavljajo zgolj izboljšave že obstoječih postopkov. V okviru teh predavanj si bomo pogledali potek digitalne obdelave ene slike. Pri opisu te obdelave se bomo zgledovali po sliki (*Slika 1*).



Slika 1: Hierarhija nalog pri digitalni obdelavi slik

Prva operacija v hierarhiji operacij za digitalno obdelavo slik je snemanje oz. pridobivanje slik. Tukaj se bomo seznanili z raznimi vrstami pridobivanja slik, jedro pa bo posvečeno sistemom PC za obdelavo slik. Po snemanju dobimo digitalno sliko (digitalizacija je ponavadi že kar vključena v sistem za obdelavo slik). Naslednji korak v digitalni obdelavi slik je priprava oz. predobdelava slike za kasnejšo analizo vsebine v sliki. Predobdelava slik je namenjena predvsem za poudarjanje določenih značilnosti v sliki. V tem delu naloge bomo spoznali točkovne operacije, metode za popravljanje geometrijskih deformacij, ki so nastale zaradi kamere ali nenatančnosti pri snemanju, na koncu pa se bomo seznanili s filtriranjem slik. Rezultat predobdelave slik je lahko binarna slika ali slika, v kateri so poudarjeni robovi ali vogali objektov, ali slika, v kateri je odstranjen šum itd., v splošnem pa je to slika, ki nam olajša nadaljnje delo - analizo slike (določanje objektov itd.). Segmentacija slike je naslednja operacija pri digitalni obdelavi slik. S segmentacijo skušamo določiti tista področja v sliki, ki kar najverjetneje predstavljajo objekte. V okviru segmentacije se bomo seznanili, kako lahko objekt zaznamo v sliki ter kako ga lahko popišemo z njegovo mejo oz. regijo. S segmentacijo, s prepoznanimi (določenimi) objekti v sliki, se lahko proces digitalne obdelave slik že tudi konča. Nekateri realni problemi pa zahtevajo tudi, da primerjamo objekte med seboj. Zato si bomo pogledali tudi, s kakšnimi parametri lahko najbolj ustrezno popišemo objekt. Glavni razlog za opis objektov s

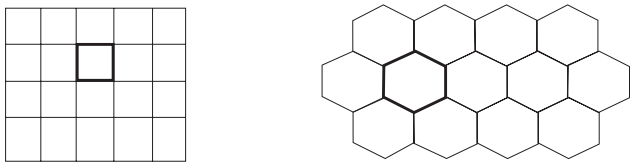
parametri je zmanjšanje količine podatkov, ki jih je potrebno obdelati. Na koncu celotne hierarhije postopkov za digitalno obdelavo ene slike si bomo ogledali še klasifikacijo. Klasifikacija je postopek, s katerim primerjamo objekte med seboj in jih zatem združujemo v razrede na osnovi podobnosti.

5.1 DIGITALNE SLIKE

Kaj je slika? Pod pojmom slika razumemo kakršenkoli posnetek opazovane scene oziroma okolja, npr. slika na očesni mrežnici ali slika na televiziji, umetniška slika, fotografije, ultrazvočne slike, sintetične slike itd. Takšna definicija slike je za računalniško obdelavo preveč splošna, zato jo v praksi zožimo ter sliko zapišemo matematično.

Sliko matematično modeliramo z zvezno funkcijo dveh neodvisnih spremenljivk. Označimo jo z oznako *f(x,y)*, kjer predstavljata spremenljivki *x* in *y* koordinati v ravnini. Vrednost slikovne funkcije *f(x,y)* največkrat pomeni **svetlost** *(angl. brightness)* v točki *(x,y)*. Pomen slikovne funkcije lahko interpretiramo tudi na druge načine, npr. kot nekatere fizikalne veličine: temperaturo, porazdelitev tlaka, razdalja od opazovalca itd.

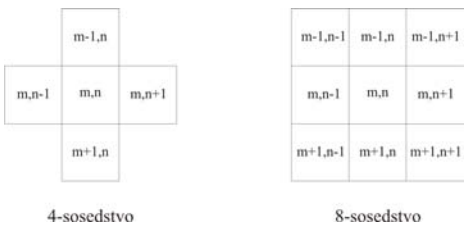
Z zgoraj omenjeno predstavitvijo popolnoma popišemo sliko. Kot vemo, je računalnik digitalna naprava, ki operira z digitalnimi podatki (biti), zato pri računalniški obdelavi slik potrebujemo digitalne in ne zvezne slike. **Digitalna slika** *(angl. digital image)* je diskretno predstavljena slika. Digitalno sliko največkrat predstavimo kot 2D mrežo (matriko) in jo označimo z *f(i,j)*, pri čemer indeks vrstic *i* teče od *0* pa do *M-1*, indeks stolpcev *j* pa od *0* do *N-1*, kjer je *M* število vrstic in *N* število stolpcev v mreži, vrednost funkcije *f(i,j)* pa je celo število in predstavlja sivino (nivo sivine). V praksi srečamo dve možni obliki mreže: pravokotno in šesterokotno mrežo (*Slika 2*).



(a) (b)

Slika 2: Predstavitev digitalne slike s pravokotno in šesterokotno mrežo, z debelo je označen piksel

Osnovni gradnik 2D digitalne slike je **piksel** *(angl. pixel)*. To je najmanjša enota na 2D mreži, nad katero izvajamo razne operacije (spreminjamo njegovo vsebino). Posamezen piksel je popolnoma določen z indeksom vrstice *i* in z indeksom stolpca *j*. Torej, točka *(i,j)* predstavlja piksel v i-ti vrstici ter j-tem stolpcu, vrednost piksla na tem mestu pa označujemo z *f(i,j)*. Obe obliki mreže prideta do izraza, kadar opazujemo okolico določenega piksla. V pravokotni mreži lahko definiramo sosednje piksle na dva načina. Piksel imenujemo sosednji piksel trenutnega piksla, kadar imata en stikajoč se rob - 4-sosedstvo, ali pa kadar imasta skupen vsaj en stikajoč se vogal - 8- sosedstvo (*Slika 3*). Šesterokotna mreža omogoča zgolj 6-sosedstvo. Zgoraj navedene definicije pridejo do izraza ob študiju sosedstva objektov v sliki.



Slika 3: Sosedstvo piksla (m,n) v pravokotni mreži

5.2 ZAJEMANJE IN SHRANJEVANJE SLIK

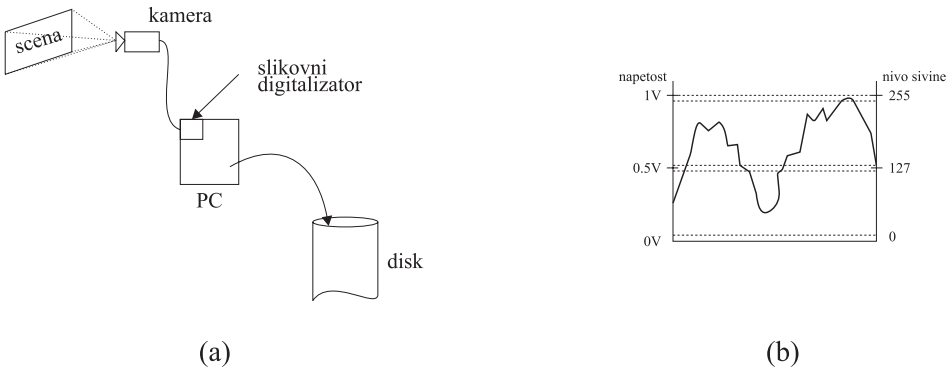
Digitalna slika je diskretna struktura, ki jo največkrat predstavimo z matriko celih števil. Prvi način pridobivanja slik je **sintetičen način**, kjer sami poljubno generiramo vrednosti pikslov slikovne matrike. Če želimo, da imajo te vrednosti tudi kakšen realen pomen (npr. siva hiša na belem ozadju), je dobro, če si pri sintetičnem načinu generiranja slik pomagamo z računalniškimi orodji. Tako lahko slike generiramo s pomočjo raznih računalniških programov za risanje, npr. programi za okolje MS-WINDOWS, kot so Paintbrush, Corel Photo-Paint, Paint Shop PRO, Borlandov C++ Workshop itd.

Drugi način pridobivanja oz. zajemanja slik je s pomočjo **prebirnika** (*angl. scanner*). Prebirnik je naprava, ki preslika sliko s papirja vrstico za vrstico v računalnik. S pomočjo prebiranja lahko dobimo digitalne slike iz fotografij, slik v knjigah in revijah, ročno risanih slik itd.

Tretji način pridobivanja slik je snemanje slik **direktno iz naprav**, ki so priključene na računalnik preko ustrezne video kartice. Takšne naprave so npr. kamere, video rekorderji, ultrazvočne naprave, naprave CT *(angl. computed tomography)* itd.

Večina sistemov za pridobivanje slik, tudi bolj sofisticirani sistemi za obdelavo slik, deluje na osnovi **CCD-naprav** *(angl. Charge Coupled Device)*. CCD-naprava je sestavljena iz polja elementov, ki hranijo energijo, transportnega mehanizma in izhodne naprave. Primer CCD-naprave je CCD-kamera (npr. video kamera). Elementi, ki hranijo energijo, so urejeni linearno (CCD-naprave, ki prebirajo po vrsticah) ali pa kot matrična struktura. Vsak element, ki hrani energijo, lahko pretvori energijo svetlobe na svoji površini v električno napetost. Transportni mehanizem zatem to električno napetost iz vsakega elementa poda na izhod CCD-naprave. Izhod iz CCD-naprave je analogni video signal, ki je vhod v slikovni digitalizator. Slikovni digitalizator zatem ta analogni video signal digitalizira, in sicer tipično z ločljivostjo 8 bitov in s hitrostjo vsaj 10 milijonov pikslov na sekundo. Digitalizacija pomeni, da se vsaka analogna vrednost posameznega elementa (vrednost je podana v obliki električne napetosti), pretvori v eno izmed 256 sivin (8 bitov nam lahko namreč določa 2^8=256 različnih vrednosti sivin). Ta pretvorjena vrednost pa dejansko pomeni vrednost piksla v digitalni sliki (informacijo, za kateri piksel je pripadajoča vrednost, nam poda slikovni digitalizator). Na podoben način pridobivamo tudi barvne slike, le da moramo v tem primeru analogni video signal razbiti na osnovne tri gradnike: rdečo, zeleno in modro komponento barve, vsako komponento pa zatem v slikovnem digitalizatorju digitalizirati po zgoraj opisanem postopku. *Slika 4* nam shematsko prikazuje postopek zajemanja digitalnih slik.

Velikost digitalizirane slike je standardizirana. Standardna digitalna slika meri v evropskem formatu 512x512 pikslov, v ameriškem formatu pa 480x512 pikslov. Mnogi novejši slikovni digitalizatorji so mnogo fleksibilnejši, saj je pri njih možna nastavitev poljubnega števila vrstic in stolpcev slike ter pri nekaterih sistemih tudi frekvenca. S takšnimi sistemi lahko digitaliziramo tudi nestandardne video signale, kot npr. signale iz elektronskega mikroskopa, ultrasoničnih in termičnih senzorjev, visokoresolucijskih CCD-kamer z resolucijo 1000x1000 pikslov in večjo.



Slika 4: Shema snemanja slik: a) temeljni koraki pri snemanju slik: sceno posnamemo s CCD-napravo, jo s pomočjo slikovnega digitalizatorja včitamo v računalnik in shranimo; b) primer digitalizacije video signala: napetost video signala se pretvori v nivo sivine.

5.2.1 Računalniški formati za shranjevanje slik

Pri vseh metodah pridobivanja slik je treba vedeti, v kakšnem podatkovnem formatu nam posamezno računalniško orodje shrani sliko na trdi disk. Vsem formatom za shranjevanje slik je skupno, da imajo na začetku datoteke, kar imenujemo tudi glava, zapisane podatke o številu pikslov v vrstici in številu pikslov v stolpcu digitalne slike, zatem pa sledijo zapisi, ki nosijo informacijo o vrednosti (sivin ali barv) pikslov po vrsticah od prve do zadnje vrstice. Seveda večina računalniških formatov za shranjevanje slik, ki so danes v uporabi, ni tako preprostih, saj imajo na začetku zapleteno glavo in šele nato sledijo zlogi z informacijami o sliki, ki so pa prav tako na nek način kodirani.

Zgled: Kot zgled si poglejmo format za shranjevanje slik **BMP** (*angl BitMap*). To je format, ki ga uporabljajo aplikacije v okolju MS-WINDOWS. Datoteka, v katero shranimo digitalno sliko v formatu BMP (ponavadi ima takšna datoteka končnico .BMP), je sestavljena iz treh delov, in sicer iz glave, informacijskega dela ter iz podatkovnega dela. Ti trije deli si znotraj datoteke zaporedno sledijo. *Slika 5* podrobno opisuje format BMP. V glavi bitne slike so podatki o dolžini datoteke ter o odmiku do podatkovnega dela. Po koncu glave se takoj začne informacijski del, kjer najdemo podatke o velikosti bitne slike, horizontalni in vertikalni ločljivosti, tipu kompresije (podatki o pikslih so lahko tudi kompresirani) itd. Na koncu informacijskega dela je barvna paleta. Vsak vstop v barvno paleto določa eno barvo. Če imamo opravka s sivinami, potem so vse tri vrednosti pri posameznem vstopu enake (rdeča, zelena in modra), sicer se pa razlikujejo. V podatkovnem delu pa so zaporedno shranjeni podatki o vrednostih pikslov.

Glava bitne slike

Zlog	Podatek	Opis
1-2	Tip datoteke	V ASCII zapisano ‘BM’
3-6	Dolžina datoteke	V dvojnih besedah (32 bitov)
7-10	Rezervirano	Mora biti 0
11-14	Odmik do podatkovnega dela	Odmik je merjen od začetka datoteke

Informacijski del bitne slike

Zlog	Podatek	Opis
1-4	Število zlogov v informacijskem delu	Trenutno 40 zlogov
5-8	Širina bitne slike	V pikslih
9-12	Višina bitne slike	V pikslih
13-14	Število barvnih ravnin	Mora biti 1
15-16	Število bitov na piksel	Možnosti so: 1, 4, 8, 24

17-20	Tip kompresije podatkov	0: brez kompresije; 1: tekoča dolžina (8 bitov na piksel); 2: tekoča dolžina (4 biti na piksel)
21-24	Velikost slike	V zlogih
25-28	Horizontalna ločljivost	V pikslih na meter
29-32	Vertikalna ločljivost	V pikslih na meter
33-36	Število indeksov barv, ki jih uporablja bitna slika	0 pomeni, da so vse barve pomembne
37-40	Število indeksov barv, ki so pomembni za prikaz bitne slike	0 pomeni, da so vse barve pomembne
41	Vrednost modre barve	Začetek barvne palete (vstop 0)
42	Vrednost zelene barve	
43	Vrednost rdeče barve	
44	Rezervirano	Mora biti 0
…	…	4 zlogi za vsak vstop v barvno paleto. Barve naj bodo razvrščene po pomembnosti.
	Preostali vstopi v barvno paleto	

Podatkovni del bitne slike

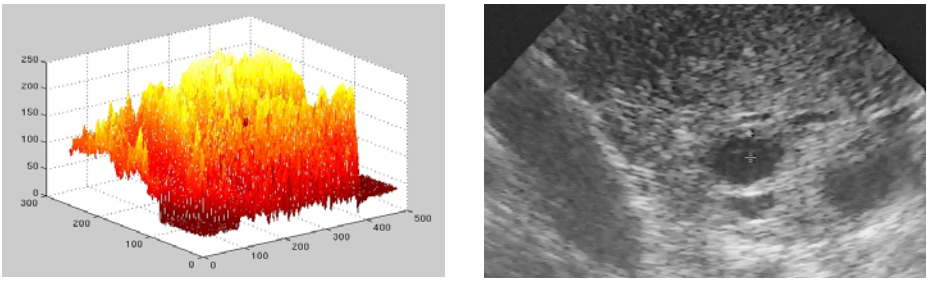
Zlog	Podatke	Opis
1	Vrednost piksla	Piksli se shranjujejo vrstico za vrstico, od leve proti desni znotraj vsake vrstice. Vrstice se shranjujejo od dna proti vrhu.
…	….	Začetek bitne slike, piksel (0,0), je
	Preostali piksli.	v spodnjem levem kotu.

Slika 5: Opis formata za shranjevanje slik BMP. Format je sestavljen iz treh delov: glave, informacijskega dela ter podatkovnega dela.

Iz pravkar opisanega zgleda smo dobili predstavo o dejanski zgradbi formatov za shranjevanje slik. Ob formatu BMP najdemo danes še mnogo drugih formatov, kot so npr. format GIF (Graphics Interchange Format), JPEG (Joint Photographic Experts Group), EPS (Encapsulated PostScript), TIFF (Tag Image File Format), če naštejemo le nekatere najbolj znane.

5.3 VIZUALIZACIJA SLIK

Videli smo, da digitalne slike največkrat predstavimo kot 2D matrike celih števil. Takšne matrike pa zatem na nek način interpretiramo. *Slika 6* nam kaže dve različni interpretaciji iste digitalne slike.

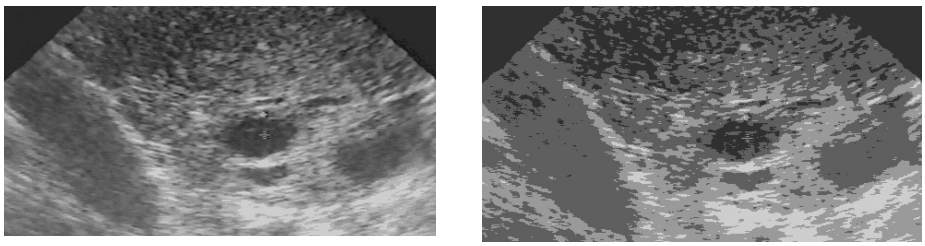


(a) (b)

Slika 6: Dve različni interpretaciji iste digitalne slike. Sliki, velikosti 256x478 pikslov, prikazujeta jajčnik z 6 jajčnimi mešički.

Interpretacija slike, ki je prikazana na levi (*Slika 6.a*), je neobičajna, saj s takšne predstavitve le težko zajamemo informacijo oziroma vsebino, ki jo nosi slika. Desna predstavitev slike nam je seveda veliko bližja, saj z nje enostavno razberemo informacijo, ki jo nosi. Na tej sliki vidimo ultrazvočni posnetek jajčnika (v sredini), znotraj katerega se nahaja 6 jajčnih mešičkov, na desni vidimo krvno žilo, zgoraj pa maternično sluznico.

Pri takšni predstavitvi slike smo vsaki vrednosti piksla priredili eno sivino. Število nivojev sivine, ki jih lahko ima posamezen piksel, nam določa **kontrastno ločljivost slike** *(angl. contrast resolution)*. Pri digitalni obdelavi slik imamo tipično 256 nivojev sivin (toliko sivin lahko človeško oko še enostavno razlikuje), pri tem vrednost 0 pomeni črno barvo, vrednost 255 belo, vrednost 127 sivo, vrednosti od 0 do 126 so temne sivine urejene od najtemnejših sivin do sive barve, vrednosti od 128 do 255 pa so svetle sivine urejene od sive do bele barve. V slikah lahko imamo seveda tudi več ali pa manj kot 256 različnih sivin (npr. 16 nivojev), paziti moramo le, da vrednost 0 ustreza črni barvi, maksimalna vrednost pa beli barvi (Maksimalna vrednost je za ena manjša od števila vseh možnih sivin, saj začnemo šteti z 0!). *Slika 7* prikazuje primer, kjer smo isto sliko (*Slika 6*) prikazali z različnim številom sivin. Lepo lahko vidimo, da se ob zmanjševanju kontrastne ločljivosti (zmanjševanju števila nivojev sivin), izgubljajo informacije v sliki, predvsem detajli. Pri digitalni obdelavi slik si želimo čim večjo kontrastno ločljivost, saj s tem postane obdelava slik lažja, hkrati pa je tudi lažje ocenjevati dobljene rezultate. To še posebej velja za medicinske slike.



(a) (b)

Slika 7: Slika 6 predstavljena z a) 16 nivoji sivin in b) 4 nivoji sivin.

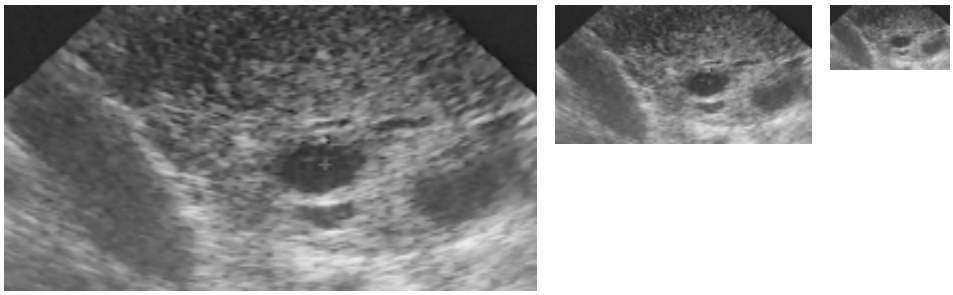
Ko prikazujemo digitalno sliko pa ni pomembna le njena kontrastna ločljivost, ampak tudi kontrastna ločljivost grafičnega prikazovalnika (npr. monitorja). Če ima grafični prikazovalnik večjo ločljivost, kot je število različnih sivin v sliki, potem lahko vrednosti sivin v sliki tako preoblikujemo – raztegnemo, da po transformaciji pokrivajo celoten pas sivin, ki jih je prikazovalnik zmožen prikazovati. Temu postopku transformiranja sivin pravimo **linearizacija** *(angl. linearization)*. Če z *min* označimo najmanjšo sivino, ki jo najdemo v digitalni sliki, z *max* pa največjo sivino, in če je prikazovalnik zmožen prikazati Q različnih sivin, potem linearizacijo sivine g opravimo po naslednjem postopku:

$$g' = \frac{Q}{\max - \min}(g - \min),$$

kjer smo z g' označili sivino po transformaciji. Celotno digitalno sliko pa linearizirамo piksel za pikslom tako, da vrednost (sivino) vsakega piksla transformiramo po zgoraj opisanem postopku. Linearizacijo lahko seveda opravimo tudi, če je število sivin v sliki večje od števila sivin, ki jih je prikazovalnik zmožen prikazati, vendar se moramo zavedati, da se ob tem zmanjša kontrastna ločljivost slike.

Ob kontrastni ločljivosti digitalnih slik pa je izredno pomembna tudi **prostorska ločljivost slike** *(angl. space resolution)*. Prostorsko ločljivost ponavadi podajamo v pikslih na enoto dolžine (npr. pikslov na cm). Prostorska ločljivost je tesno povezana z velikostjo slik. Digitalne slike so lahko v splošnem poljubnih velikosti (dimenzij), v praksi pa so sodih ali celo dimenzij, ki so potence števila 2 (npr. 256x256 pikslov, prvo število pomeni število vrstic, drugo pa število stolpcev). Velikost slike je

pomembna za analizo slike, predvsem pri opazovanju majhnih predmetov oziroma detajlov v slikah. *Slika 8* nam kot zgled prikazuje isto sliko, ki smo jo prikazali v različnih dimenzijah. Opazimo lahko, da se z zmanjševanjem dimenzije (prostorske ločljivosti), ohranijo v sliki le grobi obrisi največjih objektov, medtem ko se detajli izgubijo.



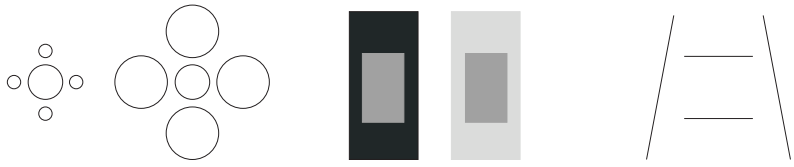
(a) (b) (c)

Slika 8: Slika 6 prikazana v velikosti a) 128x239 pikslov, b) 64x119 pikslov, c) 32x60 pikslov.

5.3.1 Človeško zaznavanje

Preden se lotimo obdelavo digitalnih slik, ne smemo mimo principov človeškega zaznavanja slik. Pri analizi slik moramo informacijo o sliki izraziti s spremenljivkami, ki jih človek z lahkoto zazna, to so npr. psihofizični parametri, kot so kontrast, meja, oblika, vzorec, barva itd. Človek bo zaznal objekt v sliki le, če ga lahko dovolj enostavno loči od ozadja. Celoten problem bi bil zelo enostaven, če bi človeški vizualni sistem imel linearni odziv na sestavljen vhodni dražljaj. Vendar ni tako. Občutljivost človeških čutil je logaritemsko proporcionalna intenzivnosti vhodnega signala (npr. svetlosti).

Kontrast *(angl. contrast)* je lokalna sprememba v svetlosti in je definirana kot razmerje med povprečno svetlostjo objekta in svetlostjo ozadja. Človeško oko je logaritemsko občutljivo na svetlost, kar pomeni, da je za isto zaznavanje večje svetlosti potreben večji kontrast. Hkrati pa je svetlost, ki jo zazna človeško oko, zelo odvisna od lokalnega ozadja opazovanega objekta. Ta efekt imenujemo pogojni kontrast, povzroča pa razne vizualne fenomene oziroma iluzije. *Slika 9* prikazuje nekaj vizualnih efektov. **Ostrina** *(angl. acuity)* je sposobnost za zaznavanje detajlov v sliki. Človeško oko je manj občutljivo na počasne in hitre spremembe v svetlosti, bolj pa je občutljivo na srednje hitre spremembe. Pomemben parameter je tudi **meja objektov** *(angl. object border)*, ki nosi veliko informacije ter omogoča lažje zaznavanje.



(a) (b) (c)

Slika 9: Vizualni efekti: a) Ebbinghausova iluzija: oba majhna kroga v centru slike izgledata kot da sta različnih premerov, čeprav imata enake; b) pogojni kontrast: levi pravokotnik na svetlem ozadju izgleda temnejši, čeprav imata oba pravokotnika isto sivino; c) zgornja daljica (v ozadju) izgleda daljša kot spodnja daljica (od spredaj), čeprav sta obe enako dolgi.

Barve so naslednji parameter, s katerim lahko uravnavamo človeško zaznavanje slik, saj je znano, da je človeško oko bolj občutljivo na barve kot pa na svetlost pri normalni osvetlitvi. Barvo lahko izrazimo kot kombinacijo rdeče, zelene in modre komponente – **barvni model RGB** *(angl. RGB color model)*. V primeru slik sivin je vrednost piksla celo število, ki predstavlja svetlost za dano točko, v primeru barvnih slik pa je piksel vektor treh celih števil, ki določa barvo za dano točko, in sicer vsako

število od te trojice določa svetlost za eno komponento barve: rdečo, zeleno in modro. Tako npr. vektor (0, 0, 0) določa črno barvo, vektor (255, 255, 255) belo barvo, vektor (255, 0, 0) rdečo, vektor (0, 255, 0) zeleno barvo itd. Če lahko posamezna komponenta barve zavzame 256 različnih vrednosti (kar je tipično pri slikah sivin), potem lahko določimo 256^3 = 16 777 216 različnih barv. Tudi pri barvnem zaznavanju se lahko pojavijo podobne iluzije kot pri drugih psihofizičnih kvantitetah.

5.4 PREDOBDELAVA DIGITALNIH SLIK

V dosedanjih podpoglavjih smo podrobno opisali načine pridobivanja digitalnih slik in njihovo vizualizacijo. V tem podpoglavju pa se bomo srečali z osnovnimi koraki obdelovanja digitalnih slik. Digitalne slike, ki smo jo dobili po snemanju, lahko seveda takoj shranimo na disk in jih kot take, lahko po želji prikazujemo. Vendar namen digitalnega obdelovanja slik ni le zajemanje in prikazovanje slik, ampak tudi dodajanje raznih informacij na sliko (npr. datum pregleda pacienta), spreminjanje kontrastov, avtomatsko iskanje objektov v slikah itd. V grobem lahko samo obdelovanje digitalnih slik razdelimo na več skupin:

- predobdelavo, kjer odpravljamo določene degradacije v slikah (npr. šum) ali poudarjamo posamezne lastnosti slik (npr. poudarimo robove, določene sivine),
- segmentacijo, kjer združujemo v regije glede na neko lastnost tiste piksle, ki verjetno tvorijo isti objekt,
- klasifikacijo, kjer posamezno regijo uvrstimo v določen razred objektov,
- obdelovanje 3D digitalnih slik (3D digitalna slika je zaporedje večih 2D digitalnih slik).

Operacije na digitalnih slikah, ki se izvajajo na najnižjem nivoju abstrakcije slike (pikslov), imenujemo **predobdelava slik** *(angl. image preprocessing)*. Cilj predobdelave slik je izboljšati podatke (vrednosti nekaterih pikslov) v sliki, ki so se na kakršenkoli način poslabšali zaradi nezaželenih deformacij, ali pa poudariti določene značilnosti slike (npr. robove), ki so pomembne za nadaljnjo obdelavo. Predobdelava je usmerjena predvsem na popravljanje določenih degradacij v sliki (npr. uravnavanje razlik osvetlitve, ki so nastale zaradi neenakomerne občutljivosti senzorjev kamere, korekcija oz. optimizacija svetlosti in kontrasta slike) in je zato zelo pomembno, koliko vnaprejšnjega znanja o sliki in degradaciji je na voljo. Obstaja cela vrsta različnih metod za predobdelavo slik, v nadaljevanju pa bomo predstavili tri najpomembnejše skupine teh metod, in sicer:

- spreminjanje kontrasta,
- lokalne operacije oziroma filtriranje, in
- slikovno matematiko.

5.4.1 Spreminjanje kontrasta

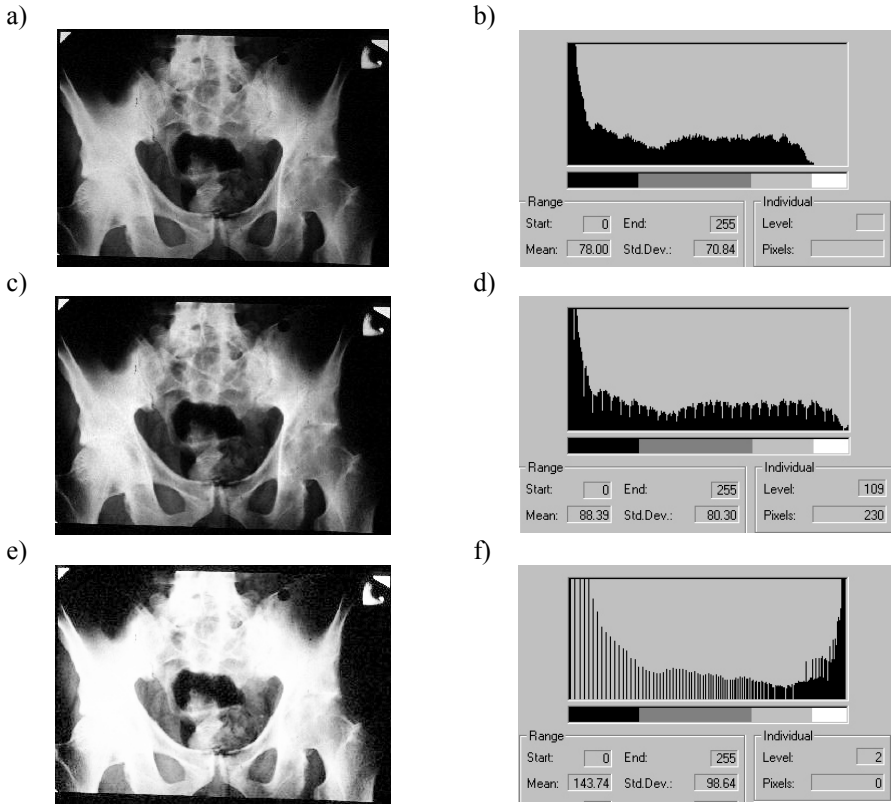
Te operacije imenujemo tudi točkovne operacije oz. operacije piksel-v-piksel. Vedeti moramo, da rezultat točkovnih operacij ni odvisen od vrednosti sivin sosednjih pikslov. Točkovna operacija preslika množico vseh vrednosti sivin samo vase. Za točkovno operacijo ne obstaja inverzna operacija.

V slikah spreminjamo kontraste, kadar smo pri zajemanju imeli slabe osvetlitvene pogoje (npr. neenakomerna osvetlitev) in se zato detajli v slikah zelo slabo razločijo. V podpoglavju 5.3 smo že opisali linearizacijo, ki je primer funkcije, s katero lahko spreminjamo kontrast. Linearizacija nam raztegne ozek pas sivin v sliki, na celoten pas sivin, ki jih je zmožen prikazati prikazovalnik. Kontrast lahko spreminjamo še na mnogo drugih načinov, pri tem pa si najpogosteje pomagamo z analizo porazdelitev sivin v slikah. Najboljši pregled nad tem, kako so v sliki porazdeljene sivine, dobimo iz histograma sivin. **Histogram sivin** *(angl. brightness histogram)* je vektor, ki ima toliko elementov, kot je kvantizacijskih nivojev oziroma nivojev sivin v sliki. Vsak element histograma hrani število pikslov, katerih vrednost sivine je enaka indeksu tega elementa. To pomeni, da npr. v polju histograma

pri indeksu 64 hranimo število pikslov, ki imajo sivino enako 64. Histogram ponavadi prikazujemo s stolpičnim grafom. Histogram *h* izračunamo po naslednjem postopku:

1. Priredi vrednost 0 vsem elementom histograma *h*.
2. Za vsak piksel *(i,j)* slike *f* povečaj element histograma *h(f(i,j))* za ena (vrednost sivine *f(i,j)* nam določa indeks v histogram).

Slika 10 nam prikazuje histograme sivin. Na levi strani vidimo digitalne slike sivin, kjer smo originalni rentgenski sliki medenice (a) na različne načine spreminjali kontrast. Na desni pa so prikazani pripadajoči histogrami sivin. Histogram je prikazan kot stolpični graf, kjer je število pikslov za sivino 0 prikazano na skrajni levi strani histograma, število pikslov za belo (sivina 255) pa na skrajni desni. V histogramu je prikazana tudi povprečna vrednost sivine (polje "Mean").



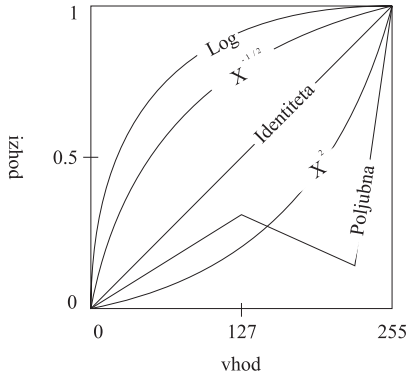
Slika 10: Histogrami sivin: a) originalna rentgenska slika medenice, b) histogram sivin za sliko a), c) originalna slika po izenačitvi histograma, d) histogram sivin za c), e) originalni sliki spremenjen kontrast po logaritemski funkciji, f) histogram sivin za e).

Izenačitev histograma *(angl. histogram equalization)* je rutina za avtomatsko spreminjanje kontrasta v slikah. Glavni cilj je, da dobimo sliko z enakomerno porazdeljenimi sivinami skozi celoten pas vseh možnih sivin. Izenačitev histograma izboljša kontrast za sivine, ki so blizu maksimumov histogramov ter zmanjša kontrast blizu minimumov. Za vsako vrednost sivine g v originalni sliki (in njenem histogramu), določimo novo vrednost sivine g' po naslednji enačbi:

$$g' = Q\sum_{i=0}^{g}\frac{h(i)}{M\ N}\ ,$$

kjer sta M in N število vrstic oziroma število stolpcev v sliki, h je histogram sivin, Q pa je število kvantizacijskih nivojev (npr. 256 nivojev).

Kontraste oziroma sivine lahko spreminjamo še po raznih drugih funkcijah ali celo popolnoma poljubno. Pri tem si pomagamo s t.i. **prenosnimi funkcijami sivin** *(angl. gray value transfer function)*. *Slika 11* prikazuje diagram z nekaj tipičnimi prenosnimi funkcijami sivin. V diagramu nanašamo na abscisno os vhodne vrednosti sivin (v našem primeru 255), na ordinatno os pa izhodne vrednosti, ki se gibljejo v intervalu od [0,1]. Ker se izhodne vrednosti sivin gibljejo med 0 in 1, jih moramo še ustrezno skalirati oziroma pomnožiti s številom kvantizacijskih nivojev Q, da dobimo pravo izhodno vrednost sivine. Kako opravimo transformacijo sivine s pomočjo prenosne funkcije sivin? Na abscisi poiščemo vrednost sivine, ki jo želimo transformirat, zatem potegnemo pravokotno črto skozi to sivino in ordinata točke, kjer se sekata pravokotnica in prenosna funkcija, nam določa izhodno vrednost, ki jo moramo le še množiti s Q. Prenosna funkcija sivin nam je v veliko pomoč pri lažjem vizualnem predstavljanju transformacij, izhodne vrednosti sivin pa moramo še vedno izračunati analitično (idealno je, če lahko prenosno funkcijo analitično popišemo). *Slika 11* prikazuje več različnih, analitično opisnih prenosnih funkcij sivin, in sicer identiteto, funkcijo kvadratni koren, kvadratno funkcijo, logaritemsko in poljubno funkcijo (npr. prenosna funkcija za kvadratni koren je oblike $1/16\sqrt{g}$). Če poteka prenosna funkcija nad identiteto to pomeni, da bodo vrednosti sivin po transformaciji svetlejše, če pa funkcija poteka pod identiteto, potem pa bodo vrednosti sivin temnejše.



Slika 11: Različne prenosne funkcije sivin.

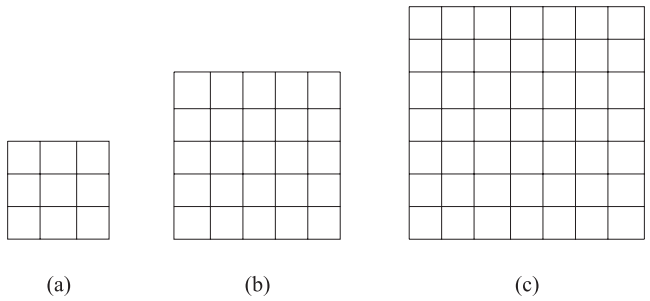
Ker imamo v digitalnih slikah opravka le z nekaj različnimi sivinami (npr. 256), lahko novo izračunane sivine shranimo v t.i. **tabele LUT** *(angl. Look-up table)*. To pomeni, da novo vrednost za npr. sivino 99 shranimo v tabelo LUT pri indeksu 99. Postopek transformiranja sivin je po uvedbi tabel LUT naslednji: za vsako možno sivino izračunamo njeno novo vrednost, to vrednost pa shranimo na ustrezno mesto v tabeli LUT. Šele zatem transformiramo vsak piksel v sliki, in sicer nam vrednost (sivina) tega piksla določa vstop v tabelo LUT, od koder odčitamo novo vrednost, to vrednost pa zatem priredimo pikslu. Na primer piksel na mestu *(45,200)* ima sivino 157, kar pove, da moramo pogledati pri indeksu 157 v tabeli LUT, od tu pa odčitamo novo vrednost, ki jo zatem priredimo temu pikslu, torej *f(45,200)* = LUT(157).

5.4.2 Lokalne operacije (filtriranje)

Pri spreminjanju kontrasta (točkovnih operacijah) smo novo vrednost piksla po nekem pravilu določili le iz njegove stare vrednosti sivine. Pri lokalnih operacijah nad piksli pa se nova vrednost sivine (svetlost) piksla izračuna kot neka funkcija vrednosti njegovih sosednjih pikslov. Ponavadi vzamemo piksle iz bližnje okolice tega piksla. Lokalne operacije imenujemo tudi **filtriranje slik**. Lokalne operacije so transformacije, ki transformirajo vrednost trenutnega piksla tako, da pri tem upoštevajo še vrednosti sosednjih pikslov.

Sosednje piksle trenutnega piksla definiramo z **okolico** *(angl. neighbourhood)*. Pod okolico razumemo vse piksle, ki se nahajajo znotraj maske (majhne podslike), ki je ponavadi kvadratne oblike, lihe

dimenzije, s središčem v trenutnem obravnavanem pikslu. *Slika 12* nam kaže primer različnih okolic trenutnega piksla.

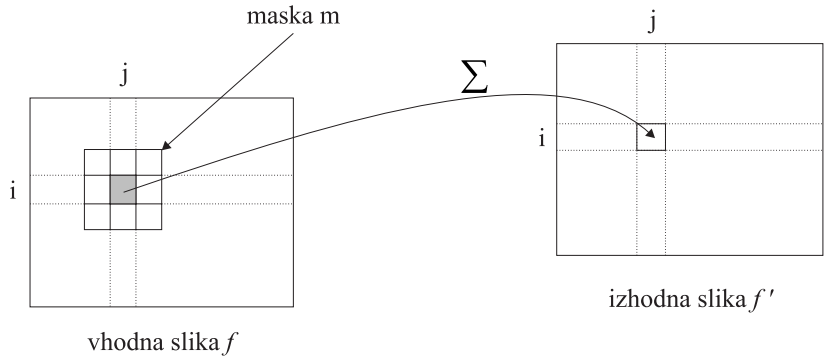


Slika 12: Okolice pikslov: a) okolica velikosti 3x3 piksle, b) 5x5 pikslov, c) 7x7 pikslov.

Lokalno operacijo (filtriranje) nad sliko izvedemo po naslednjem postopku:

1. Najprej izberemo velikost okolice (npr. 3x3 piksle).
2. Določimo uteži znotraj maske (maska je iste velikosti kot okolica).
3. Položimo masko na sliko tako, da se sredina maske prekriva s trenutnim pikslom, ki ga filtriramo. Masko začnemo polagati v zgornjem levem kotu slike, nadaljujemo v isti vrstici do konca, zatem gremo v drugo vrstico in tako ponavljamo do spodnjega desnega kota slike.
4. Ko položimo masko na ustrezno mesto, potem izračunamo konvolucijo med masko in digitalno sliko. Konvolucijo opravimo tako, da najprej izračunamo produkt med vsakim elementom maske in pripadajočim elementom digitalne slike, ki se nahaja direktno pod tem elementom maske. Vse te produkte zatem med seboj seštejemo, tako izračunano vrednost pa priredimo pikslu, ki ga filtriramo.

Slika 13 nam shematsko prikazuje računanje z lokalnim operatorjem velikosti 3x3 piksle. Prikazano je filtriranje piksla na mestu *(i,j)*, ki je na sliki tudi poudarjen. Problem za filtriranje predstavljajo piksli, ki nimajo popolnoma zapolnjene maske s sosednjimi piksli (ponavadi piksli ob robu slike). Npr. če položimo masko na piksel v prvi vrstici, potem pod celo zgornjo vrstico maske ni nobenega piksla digitalne slike (*Slika 13*). Ta problem rešimo tako, da rečemo, da so ti elementi enaki 0. Druga možna rešitev je tudi, da takšnih pikslov, ki so ob robu slike ne filtriramo. V tem primeru se nam zmanjša dimenzija izhodne slike, kar pa je lahko moteče.



Slika 13: Shematski prikaz računanja z lokalnim operatorjem

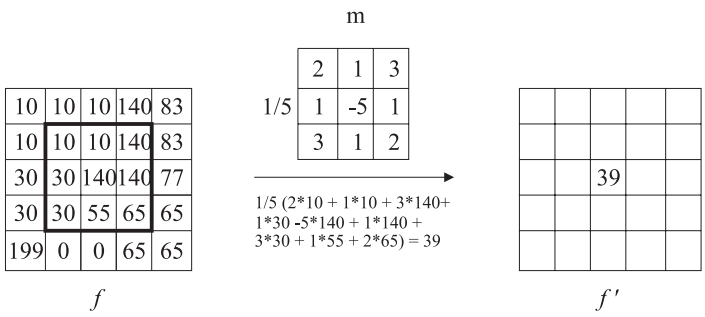
Računanje lokalne operacije lahko zapišemo tudi z matematičnim izrazom:

$$f'(i,j) = \sum_{(k,l)\in O} \sum m(K+k, L+l)\, f(i+k, j+l), \qquad k = -K,...,K; \quad l = -L,...,L$$

kjer *(i,j)* pomeni koordinati piksla, ki ga transformiramo, *f* je vhodna slika, *f ′* je izhodna slika, *m* je konvolucijska maska in *O* je okolica piksla. Maska *m* (konvolucijska maska) je ponavadi lihih dimenzij *(2K+1)x(2L+1)*.

Po filtriranju se lahko zgodi, da vrednosti vseh pikslov v izhodni sliki niso več znotraj intervala [0,255], ampak so lahko tudi manjše (negativne) ali večje. V tem primeru je najbolje, da poiščemo minimalno in maksimalno sivino v takšni sliki ter opravimo linearizacijo (podpoglavje 5.3), s katero ponovno vse vrednosti sivin omejimo na interval [0,255]. Ta problem lahko rešimo tudi s katero drugo funkcijo za spreminjanje kontrasta.

Zgled: Kot zgled si poglejmo računanje z lokalnim operatorjem *m* velikosti 3x3 piksle. Vhodna slika *f*, prav tako pa tudi izhodna slika *f ′* sta velikosti 5x5 pikslov. Prikazano je filtriranje za piksel *(2,2)* (šteti začnemo z 0!). S krepko črto je na vhodni sliki označena 3x3-okolica trenutnega piksla. Piksel filtriramo tako, da izračunamo vse produkte med masko in elementi vhodne matrike, ki so pod masko, te produkte seštejemo, dobljeno vrednost pa priredimo pikslu, ki ga filtriramo. Pod puščico je skiciran izračun.



Ločimo dve večji skupini lokalnih operatorjev, in sicer:

1. **operatorji glajenja** *(angl. smoothing operators)*. Uporabljamo jih predvsem za odstranjevanje šuma in drugih motenj v slikah. Njihova velika slabost je, da zapacajo *(angl. blur)* ostre robove objektov v slikah.
2. **detektorji robov** *(angl. edge detectors)*. Ti lokalni operatorji so namenjeni za poudarjanje robov objektov v slikah. V slikah iščejo mesta, kjer pride do velikih sprememb v sivinah (rob dejansko mejo med sivinami objekta in ozadja).

V nadaljevanju bomo našteli nekaj lokalnih operatorjev iz obeh skupin.

a. Operatorji glajenja

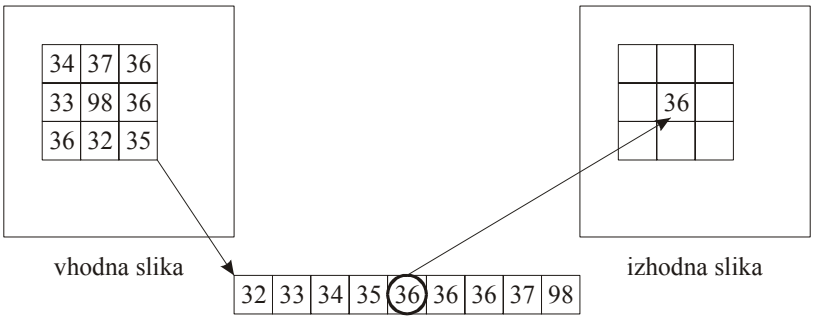
Nizko sito *(angl. low-pass filter)* je lokalni operator s katerim naredimo povprečje v okolici piksla. Je operator glajenja, ki nam v sliki ohrani počasne spremembe v sivinah, medtem ko nam drobne detajle zabriše. Konvolucijska maska *m* za nizko sito je za 3x3-okolico oblike:

$$m = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

Gaussov filter je lokalni operator, ki zelo dobro odpravlja Gaussov šum iz slik. Konvolucijska maska je oblike:

$$m = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}.$$

Mediani filter je filter za glajenje slik, ki minimalno zapaca robove. Ideja medianega filtra je v tem, da trenutni piksel zamenjamo z vrednostjo mediane iz njegove okolice. *Slika 14* prikazuje delovanje tega filtra. Mediana svetlosti ni izpostavljena individualnim šumnim konicam, zato je ta metoda primerna za odpravljanje impulznega šuma. Če je deformiran samo en piksel v okolici, je dovolj, da vzamemo mediani filter 3x3, za večje motnje pa je potrebno vzeti večji filter.



Slika 14: Delovanje medianega filtra

Glavna slabost teh filtrov je v njihovi kvadratni okolici piksla, ki poškoduje (popači) tanke linije in ostre robove v sliki. Ta problem lahko delno rešimo z izbiro drugačne okolice, npr. okolica oblike križa bi bila primerna za ohranjanje vertikalnih in horizontalnih linij. Mediani filter je le eden izmed filtrov, ki urejajo vrednosti sosednjih pikslov v vrsto in iz nje izberejo en element (*angl. rank value filter*). Trivialen primer sta minimalni in maksimalni filter, kjer izberemo minimalen oziroma maksimalen element.

b. Detektorji robov

Detektorji robov so skupina zelo pomembnih metod za lokalno predobdelavo slik in se uporabljajo za ugotavljanje nenadnih sprememb v funkciji intenzivnosti (slikovni funkciji) - robovi so piksli, kjer se funkcija nenadno spremeni. Rob je lokalna lastnost vsakega piksla in je vektorska spremenljivka, sestavljena iz dveh komponent: velikosti in smeri. Velikost roba je velikost gradienta slikovne funkcije, smer roba pa je enaka smeri gradienta minus 90^0, ob predpostavki, da je smer 0^0 definirana tako, da kaže proti vzhodu. Velikost gradienta | grad f(x,y) | in smer gradienta ψ v zvezni predstavitvi slikovne funkcije f(x,y) hitro izračunamo kot:

$$\left| \text{grad f(x, y)} \right| = \sqrt{\left(\frac{\partial \text{f}}{\partial \text{x}}\right)^2 + \left(\frac{\partial \text{f}}{\partial \text{y}}\right)^2} \quad \text{in} \quad \psi = \arg\left(\frac{\partial \text{f}}{\partial \text{x}}, \frac{\partial \text{f}}{\partial \text{y}}\right).$$

Na diskretni mreži lahko prve parcialne odvode aproksimiramo z razlikami:

$$\frac{\partial \text{f(x, y)}}{\partial \text{x}} \approx \frac{\text{f(x, y) - f(x - } \Delta\text{x, y)}}{\Delta\text{x}} \quad \text{razlika nazaj,}$$

$$\approx \frac{\text{f(x + } \Delta\text{x, y) - f(x, y)}}{\Delta\text{x}} \quad \text{razlika naprej,}$$

$$\approx \frac{\text{f(x + } \Delta\text{x, y) - f(x - } \Delta\text{x, y)}}{2\Delta\text{x}} \quad \text{simetrična razlika.}$$

Podobno bi zapisali prvi odvod po y ter tudi druge odvode. Obstaja več gradientnih operatorjev, ki merijo strmino robov. Ogledali si bomo nekaj preprostih operatorjev na osnovi razlik.

Robertsov operator je eden izmed najstarejših operatorjev za detektiranje robov. Nastal je okrog leta 1965. Njegova konvolucijska maska je oblike:

$$h_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \text{ oziroma } h_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

Velikost roba se izračuna po formuli $|f(i,j)-f(i+1,j+1)|+|f(i,j+1)-f(i+1,j)|$. Slabost Robertsovega operatorja je v veliki občutljivosti na šum.

Laplaceov filter je operator za odkrivanje oziroma za poudarjanje robov v slikah. Konvolucijska maska Laplaceovega operatorja je za 3x3-okolico oblike:

$$m = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}.$$

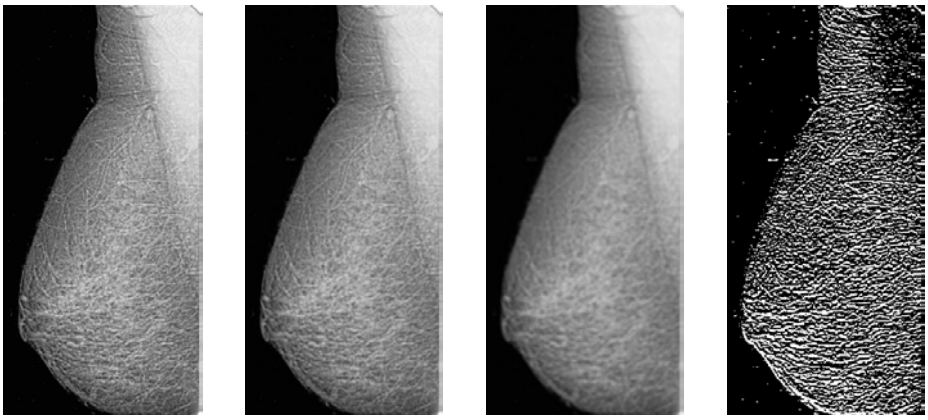
Ta operator vrne po filtriranju vrednost 0 za tiste dele slike (regije), kjer so sivine konstante, kjer pa so robovi, pa vrne neničelno vrednost. Vrednost je lahko ali pozitivna ali negativna in je odvisna od tega, kje leži središčna točka glede na rob (pozitivna vrednost je takrat, ko preidemo iz svetlih sivin objekta na temnejše sivine drugega objekta oziroma ozadja, obratno velja za negativne vrednosti). Za vizualiziranje slik po filtriranju, v katerih najdemo negativne kakor tudi pozitivne vrednosti pikslov, je običajno, da prištejemo k vsakemu pikslu srednjo vrednost sivine (npr. sivino 128 za slike z 256 sivinami). S tem lažje zaznamo robove (svetle in temne vrednosti), ki smo jih npr. dobili z Laplaceovim operatorjem. Slike, ki jih dobimo kot rezultat filtriranja z operatorji za poudarjanje robov, imenujemo **slika robov** *(angl. edge image)*.

Sobelov operator je še eden izmed množice operatorjev, ki poudarjajo robove v slikah. Ta operator ima dve konvolucijski maski, in sicer:

$$m_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \text{ s katero poudarjamo horizontalne robove, in masko}$$

$$m_2 = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \text{ s katero poudarjamo vertikalne robove.}$$

Če z y označimo odziv od m_1 in z x odziv od m_2, potem lahko velikost roba določimo kot $\sqrt{x^2+y^2}$ oziroma $|x|+|y|$, smer pa iz arctg(y/x). Kako v praksi izračunamo velikost roba? Po zgornjih enačbah izračunamo vrednost roba za vsak piksel posebej. Tako npr. velikost roba za piksel (65,150) določimo tako, da kvadriramo vrednost na mestu (65,150) iz slike y, ki smo jo dobili po filtriranju originalne slike z m_1, zatem pa to vrednost prištejemo k kvadrirani vrednosti na mestu (65,150) iz slike x, ki smo jo dobili po filtriranju originalne slike z m_2, to vsoto pa zatem še koreninimo. *Slika 15* prikazuje nekaj primerov filtriranja z lokalnimi operatorji.



(a) (b) (c) (d)

Slika 15: Primer filtriranja z lokalnimi operatorji: a) originalni rentgenski posnetek dojke (mamogram), b) filtriranje z Gaussovim filtrom 3x3 piksle, c) filtriranje z nizkim sitom 5x5 pikslov, d) filtriranje s Sobelovim operatorjem za poudarjanje horizontalnih robov (prag 20).

5.4.3 Slikovna matematika

Pri obdelovanju se srečamo tudi s t.i. **slikovno matematiko** *(angl. image math)*. K slikovni matematiki štejemo operacije kot so: seštevanje, odštevanje, množenje in deljenje slik. Aritmetične operacije nad digitalnimi slikami se izvajajo na nivoju pikslov, to pomeni, da se npr. množenje dveh slik izvede tako, da množimo isto ležeče piksle v obeh slikah, podobno velja za ostale operacije.

Operaciji seštevanja in odštevanja sta pomembni, če želimo sliki dodati informacije iz drugih slik (npr. dodajanje kakšnih oznak ali drugih napisov na sliko, vizualno poudarjanje robov – sliko robov odštejemo od originalne slike ipd.). Operacija deljenja se najpogosteje uporablja za odstranjevanje ozadja iz slik, operacijo množenja pa v praksi le redko uporabljamo. Po aritmetični operaciji so vrednosti sivin pikslov ponavadi zunaj območja možnih vrednosti sivin (npr. sivine so negativne ali večje od 255), zato moramo takšnim slikam, preden jih prikažemo, še spremeniti kontrast (npr. linearizacija).

5.5 SEGMENTACIJA SLIK

S predobdelavo slik smo želeli popraviti izgled slike, npr. odstranili smo šum ali spreminjali kontrast v slikah, dodajali določene oznake na slike ipd. V posameznih primerih je takšna obdelava dovolj in lahko tako obdelano sliko že kar shranimo na disk (npr. podatkovno bazo). Nekatere aplikacije pa poleg vizualnih popravkov v slikah, želijo določiti oziroma razpoznati, kateri objekti so v sliki. V teh primerih pa poleg faze predobdelave, potrebujemo še dve fazi obdelovanja slik, in sicer: segmentacijo in klasifikacijo.

Glavni cilj **segmentacije** *(angl. segmentation)* je, da razdeli sliko na dele, ki kar najverjetneje ustrezajo objektom ali pa območjem iz realnega sveta. S segmentacijo združujemo (grupiramo) tiste piksle v sliki, ki najverjetneje sestavljajo en objekt.

Kakšna je dobra segmentacija slike? Regije, dobljene s segmentacijo, naj bodo enotne in homogene glede na določene karakteristike, kot so npr. sivine ali teksture. Notranjost regij naj bo preprosta in brez mnogih lukenj. Sosednje regije naj imajo izrazito različne vrednosti glede na karakteristike, po katerih so enotne. Meje vsakega segmenta naj bodo preproste, nenazobčane (raskave) in prostorsko natančne. Zadostiti vsem tem želenim lastnostim segmentacije je težko, ker so striktno enotne in

homogene regije polne majhnih lukenj in imajo nazobčane meje. Če vztrajamo, da naj imajo sosednje regije velike razlike v vrednostih, lahko povzročimo, da se regije zlijejo in meje med posameznimi segmenti izginejo. Segmentacijske tehnike so večinoma »ad hoc« in se razlikujejo v načinu, po katerem poudarijo eno ali več želenih lastnosti, in v načinu, kako uravnavajo in delajo kompromise med želenimi lastnostmi.

V nadaljevanju bomo opisali segmentacijske metode, ki jih bomo glede na njihovo dominantno značilnost, ki jo uporabljajo, razdelili v tri velike skupine. Prva skupina metod – pragovna operacija – uporablja globalno znanje o sliki, ki ga ponavadi predstavimo s histogramom sivin. Segmentacija na osnovi robov tvori drugo veliko skupino metod, zadnja skupina metod pa je segmentacija na osnovi regij.

5.5.1 Pragovne operacije

Pragovna operacija *(angl. thresholding)* je segmentacijska metoda, ki na osnovi vrednosti sivin pikslov določi, kateri piksel pripada objektu (objektom) in kateri ozadju. Je transformacija vhodne slike *f* v izhodno segmentirano binarno sliko *s* (**binarna slika** je digitalna slika, ki ima le dva različna nivoja sivin: črno in belo) po naslednjem predpisu:

$$s(i,j) = \begin{cases} 1, & \text{če } f(i,j) \geq T \\ 0, & \text{sicer} \end{cases},$$

kjer je *T* konstanten **prag** *(angl. threshold)* – konstantna vrednost sivine, $s(i,j) = 1$ označuje piksel objekta, $s(i,j) = 0$ pa piksel ozadja. Zgoraj opisani predpis nam pove, da vsak piksel, ki ima vrednost sivine večjo ali enako neki vnaprej predpisani vrednosti *T*, pripada objektu, sicer pa ozadju.

Če se objekti v sliki ne dotikajo (tudi ne prekrivajo) in če se vrednosti sivin objektov bistveno razlikujejo od sivin ozadja, potem je pragovna operacija z globalnim pragom (en prag veljaven za vso sliko) primerna segmentacijska metoda. V praksi je to zelo redko. Zato raje uporabljamo variabilen prag, ki se spreminja s položajem v sliki. Pragovna operacija se torej spremeni v:

$$s(i,j) = \begin{cases} 1, & \text{če } f(i,j) \geq T(i,j) \\ 0, & \text{sicer} \end{cases}.$$

Glavna razlika med obema pragovnima operacijama je v tem, da se sedaj odločamo o posameznem pikslu, ali pripada objektu ali ne, in to na osnovi primerjave s pragom, ki je točno določen za ta piksel (vsak piksel ima svoj prag).

a. Metode za določanje globalnih pragov

Pravilna izbira praga je ključnega pomena za uspešno izvedbo segmentacije s pragovno operacijo. Prag lahko določimo na mnogo načinov, npr. vrednost praga določimo sami interaktivno, obstaja pa tudi mnogo postopkov za avtomatsko določanje pragov. Najpreprosteje avtomatsko določimo globalni prag tako, da poiščemo minimalno in maksimalno sivino v sliki, prag pa postavimo na polovico med obe vrednosti, torej $T = (min + max)/2$.

Za ostale metode bomo morali najprej izračunati histogram sivin *h* za sliko *f*. Naj bo $h = (h_0, h_1, ..., h_n)$ torej histogram sivin, pri čemer naj element h_i pomeni število pikslov v sliki z vrednostjo sivine *i*, kjer je *n* maksimalna sivina v sliki (ponavadi 255). Izračunajmo še dva pomožna izraza:

$$A_j = \sum_{i=0}^{j} h_i, \quad B_j = \sum_{i=0}^{j} i h_i; \quad j = 0,...,n.$$

Tako A_j pomeni število vseh pikslov v sliki, ki imajo svetlost j ali manj, B_j pa predstavlja obteženo vsoto pikslov do svetlosti *j*, kar je proporcionalno povprečni svetlosti v sliki, če upoštevamo le svetlosti do *j*-te.

Globalni prag določen z mediano (MEDIAN) izberemo tako, da je izraz

$$\frac{A_T}{A_n} \approx 0{,}5$$

čim bližje polovici. Vrednost *T*, torej prag, tedaj pomeni svetlost, pod katero in nad katero je ravno polovica pikslov. **Globalni povprečni prag** (MEAN) pa je določen kot

$$T = \frac{B_n}{A_n}.$$

Za določitev **globalnega optimalnega praga** (OPTIMAL) moramo poznati približno začetno vrednost praga (npr. vzamemo povprečni prag). V nadaljevanju se z iterativno formulo približujemo optimalni pragovni vrednosti:

$$T_{k+1} = \frac{\mu_{T_k} + \gamma_{T_k}}{2}, \quad \text{kjer sta } \mu_{T_k} = \frac{B_{T_k}}{A_{T_k}} \text{ in } \gamma_{T_k} = \frac{B_n - B_{T_k}}{A_n - A_{T_k}}.$$

Vidimo, da je novi prag v *k+1* iteraciji določen kot aritmetična sredina povprečnih pragov na slikovnih območjih, ki so pod sedanjim pragom in nad njim. Iteracija se zaključi, ko dosežemo konvergenco in se prag ne spreminja več bistveno (npr. $T_{k+1} = T_k$).

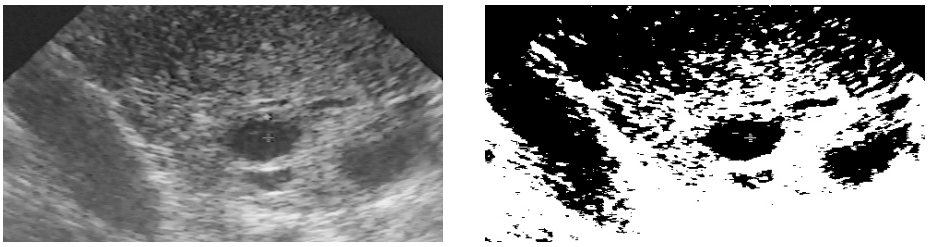
b. Metode za določanje variabilnih pragov

Variabilni pragi niso enotni za celo sliko, kakor so globalni pragi, temveč se prilagajajo vsakemu pikslu posebej. Pri tem odigra pomembno vlogo pikslova okolica. Za osnovo vzamemo enega od globalnih pragov, najraje optimalnega (T_{opt}). Prag za vsak piksel sproti pa se izračuna kot $T = T_{opt}(1-x),$ pri čemer se vrednost *x* določi z izrazom

$$x = \frac{1}{3Q}(\lfloor a-b-c+d \rfloor + \lfloor a-b+c-d \rfloor + \lfloor a+b-c-d \rfloor).$$

Q je število kvantizacijskih nivojev (ponavadi 255), *a,b,c* in *d* pa so vrednosti pikslov, ki so razporejeni v ogliščih matrike 3x3 s središem nad obravnavanim pikslom. Postopek povzroči, da se optimalni prag tem bolj spremeni, čim bolj so prisotne spremembe v svetlosti horizontalno, vertikalno in diagonalno v okolici opazovanega piksla.

Slika 16 nam kaže primer pragovne operacije z globalnim pragom. Kot rezultat smo dobili binarno sliko. Opazimo lahko, da nam bela barva določa ozadje, črna barva pa objekte (zaradi boljšega prikaza smo vrednosti za objekt in ozadje invertirali).



(a) (b)

Slika 16: Primer pragovne operacije: a) originalna slika, b) binarna slika, prag T je enak 100.

5.5.2 Segmentacijske metode na osnovi robov

Segmentacijske metode na osnovi robov imajo na vhodu sliko, ki je rezultat operatorjev za detektiranje robov iz faze predobdelave (glej lokalno predobdelavo slik). Operatorji za detektiranje robov nam v izhodno sliko res dajo vrednosti robov, vendar takšne slike ne moremo uporabiti kot rezultat segmentacije. Zato moramo takšno sliko še dodatno obdelati. Segmentacijske metode na osnovi robov združujejo robove v verige robov, ki še bolje aproksimirajo mejo objekta.

Glavna problema segmentacijskih metod na osnovi robov, ki nastaneta zaradi šuma ali nezanesljive informacije v sliki, sta prisotnost robov na področjih, kjer ni mej, in odsotnost robov, kjer te meje obstajajo. V nadaljevanju si bomo ogledali nekaj preprostih metod.

a. Pragovna operacija na sliki robov

V sliki robov (rezultat predobdelave) ne najdemo pikslov z vrednostjo 0, ampak le majhne vrednosti, ki so ponavadi rezultat nepomembnih sprememb nivoja sivine zaradi kvantizacijskega šuma, majhnih nepravilnosti v osvetlitvi itd. Preprosta pragovna operacija odpravi te majhne vrednosti (recimo Sobelov operator).

b. Sledenje meji

Če meje regij niso znane, regije pa so v sliki definirane (npr. s pragovno operacijo), potem lahko meje enolično določimo. Obstajata dve meji regije: zunanja in notranja. Notranja meja regije je dejanska meja regije in je vedno del regije. Zunanja meja regije pa je meja ozadja in ni del regije. Zapišimo psevdokod za sledenje notranji meji regije:

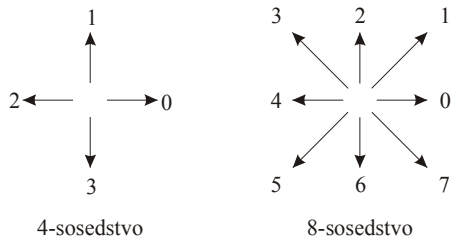
1. Preiskuj sliko iz levega zgornjega kota vrstico za vrstico, dokler ne naletiš na piksel, ki pripada novi regiji. Označi ta piksel s P_0. Piksel P_0 je začetni piksel meje regije. Spremenljivka dir hrani smer prejšnjega pomika vzdolž meje iz prejšnjega piksla meje do trenutnega piksla. Priredi:
a) dir = 3, če odkrivamo mejo v 4-sosedstvu;
b) dir = 7, če odkrivamo mejo v 8-sosedstvu.

2. Preišči 3x3-sosesko trenutnega piksla proti urni smeri (Slika 17 prikazuje možne smeri), začni pri pikslu, ki je v smeri:
a) (dir+3) mod 4,
b) (dir+7) mod 8, če je dir lih in
(dir+6) mod 8, če je dir sod.

Prvi piksel, ki ima enako vrednost, kot jo ima trenutni piksel, je novi element meje in ga označimo s P_n. Ažuriraj spremenljivko dir.

3. Če je trenutni element meje P_n enak elementu meje P_1 in če je prejšnji element meje P_{n-1} enak P_0, potem je konec, sicer ponovi korak 2.

4. Zunanja meja regije je določena s piksli $P_0,\ldots,P_{n-2}$.



Slika 17: Vnaprej določene smeri za sledenje meji v 4- in 8-sosedstvu.

Algoritem za sledenje notranji meji lahko uporabimo tudi kot algoritem sledenja zunanji meji regije – zamenjamo le pomen objekta in ozadja. Algoritem za sledenje notranji meji dela zelo dobro za vse regije, ki so večje od enega piksla – kar pa je trivialen problem. S tem algoritmom ne moremo najti mej lukenj znotraj regij. V takšnih primerih je potrebno algoritem malo modificirati.

Zunanja in notranja meja regije sta velikega pomena za nadaljnje korake obdelave slik (klasifikacijo), saj lahko iz njiju hitro določimo nekatere značilnosti (npr. obseg regije, kompaktnost itd.), na osnovi katerih se lahko odločimo o pripadnosti objekta.

Veliko težje pa je slediti meji, če imamo na vhodu le sliko robov. V takšnih primerih moramo povezovati robove po nekem kriteriju (npr. podobne velikosti in smeri robov). Tako najdemo metode, ki določajo meje na osnovi preiskovanja grafa (hevristično algoritem A, dinamično programiranje), s pomočjo Houghove transformacije itd. Meji regije je možno slediti tudi v slikah z več nivoji sivin.

5.5.3 Segmentacijske metode na osnovi regij

V prejšnjem podpoglavju smo iskali meje med regijami, v tem podpoglavju pa se bomo srečali z metodami, ki konstruirajo regijo direktno iz pikslov. Tehnike konstruiranja regije se izkažejo primerne za šumne slike, kjer tehnike na osnovi robov odpovejo. Homogenost regije je pomembna lastnost regije in se največkrat uporablja kot glavni segmentacijski kriterij teh metod. Tehnike rasti regije delijo sliko na območja z maksimalno homogenostjo. Kriteriji za homogenost so lahko na osnovi sivin, vzorcev idr. Segmentacijske metode na osnovi regij imenujemo tudi metode rasti regije (*angl. region growing*). V nadaljevanju si bomo pogledali preprosto segmentacijsko metodo na osnovi regije.

Zlitje regij

Vse segmentacijske metode zlitja regij lahko popišemo s splošnim algoritmom:

1. Definiraj začetno metodo, ki bo razdelila sliko v mnogo majhnih regij, tako da bo izpolnjen kriterij

$H(R_i) = TRUE\ ,\ \ i=1,2,...S\ ,$

$H(R_i \cup R_j) = FALSE\ ,\ \ i \neq j$ *ter* R_i *in* R_j *sta sosednji regiji,*

kjer so R_i *regije, S je število vseh regij in* $H(R_i)$ *je kriterij homogenosti za regijo* R_i.

2. Definiraj kriterij za zlitje dveh sosednjih regij.

3. Zlij vse sosednje regije, ki zadovoljujejo kriterij za zlitje. Če ne moremo več zliti nobenega para regij, potem KONEC.

Vse metode zlitja regij imajo enako ogrodje, razlikujejo se le v definiciji začetne segmentacijske metode in kriterija zlitja. Rezultat metod zlivanja regij je ponavadi odvisen od vrstnega reda, po katerem zlivamo regije – tako je odvisno, ali segmentacijo začnemo v zgornjem levem kotu ali pa v spodnjem desnem kotu.

Preproste metode zlivanja regij začnejo segmentacijo z začetnimi regijami velikosti 2x2 pikslov (nekatere tudi 4x4 in več). Opis regije ponavadi podamo z njihovimi statičnimi vrednostmi sivin. Pri tem si lahko pomagamo tudi z lokalnim histogramom. Zatem primerjamo opisa dveh sosednjih regij in če se skladata, ju zlijemo ter določimo opis za to združeno regijo. Proces segmentacije je končan, ko ne moremo več zliti nobenih sosednjih regij.

V tem algoritmu dve sosednji regiji zlijemo, če nova združena regija izpolnjuje homogenostni kriterij. Homogenosti kriterij lahko npr. določimo kot produkt razlike povprečnih vrednosti svetlosti v dveh

regijah in standardnega odklona svetlosti, ki bi ga izračunali znotraj nove, združene regije. Označimo povprečno svetlost regije k z m_k, standardni odklon pa s σ_k. Regiji k in l zlijemo, če

$$|m_k - m_l|\sigma_{(k \cup l)} < T,$$

pri čemer je T vnaprej predpisan prag za zlivanje.

Pri postopku rasti regij se mnogokrat dogodi, da ostanejo prav majhne regije nezlite z večjimi in razpršene po sliki. V takem primeru lahko uporabimo še poobdelavo, s katero tem majhnim, nepovezanim otočkom najdemo tisto sosednjo večjo regijo, s katero se kar najbolje ujemajo po združitvenem (homogenem) kriteriju, čeprav ga v celoti ne izpolnjujejo.

5.6 PARAMETRI ZA OPISOVANJE OBLIK OBJEKTOV

Segmentacija je priredila vrednost 1 vsem pikslov, ki verjetno tvorijo objekte, s tem pa nam je tudi definirala regije v sliki. **Regija** *(angl. region)* je množica pikslov, v našem primeru pikslov z vrednostjo 1, kjer so vsi sosednji piksli povezani med seboj oziroma se stikajo. Ponavadi sta pojma regija in objekt (ozadje) ekvivalenta, vendar dokler ne vemo, da gre res za objekt (ozadje), raje uporabljamo izraz regija. V binarni sliki lahko imamo več takšnih regij (Slika 16 – med posameznimi regijami je ozadje – tudi ozadje je regija, vendar za nas ni zanimiva). Za nadaljnjo obdelavo je nujno, da takšne regije ločimo med seboj, zato jih moramo na nek način označiti oziroma identificirati.

5.6.1 Označevanje regij

Označevanje regij (*angl. region labelling)* oziroma barvanje regij dodeli vsaki regiji svoje število. Največje celo število oznake pomeni število vseh regij (objektov) v sliki. Vhod v algoritem označevanja je binarna slika, pri čemer so objekti predstavljeni z neničelnimi vrednostmi, ozadje pa z 0. Algoritem za označevanje regij v 8-sosedstvu:

1. Prvi prehod: Preišči celotno binarno sliko ***s*** *vrstico za vrstico in priredi vsakemu neničelnemu pikslu s(i,j) neničelno vrednost v. Vrednost v izberemo glede na oznake sosedov trenutnega piksla:*

s(i-1,j-1)	s(i-1,j)	s(i-1,j+1)
s(i,j-1)	s(i,j)	

Slika 18: Sosednji piksli v 8-sosedstvu

- *če so vsi sosedi iz ozadja, potem pikslu s(i,j) priredi novo, še neuporabljeno oznako;*
- *če ima samo en sosed neničelno oznako, potem priredi to oznako pikslu s(i,j);*
- *če ima več kot en sosed neničelno oznako, potem priredi pikslu s(i,j) oznako enega izmed teh sosedov. Če se oznake sosedov razlikujejo (trk oznak), potem shrani vse te oznake in jih označi, da so ekvivalentne. Ekvivalentni par shrani v posebno tabelo ekvivalence.*

2. Drugi prehod: Celotno sliko preglej še enkrat in popravi oznake pikslov glede na tabelo ekvivalence. S tem odpravimo trk oznak.

Izhod tega algoritma je slika, kjer ima vsak objekt svojo oznako (število) v. Vrednost v pa zajema iz obsega celih števil med 1 in S, kjer je S število vseh objektov v sliki.

5.6.2 Popisovanje regij z značilkami

Posamezne regije so ponavadi sestavljene iz velikega števila pikslov, to veliko število pa hkrati zelo oteži primerjavo regij med seboj. Zato v praksi skušamo iz množice pikslov, ki sestavljajo posamezno regijo, izluščiti najbolj tipične lastnosti – **značilke** *(angl. features)*, s katerimi lahko prav tako dobro popišemo posamezno regijo. Ponavadi regije z značilkami dovolj natančno aproksimiramo. Omeniti moramo, da je teh značilk veliko manj kot je število vseh pikslov v regiji. Za popis krožnice (ali kroga) popolnoma zadostujeta dve značilki, in sicer: težišče in radij. V splošnem lahko tvorimo kakršnokoli značilko, najdemo pa tudi značilke, katerih vrednosti imajo fizikalen pomen. Poglejmo si nekaj takšnih.

Ploščina regije *(angl. area)* je najbolj preprost parameter regije. V digitalnih slikah je ploščina podana s številom pikslov, ki sestavljajo regijo. Če je regija (objekt) predstavljena z matriko, potem jo izračunamo tako, da kar preštejemo vse piksle, ki jo sestavljajo.

Kompaktnost regije *(angl. compactness)* je zelo kvaliteten parameter, saj je neodvisen od linearnih transformacij. Izračunamo jo kot ulomek $c = o^2/p$, kjer je o **obseg regije** (dolžina meje) in p njena ploščina. Kompaktnost leži v intervalu $[1,\infty)$, pri čemer vrednosti blizu 1 pomenijo, da je regija bolj okrogle oblike (kompaktne), čim večja je kompaktnost, tem bolj razvejana je regija (npr. morska zvezda je izredno nekompaktne oblike).

Težišče regije *(angl. centre of gravity)* oziroma masno središče določimo s pomočjo naslednjih dveh pomožnih izrazov:

$$m_{10} = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} i\, s(i,j) \text{ in } m_{01} = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} j\, s(i,j) ,$$

k obema izrazoma prištevamo le tiste piksle, ki pripadajo regiji za katero računamo masno središče. Težišče, njegov položaj označimo s (x_T,y_T), pa določimo kot $x_T = m_{10}/p$ in $y_T = m_{01}/p$, kjer p pomeni ploščino regije.

Polarni histogram. Pretežni del informacije o obliki predmeta nosi kontura (obris) njegovega tlorisa. To konturo pa lahko povzorčimo. Obris predmeta lahko tako opazujemo izhajajoč iz njegovega težišča. Zato merimo radialne razdalje od težišča do robov (stranic) lika naokrog v izbrani ravnini. Čim bolj je ta obris razgiban (višje frekvence sprememb oblike), tem večja mora biti vzorčevalna frekvenca. To pomeni, da sedaj radialne odmike ugotavljamo v manjših kotnih razmikih. Na koncu dobimo niz značilk, ki jih enostavno zapišemo kot:

$$x = (x_{\varphi_0}, x_{\varphi_1}, ..., x_{\varphi_{N-1}}).$$

Pri tem so $x_{\varphi r}$ radialne razdalje med težiščem in obodom lika, razsežnost opazovanega prostora N pa se izbere kot:

$$N = \left\lfloor \frac{2\pi}{\varphi} \right\rfloor ,$$

kjer je φ kot med dvema sosednjima vzorcema.

5.7 KLASIFIKACIJA REGIJ (OBJEKTOV)

V fazi segmentacije smo definirali regije, jih označili ter vsako popisali z značilkami. Kljub vsemu delu, ki smo ga opravili, pa še vedno ne vemo, kaj oziroma kakšen objekt opisuje posamezna regija. Ali predstavlja ta regija znan objekt ali ne? Če znan, kateri objekt? Odgovore na ta in podobna vprašanja nam vrne klasifikacija.

Klasifikacija (*angl. classification)* nam vsako regijo razvrsti na osnovi nekih kriterijev v določen razred objektov. Poglejmo si primer. Če npr. v sliki iščemo jabolka, hruške in slive, ostali objekti pa nas ne zanimajo, potem to pomeni, da vsako regijo uvrstimo v enega izmed štirih definiranih razredov objektov, in sicer: v razred jabolk ali hrušk ali sliv ali pa v razred ostalih objektov. Za objekt pravimo, da je razpoznan, šele ko ga uvrstimo v določen razred. Seveda lahko objekt uvrstimo tudi v napačen razred. Takrat govorimo o napaki klasifikacije. Če želimo uspešno klasifikacijo, potem moramo zelo dobro določiti kriterijsko funkcijo (odločitvena pravila). Ta kriterijska funkcija mora natančno opredeliti, za katere vrednosti značilk, lahko posamezno regijo še uvrstimo v določen razred objektov (npr. v razred hrušk).

Slika 19: Postopek klasifikacije

Slika 19 nam shematsko prikazuje postopek klasifikacije. Vidimo, da najprej za posamezno regijo (prostor oblik) določimo značilke (najpomembnejše lastnosti objekta), na osnovi teh značilk pa kriterijska funkcija (odločitveno pravilo) razvrsti posamezno regijo v enega izmed K razredov.

Zakaj sploh potrebujemo značilke? Vedeti moramo, da lahko v digitalni sliki velikosti 256x256 pikslov objekt zavzema tudi 100x100 pikslov in več. To pomeni, da bi pri klasifikaciji tega objekta imeli opravka z 10^4 podatkov. Seveda pa v realnosti obstajajo slike veliko večjih dimenzij (npr. 2048x2048). To pa postane računsko in tudi časovno neobvladljivo, zato si pomagamo z značilkami. Množica P značilk tvori P-dimenzionalen prostor, ki ga imenujemo prostor značilk. Vsaka regija je v tem prostoru predstavljena kot P-dimenzionalni vektor značilk. Dimenzija prostora značilk P naj bi bila veliko manjša od dimenzije prostora oblik ($P<<N$).Če so bile značilke izbrane dobro, potem vsi vektorji značilk objektov iz določenega razreda ležijo skupaj, sicer pa lahko pride do prekrivanja razredov.

Slika 19 tudi nazorno prikazuje, da odločitveno pravilo (kriterijska funkcija) razdeli prostor značilk v K razredov. Dobra izbira značilk poenostavi odločitveno pravilo in tudi razredi se v prostoru značilk ne prekrivajo. Odločitveno pravilo je neka funkcija, ki ima na vhodu P-dimenzionalen vektor značilk, kot izhod pa vrne indeks razreda, v katerega je objekt razvrščen (klasificiran).

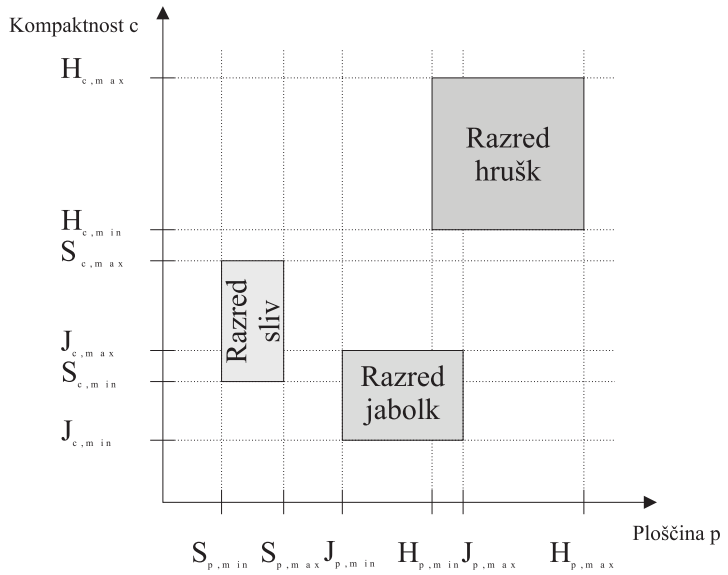
Zgled: Poglejmo si primer razpoznavanja sadežev: jabolk, hrušk in sliv. Vsako regijo bomo popisali z dvema značilkama, in sicer s ploščino ter s kompaktnostjo. Regija je torej popisana z 2-dimenizionalnim vektorjem značilk **x**, ki je oblike (x_1,x_2), kjer x_1 pomeni ploščino regije, x_2 pa kompaktnost. Hkrati pa imamo tudi 2-dimenzionalen prostor značilk. Prostor značilk moramo z odločitvenim pravilom razdeliti na 4 dele (na tri razrede sadežev ter razred drugih objektov).

Slika 20 prikazuje eno izmed možnih razdelitev prostora značilk. Z oznako *S* smo označili slive, z *J* jabolka, z *H* hruške. Ploščino smo označili z *p*, kompaktnost pa z *c*. Oznaka *min* pomeni minimalno, *max* pa maksimalno vrednost, npr. $S_{p,min}$ označuje minimalno vrednost ploščine za slive. Zapišimo še odločitveno pravilo *d*, ki nam razdeli prostor značilk na takšen način:

$$d(\mathbf{x}) = \begin{cases} \text{slive}, & \text{če } x_1 \in \left[S_{p,\min}, S_{p,\max}\right] \wedge x_2 \in \left[S_{c,\min}, S_{c,\max}\right] \\ \text{jabolka}, & \text{če } x_1 \in \left[J_{p,\min}, J_{p,\max}\right] \wedge x_2 \in \left[J_{c,\min}, J_{c,\max}\right] \\ \text{hruške}, & \text{če } x_1 \in \left[H_{p,\min}, H_{p,\max}\right] \wedge x_2 \in \left[H_{c,\min}, H_{c,\max}\right] \\ \text{drugo}, & \text{sicer} \end{cases}.$$

Zgornji izraz nam pove, da če je ploščina za regijo, ki jo klasificiramo, npr. znotraj intervala za slive ($[S_{p,min},S_{p,max}]$), in če je tudi kompaktnost znotraj intervala za slive, potem to regijo uvrstimo v razred sliv, sicer po enakem postopku poskusimo za jabolka in hruške. Če nam regije ne uspe razvrstiti v

nobenega od treh razredov, potem rečemo, da je ta regija neklasificirana oziroma jo uvrstimo v razred ostalih objektov.



Slika 20: Možna razdelitev prostora značilk za primer sadežev.

5.7.1 Princip avtomatskega razpoznavalnega sistema

Vsak neznan objekt, ki ga razpoznavamo z razpoznavalnim sistemov, je po koncu obdelovanja opisan z vektorjem značilk (glej *Slika 19*):

$$\mathbf{x} = (x_1, x_2, ..., x_P)^T .$$

Vektor **x** si lahko predstavljamo kot točko v prostoru z razsežnostjo *P*. Ker želimo, da bi računalnik sam uspel razvrščati nove objekte (oblike) v že znane razrede, moramo znati te razrede v popolnosti predstaviti. Tako tvorimo zbirko prepoznanih oblik (objektov), za katere vemo, katerim razredom pripadajo. Takšnim oblikam pravimo **vadbeni vzorci** ali **prototipi** ali **praliki**. Pralike označimo s simbolom $w_m^{(k)}$, kjer indeks *k* pomeni, da je pralik vzet iz razreda S_k, indeks *m* pa, da gre za pralik *m* iz tega razreda. Pralik $w_m^{(k)}$ je torej spet predstavljen kot vektor:

$$w_m^{(k)} = (w_{1m}^{(k)}, ..., w_{rm}^{(k)}, ..., w_{Pm}^{(k)}).$$

Naš končni cilj je torej dobiti odgovor na vprašanje: kateremu praliku je naš neznan objekt najbolj podoben. Predpostavimo, da imamo pri *k*-tem razredu M_k primerkov (*m* teče od *1* do M_k), *K* je število razredov (*k=1...K*) in *P* je število značilk v vektorju značilk (*r=1...P*). Dejansko moramo na nek način določiti mejo med posameznimi razredi – določiti odločitveno pravilo. V praksi največkrat temelji odločitveno pravilo na merjenju razdalje (npr. Evklidske razdalje) med znano in neznano obliko (pralikom). To lahko formalno zapišemo

$$d(\mathbf{x}, w_m^{(k)}) .$$

Ponavadi iščemo minimalno razdaljo za vsak razred. V tem primeru dejansko izračunamo razdalje preko vseh indeksov *m* ter določimo minimalno (indeks razreda *k* je fiksiran):

$$\min_{m=1:M_k} d(\mathbf{x}, w_m^{(k)}) .$$

Pri odločitvenem pravilu »**Najbližji sosed«,** ki se v praksi najpogosteje uporablja, pa poiščemo najmanjšo razdaljo med našim neznanim objektom in vsemi praliki. Neznan objekt pa uvrstimo v tisti

razred iz katerega prihaja njegov najbližji sosed – torej pralik, ki je od njega najmanj oddaljen. Zapišimo pravilo najbližji sosed še formalno:

Objekt **x** uvrstimo v razred S_i, če $i = \left\{ \min_{\mathrm{m}} d(x, w_m^{(i)}) \leq \min_{\forall \mathrm{k} \neq \mathrm{i,m}} d(x, w_m^{(k)}) \right.$.

V literaturi obstaja še cela vrsta drugih odločitvenih pravil. Kako se odločiti katero odločitveno pravilo je dobro? Velik problem predstavljajo tudi zelo pomešani razredi objektov (vzorcev). V tem primeru nobeno odločitveno pravilo ne izkazuje 100 % uspešnosti. Obstajata dve meri, **senzitivnost** in **specifičnost**, ki opredeljujeta uspešnost odločitvene metode (razpoznavalnega sistema).

Senzitivnost

Senzitivnost (*angl. sensitivity*) je mera, ki podaja odstotek pravilno razpoznanih objektov (oblik, odločitev) določene vrste. Definirana je kot razmerje:

Senzitivnost (S_k) = število_pravilno_uvrščenih_v_razred_S_k / število_vseh_nastopajočih_v_S_k.

Senzitivnost razpoznavalnega postopka se nanaša na opazovani razred S_k. Čim bližje je 100 %, tem bolj se lahko zanesemo, da bomo oblike razreda S_k pravilno razpoznali.

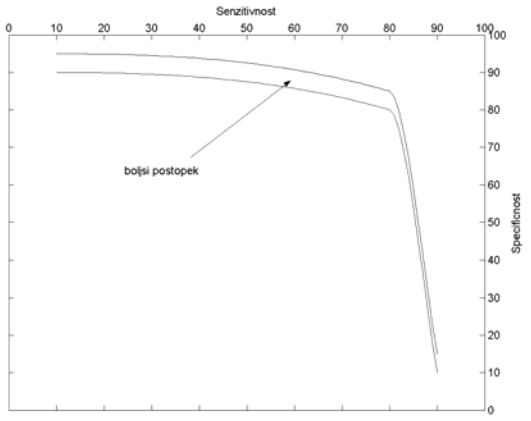
Specifičnost

Specifičnost izraža odstotek pravilno opredeljenih objektov, ki ne pripadajo opazovanemu razredu S_k. Specifičnost glede na Sk je torej:

Specifičnost (S_k) = število_vseh_pravilno_uvrščenih_izven_S_k / število_vseh_izven_razreda_S_k.

Čim višja je specifičnost, tem bolj smemo biti prepričani, da objekt, ki ga nismo spoznali »za svojega«, res ne sodi v razred S_k.

Najboljši bi bil seveda tak razpoznavalni postopek (odločitveno pravilo), ki bi popolnoma prav razvrščal vse možne vhodne vzorce. Zanj bi bili specifičnost in senzitivnost 100 %. Žal ga v praksi večinoma ni mogoče uresničiti, zato se zadovoljimo pač s čim višjim odstotkom tako senzitivnosti kot specifičnosti. Izkaže se namreč, da zviševanje senzitivnosti za vsako ceno začne občutno zniževati specifičnost in seveda obratno. *Slika 21* prikazuje navedeno soodvisnost.



Slika 21: Soodvisnost senzitivnosti in specifičnosti

Literatura

Brown CW, Shepherd BJ. Graphics file formats. Greenwich: Manning publications, 1995.

Castleman KR. Digital image processing. Englewwod Cliffs: Prentice Hall, 1996.

Heijden F. Image based measurement systems. London: Wiley and Sons, 1994.

Jahne B. Digital image processing. Berlin: Springer–Verlag, 1993.

Keller PR, Keller MM. Visual Cues. Los Alamitos: IEEE Press, 1993.

Parker JR. Practical computer vision using C. New York: Wiley and Sons, 1993.

Pavešić N. Razpoznavanje vzorcev. Ljubljana: Fakulteta za elektrotehniko in računalništvo, 1992.

Potočnik B. Uporaba segmentacije pri analizi medicinskih slik. Magistrsko delo. Maribor: Fakulteta za elektrotehniko, računalništvo in informatiko, 1998.

Russ JC. The image processing handbook. London: CRC Press, 1995.

Sonka M, Hlavac V, Boyle R. Image processing, analysis and machine vision. London: Chapman Hall, 1994.

Vprašanja in naloge

1. Kaj je razpoznavalni sistem?
2. Na kakšne načine lahko pridobivamo digitalne slike? Kakšen je pomen digitalizacije pri zajemanju slik?
3. Opišite delovanje ene izmed naprav za zajemanje medicinskih slik!
4. Kaj je kontrastna ločljivost digitalne slike? Kako jo lahko spreminjamo?
5. Na katere faze v grobem razdelimo obdelovanje slik? Kaj je glavni namen posamezne faze?
6. Izračunajte novo (filtrirano) vrednost za piksel v 3 vrstici in 2 stolpcu spodaj zapisane matrike, če to matriko filtriramo z Gaussovim filtrom velikosti 3x3 piksle.

40	45	43	42	42	40
35	155	156	165	175	30
30	153	160	163	178	25
30	30	120	115	45	35
25	25	110	100	40	40
20	20	80	95	45	45

7. Zapišite rezultat po pragovni operaciji nad zgoraj zapisano matriko, če uporabimo za prag T vrednost 50.

Odgovori:

1. Pod imenom razpoznavalni sistem združujemo vse postopke digitalne obdelave slik, in sicer od formiranja slike do njenega razumevanja. Tipičen razpoznavalni sistem ima naslednjo zgradbo: zajemanje slik, predobdelava (odstranjevanje šuma), segmentacija, klasifikacija in shranjevanje rezultatov.
2. Digitalne slike lahko dobimo na tri različne načine: sintetičen način, s pomočjo prebirnika in s snemanjem direktno iz naprav. Digitalizacija (slikovni digitalizator) pretvori analogno vrednost signala (napetost), ki je lahko poljubna realna vrednost, v določeno celoštevilsko vrednost sivine.
4. Kontrastna ločljivost slike določa število različnih sivin, ki jih najdemo v sliki. Spreminjamo jo lahko z različnimi funkcijami za spreminjanje kontrasta, npr. linearizacijo, izenačitev histograma, s pomočjo prenosnih funkcij.
5. Obdelovanje slik v grobem delimo na: fazo predobdelave, segmentacije in klasifikacije. S predobdelavo pripravimo digitalno sliko za nadaljnje obdelovanje, in sicer spreminjamo sliki kontraste ali odstranjujemo šum, odkrivamo robove itd. Namen segmentacije je, da združimo v skupno regijo tiste piksle v slikah, ki najverjetneje tvorijo objekte. V fazi klasifikacije pa skušamo posamezno regijo uvrstiti v določen razred objektov.
6. Nova filtrirana vrednost je 106.

$$\frac{1}{16}\begin{bmatrix} 1*35+2*155+1*156+ \\ 2*30+4*153+2*160+ \\ 1*30+2*30+1*120 \end{bmatrix} = \frac{1703}{16} \cong 106$$

7. V spodaj zapisani matriki je prikazan rezultat po pragovni operaciji. Opazimo lahko, da smo za rezultat dobili regijo v obliki črke T.

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