FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ZAGREB

DIPLOMSKI RAD

Željko Vrba

Zagreb, rujan 2001.

Ovom prilikom zahvaljujem mentoru prof. dr. sc. Slobodanu Ribariću na savjetima prilikom izrade rada. Također zahvaljujem osoblju zračne luke Zagreb što su omogućili pokusno snimanje prtljage.

1 Uvod

1.1 Opis problema

Osnovni zadatak je raspoznavanje oružja u slikama ručne prtljage putnika koji putuju avionom. Sustav nije zamišljen kao *zamjena* za djelatnike aerodromskog osiguranja koji pregledavaju prtljagu, nego kao *pomoćno sredstvo* koje djelatniku treba skrenuti pažnju na potencijalno opasne predmete unutar prtljage. Stoga je otpočetka sustav rađen tako da radije daje lažne pozitivne rezultate (prijavi opasnost gdje je nema) nego da ne prijavi opasnost tamo gdje postoji.

Slike su dobivene pokusnim snimanjem vlastite prtljage s podmetnutim (lažnim) oružjem u Zračnoj luci Zagreb (aerodrom Pleso). Za pokusno snimamnje upotrijebljena su 4 različita pištolja različitih veličina kao i jedan nož. Tako je snimljeno više slika s oružjem u različitim položajima, kao i slike bez ikakvih opasnih predmeta. Osim opasnih predmeta u prtljazi je bilo par knjiga, košulje, torbe, sprej, baterije, miš (uređaj) te kišobran. Sve je to prilikom snimanja bilo na različite načine razmješteno u koferu.

Sam postupak raspoznavanja ima tri faze:

- Segmentacija je postupak koji sliku rastavlja na dijelove koji su zanimljivi za sljedeći korak.
- Izlučivanje značajki raspoznavanja: u ovom slučaju to su konture zanimljivih objekata. Na temelju kontura računaju se stvarne značajke.
- Raspoznavanje: odlučivanje predstavlja li kontura nekog objekta stvarno zanimljivi predmet.

1.2 Tehnika snimanja prtljage

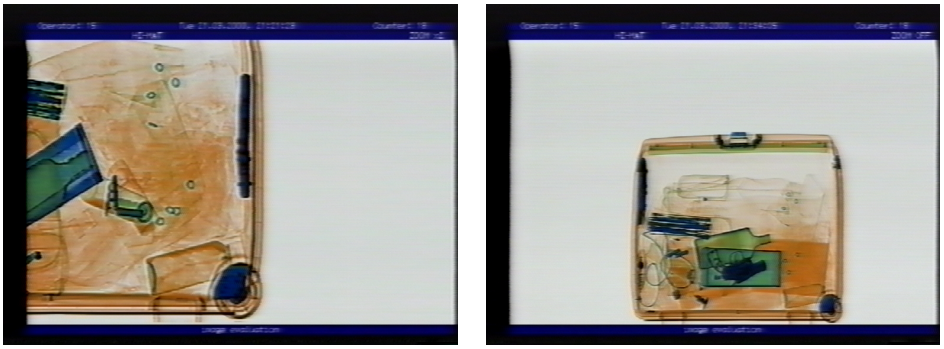
U zračnoj luci se koristi moderna oprema: umjesto snimanja klasičnim rendgenom, snimanje se obavlja tehnikom sličnom CT-u. Naime, umjesto snimanja dvodimenzionalne slike (kao što je slučaj s rendgenom), stroj sloj po sloj snima *trodimenzionalnu sliku* cjelokupnog prostora predviđenog za prtljagu, i to sa prilično velikom rezolucijom: radnici u osiguranju kažu da je, primjerice, moguće razaznati izbočenja na registarskim tablicama (motornih vozila), pa i pročitati oznake, ukoliko su snimljene u frontalnom položaju.

Međutim zbog tehničkih ograničenja nisam imao pristup potpunim 3D podacima, već samo dvodimenzionalnoj slici nakon računalne obrade. Operater može birati koji će sloj gledati, povećati određene dijelove slike, kao i mijenjati *paletu* boja.

Osim što stroj omogućava 3D snimanje prtljage, također mjeri atomsku masu i masivnost predmeta u prtljazi: što je neki predmet masivniji (deblji, teži) bit će prikazan u tamnijoj

nijansi. Što ima veću atomsku masu (metal) pojavljuje se u hladnijim bojama (plavo, zeleno...). Operater ima izbor kodiranja boje: tako može izabrati da na slici budu izražene organske tvari (npr. za otkrivanje krijumčara cigareta) ili metali (oružje). U razgovoru s radnicima osiguranja saznao sam da se pri pregledavanju prtljage većinom oslanjaju na iskustvo: iako je laiku predmet po konturama neprepoznatljiv (npr. kišobran), radnik odmah vidi da predmet ne predstavlja prijetnju. Ukoliko je predmet jako masivan i dovoljno velik, tada je putnik sumnjiv i prtljaga se ručno pregledava.

Za potrebe ovog rada, operater je odabrao paletu u kojoj se najbolje ističu metalni predmeti, a prikazani su svi slojevi prtljage na istoj slici. Slike su snimljene na VHS kazetu, a kasnije digitalizirane i spremljene na disk u PNG formatu [11]. Treba istaknuti da ovaj postupak snimanja unosi velike teškoće u postupak raspoznavanja: golema količina informacija se izgubi već analizom 2D slike umjesto 3D modela. Zatim je tu šum na kazeti, kao i manja rezolucija VHS kazete (PAL standard: 768×576 piksela) od rezolucije snimanja. Slika 1.1 prikazuje dvije snimke na kojima će se prikazati pojedini koraci raspoznavanja.



Slika 1.1

1.3 Fizikalne osnove boje

Slike koje se obrađuju su u boji te će se zato ovdje izložiti neki osnovni pojmovi o boji.

Godine 1666. je Isaac Newton propustio sunčevu svjetlost kroz staklenu prizmu i otkrio da izlazno svjetlo nije bijelo, već se da sastoji od kontinuiranog *spektra* boja, od crvene do ljubičaste. Te boje su ljubičasta, plava, zelena, žuta, narančasta i crvena (vidi trokut boje na slici 1.3). To se naziva *vidljivi spektar*, a raspon valnih duljina je $\sim 400 - 700$nm.

Boja koju ljudi vide određena je prirodom svjetlosti koja je *reflektirana* od objekta. Tako primjerice, zeleni objekti reflektiraju najveći dio svjetlosti u rasponu 500-570nm, dok apsorbiraju većinu ostalih valnih duljina. Međutim boja je isključivo psihološki osjet: postoje boje koje ljudi vide, a *ne nalaze* se u spektru, kao npr. smeđa.

Ukoliko je svjetlo *akromatsko*, tada je jedina karakteristika takvog svjetla *intenzitet*. Zbog građe ljudskog oka, sve boje se mogu dobiti pomoću raznih omjera *osnovnih boja*: crvene

(R, red), zelene (G, green) i plave (B, blue). Zapravo je moguće uzeti bilo koje tri boje kao osnovne, ali RGB sustav je posebno pogodan zbog tehnologije izrade katodne cijevi. Tablica 1.1 daje valne duljine triju osnovnih boja, kako ih je 1931. standardizirala Međunarodna komisija za osvijetljenje (CIE).

plava	zelena	crvena
435.8	546.1	700

Tablica 1.1 Valne duljine osnovnih boja (nm)

Osnovne boje se mogu miješati u kombinacijama po dvije kako bi se dobile *sekundarne boje*: ljubičasta (magenta; crvena i plava), cijan (zelena i plava) te žuta (crvena i zelena). Kod miješanja boja razlikuju se dvije vrste primarnih boja:

Boje svjetlosti Miješanjem sve tri dobiva se bijela svjetlost (tzv. *aditivno miješanje*).

Boje pigmenta Primarna boja je ona koja *oduzima* primarnu boju *svjetlosti* i reflektira ostale dvije. Dakle, primarne boje pigmenta su ljubičasta, cijan i žuta. Kad se sve tri pomiješaju dobije se crna boja *(suptraktivno miješanje)*.

Posljednji sustav se naziva CMY (cyan, magenta, yellow). Prijelaz iz RGB u CMY sustav opisan je sljedećom jednadžbom:

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

Karakteristike na temelju kojih se jedna nijansa boje razlikuje od drugih su:

Svjetlina (intensity) govori o *intenzitetu* svjetlosti (bijelo, sivo).

Ton (hue) ovisi o dominantnoj valnoj duljini u mješavini valnih duljina koje čine boju. Kada se kaže da je neki predmet crven, narančast ili žut, govori se o njegovom tonu.

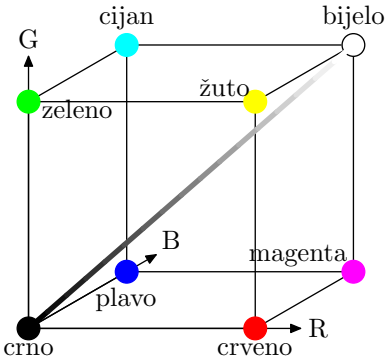
Zasićenost (saturation) je relativna čistoća boje, odn. iznos bijele svjetlosti pomiješane s tonom. Boje spektra su potpuno zasićene. Npr. ružičasta boja je manje zasićena od crvene, a iznos zasićenosti obrnuto je proporcionalan količini pomiješane bijele svjetlosti.

Ton i zasićenost zajednički se nazivaju **kromatičnost** boje.

Već je pokazano da se neka boja može opisati pomoću tri broja. Koja tri broja se odaberu za opis boje određuje *model boje*. Ovdje su bila razmatrana dva modela: RGB (zato što je u takvom modelu ulazna slika) i HSI (zato što je vrlo sličan ljudskoj percepciji). U nastavku će se opisati osnova svakog modela, kao i konverziju među njima.

1.3.1 RGB model

Ovaj model najlakše se opisuje Kartezijevim koordinatnim sustavom (vidi sliku 1.2). Dijagonala kocke određuje razne intenzitete *sive* boje, a u vrhovima se nalaze primarne i sekundarne boje.



Slika 1.2 Kocka boje

Na temelju slike moglo bi se reći da se siva slika iz RGB slike dobije računanjem srednje vrijednosti 3 komponente:

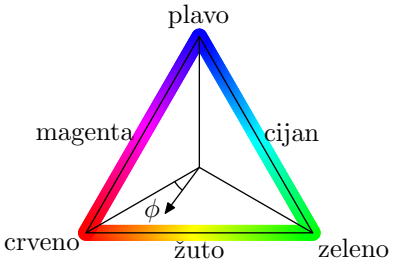
$$I = \frac{R + G + B}{3} \tag{1.1}$$

Međutim u zahtjevnijim primjenama obrade slike se koristi složenija formula zato što ljudsko oko *nije jednako osjetljivo* na sve tri primarne boje. Pokazuje se da je najosjetljivije na zelenu, a najmanje osjetljivo na plavu boju, što znači da većina osvijetljenja koju čovjek vidi dolazi od zelenih nijansi. Zato se intenzitet računa prema modificiranim formulama koje uzimaju u obzir osjetljivost oka:

$$I = 0.299R + 0.587G + 0.114B \tag{1.2}$$

1.3.2 HSI model

Ovaj model temelji se na perceptivnim svojstvima boje (svjetlina (I), ton (H) i zasićenost (S)) i ima dvije temeljne prednosti. Jedna je što je svjetlina odvojena od informacije o boji. Druga je ta što su komponente tona i zasićenosti jako bliske ljudskom poimanju boje. Grafički odnos između komponenti HSI modela je prikazan na slici 1.3.



Slika 1.3 Trokut boje

Za zadani intenzitet, boja se može predočiti kao točka unutar trokuta boje. Kut ϕ u odnosu na neku od 3 spojnice težišta i vrha trokuta određuje ton, a udaljenost od težišta određuje zasićenost: što je točka bliže rubu, boja je zasićenija.

1.3.3 Konverzija između modela

Budući da je ulazna slika u RGB modelu, a razmatrala se segmentacija temeljena na HSI modelu, potrebna je konverzija iz jednog u drugi model. Izvod je podugačak pa će se samo navesti formule; detalji se mogu vidjeti u [5].

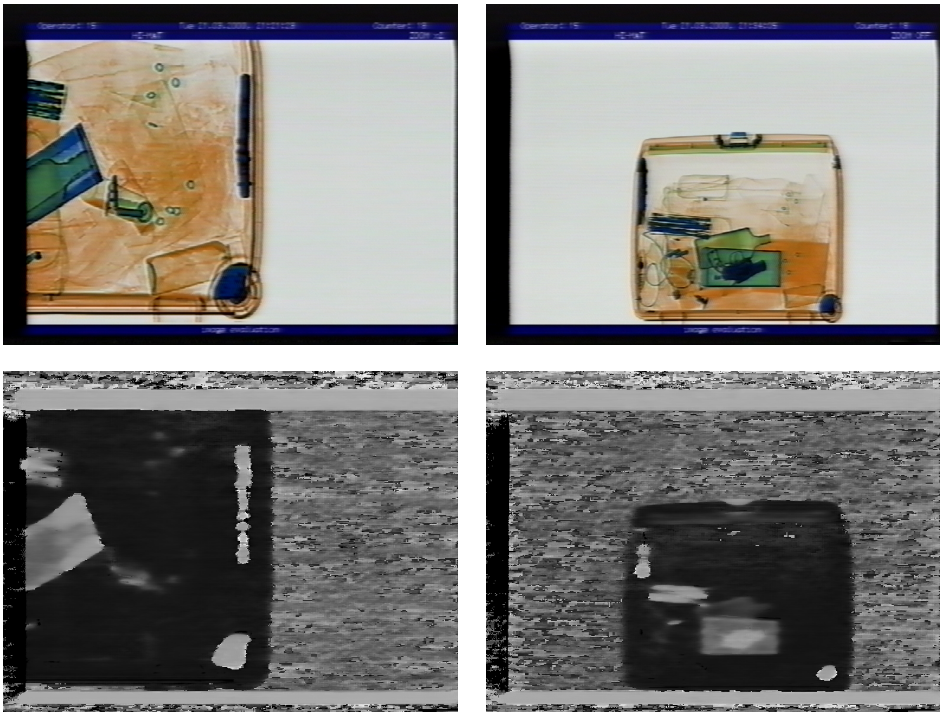
$$
\begin{aligned}
I &= \frac{1}{3}(R+G+B) \\
H &= \cos^{-1}\frac{1}{2}\frac{((R-G)+(R-B))}{\sqrt{(R-G)^2+(R-B)(G-B)}} \\
S &= 1-\frac{1}{I}\min(R,G,B)
\end{aligned}
\tag{1.3}
$$

Ovdje treba paziti na jedan važan detalj: treba staviti $H = 2\pi - H$ ukoliko je $B > G$. Rezultat za H je u intervalu $[0, 2\pi]$ i treba se normalizirati na odgovarajuće vrijednosti (obično cijeli brojevi 0-255 ili realni brojevi 0-1). Za pretvorbu iz HSI u RGB model postoje 3 kombinacije formula koje se mogu pogledati u [5].

2 Segmentacija

Segmentacija je postupak analiziranja slike koji u idealnom slučaju rastavlja sliku na područja tako da jedno područje odgovara jednom cijelom objektu. U ovom radu se koriste dvije vrste segmentacije: segmentacija na područja i segmentacija detekcijom rubova. Segmentacija na područja traži homogena područja na slici (slične svjetline i/ili boje; to je *amplitudna segmentacija*), a rubna segmentacija traži velike iznose druge derivacije slike. U idealnim uvjetima obje vrste segmentacije trebaju dati iste rezultate. Međutim, zbog neidealnih uvjeta (šum, preklapanje objekata...) segmentacija ne daje potpuni rezultat (u smislu da je dio objekta klasificiran kao pozadina ili obratno) što uvelike otežava daljnji postupak raspoznavanja.

Za segmentaciju na područja koristi se Kohonenova neuronska mreža [4] koja na ulaze prima R, G i B vrijednosti piksela na slici i pokazala se kao vrlo robustna metoda klasifikacije boja. Za segmentaciju na područja se razmatrala amplitudna segmentacija temeljena na HSI modelu, međutim formule 1.3 daju vrlo nepouzdane rezultate za H komponentu i unose mnogo šuma što jako otežava prepoznavanje (primjer su slike 2.1). Šum je pogotovo izražen u jako tamnim i jako svjetlim područjima.



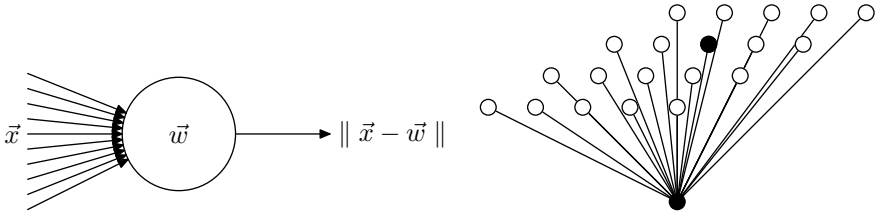
Slika 2.1 Šum H komponente u HSI modelu

Segmentaciju na područja slijedi detekcija rubova Cannyjevom metodom [10] kako bi se dobile granice između područja. Dobiveni rubovi su temelj za detekciju objekata u sceni.

2.1 Kohonenova neuronska mreža

2.1.1 Općenito o Kohonenovoj mreži

Ovu neuronsku mrežu prvi je proučavao Kohonen [8] za potrebe raspoznavanja govora [9]. Neuroni su raspoređeni u dvodimenzionalnu rešetku veličine $N = m \times n,\ m, n > 1$ (m je broj redaka, n broj stupaca), ali ne postoji pojam povezanosti neurona kao u npr. višeslojnom perceptronu zato što se ulaz dovodi na *sve neurone istodobno* (slika 2.2). Odabire se onaj koji daje najmanji izlaz.



Model neurona. $\vec{w}$ je k-dimenzionalni vektor težina, $\vec{x}$ je k-dimenzionalni vektor pobude.

Mreža na koju je doveden ulaz. Crni neuron odgovara.

Slika 2.2 Kohonenova neuronska mreža

U daljnjem tekstu će se pojedini neuron označavati samo jednim indeksom. Jednodimenzionalni indeks i neurona s težinskim vektorom $\vec{w}_i$ i dvodimenzionalne koordinate (x, y) u mreži (x je koordinata stupca, y je koordinata retka) povezane su relacijom

$$i = ny + x \tag{2.1}$$

($0 \leq x < n - 1,\ 0 \leq y < m - 1$).

U toku učenja razlikuju se tri procesa koji se izvode za svaki uzorak. t označava iteraciju algoritma.

- **Natjecanje:** Traži se neuron j za koji izraz $\| \vec{x} - \vec{w}_j \|$ poprima najmanju vrijednost. Neuron $\vec{w}_j$ se u [4] naziva *pobjednički*.

- **Suradnja:** Pobjednički neuron određuje susjedstvo neurona koji međusobno surađuju u donošenju odluke. Zato se definira *funkcija susjedstva* koja opada s udaljenošću od pobjedničkog neurona:

$$\begin{aligned} h_{i,j} &= \exp(-\frac{d_{i,j}^2}{2\sigma^2(t)}) \\ \sigma(t) &= \sigma_0 \exp(-\frac{t}{\tau_1}) \end{aligned} \tag{2.2}$$

$d_{i,j}$ je Euklidska udaljenost neurona u mreži. Ako neuron i ima koordinate (x_i, y_i), a neuron j (x_j, y_j) (preslikavanje 2d koordinata neurona u 1d indekse dano je jednadžbom 2.1) tada je

$$d_{i,j} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$

Formula 2.2 se koristi u [4]. Osim Gaussove funkcije u primjeni je česta i step funkcija [13,8], no pokazalo se da je Gaussova funkcija kvalitativno bolja (s biološkog stanovišta) od pravokutne, a također vodi i bržoj konvergenciji.

- **Adaptacija:** Da bi učenje bilo uspješno, težine pobjedničkog neurona $\vec{w}_j$ trebaju se promijeniti ovisno o ulaznom vektoru $\vec{x}$.

$$\vec{w}_j(t+1) = \vec{w}_j(t) + \eta(t) h_{j,i}(t)(\vec{x}(t) - \vec{w}_j(t))$$

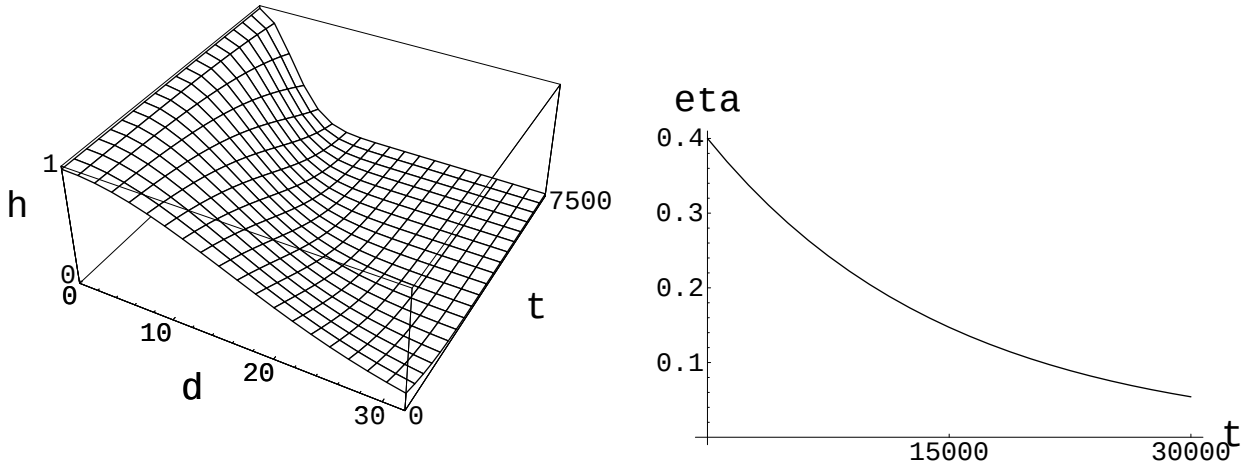
$\eta(t)$ je funkcija brzine učenja:

$$\eta(t) = \eta_0 \exp(-\frac{t}{\tau_2}) \tag{2.3}$$

Postupak se ponavlja sve dok nema značajnijih promjena u mreži.

U gornjim formulama postoje dvije vremenske konstante τ_1 i τ_2 koje određuju brzinu opadanja parametara σ i η. U ovom radu uzeto je $\eta_0 = 0.4$, $\sigma_0 = 16$, $\tau_1 = 5000$ iteracija te $\tau_2 = 15000$ iteracija. Slika 2.3 nacrtana je upravo prema ovim parametrima.

Prilikom učenja se pazi da uvijek bude $\eta(t) \geq 0.012$. Ukoliko bi formula 2.3 dala $\eta(t) < 0.012$ stavlja se $\eta = 0.012$ (ovo je također preporuka iz [4]).



Slika 2.3 Ovisnost parametara učenja o vremenu

2.1.2 Primjena na segmentaciju slike

Svaki neuron i ima pridružen trodimenzionalni vektor težina $\vec{w}_i$ (R, G i B komponente boje), a na izlazu daje Euklidsku udaljenost između naučene vrijednosti i ulaza (slika 2.2).

Za ovu primjenu izabrano je $m = n = 4$. Prilikom izbora dimenzija mreže bilo je potrebno odlučiti između malog broja neurona i *loše segmentacije* (objekti stopljeni sa pozadinom ili međusobno) te velikog broja neurona i *teške klasifikacije* (čovjek na kraju treba iz velikog broja razreda odabrati one koje odgovaraju bojama opasnih predmeta). Eksperimentiranjem je ustanvoljeno da je 16 razreda dovoljno za dobru segmentaciju, a istovremeno je lako izabrano 6, odn. 8 razreda zanimljivih boja.

U nastavku je detaljno opisan algoritam učenja (nešto izmijenjen u odnosu na algoritam opisan u [4]) koji je korišten u ovom radu za učenje mreže. Algoritam promatra sliku kao dugačak jednodimenzionalan vektor piksela. Kao i u prethodnom odjeljku, argument u zagradama (t) označava iteraciju, a indeksi označavaju pojedine neurone mreže.

1. Za svaki neuron j se generira početni slučajni vektor težina $\vec{w}_j(0)$. Svi neuroni međusobno trebaju imati različite početne vektore težina. Izabrana je inicijalizacija uniformno distribuiranim slučajnim brojevima u intervalu $[-0.5, 0.5]$.

2. Ovaj korak je početak jednog *ciklusa učenja*. Svi pikseli slike ispremiješaju se na slučajan način, tako da (gotovo) niti jedan piksel više nije na starom mjestu.

3. Uzima se uzorak $\vec{x}$ iz ulaznog prostora. Ovdje je to pojedini piksel slike. Komponente od $\vec{x}$ čine RGB komponente piksela skalirane na interval $[0, 1]$ (u ulaznoj slici svaka komponenta piksela je u intervalu $[0, 255]$).

   U [4] se preporuča slučajno izabiranje uzoraka. Ovdje to ne odgovara zato što ima mnogo crnih i vrlo svijetlih piksela (rub i pozadina) te stoga imaju i vrlo visoku vjerojatnost slučajnog odabira. Tada bi mreža u početku, još dok su parametri brzine učenja i veličine susjedstva veliki, alocirala mnogo neurona vrlo tamnim i vrlo svijetlim bojama i ne bi se kasnije mogla prilagoditi ostalim, za segmentaciju bitnim, bojama.

   Zato se umjesto slučajnog odabira pikseli uzimaju *po redu*, a slučajan odabir i izbjegavanje upravo opisanog problema velikih područja iste boje postiže se slučajnim miješanjem piksela u prethodnom koraku. Ovakav način odabira ima još jednu dobru stranu: u svakom ciklusu svaki piksel dođe na ulaz mreže *točno jedanput*.

4. Traži se neuron i koji se u iteraciji t najbolje poklapa s ulaznim neuronom (u [4] se taj neuron naziva pobjednički) koristeći kriterij najmanje udaljenosti u Euklidskom prostoru:

$$i = \min_{j=0\ldots N-1} \| \vec{x}(t) - \vec{w}_j(t) \|$$

5. Popravljaju se težine svih neurona prema formuli:

$$\vec{w}_j(t+1) = \vec{w}_j(t) + \eta(t) h_{j,i}(t)(\vec{x}(t) - \vec{w}_j(t))$$

   gdje je $\eta(t)$ parametar brzine učenja, a $h_{j,i}$ je funkcija susjedstva centrirana na ‘‘pobjedničkom’’ neuronu i. Oba parametra ovise o vremenu (vidi sliku 2.3).

U trenutku kada σ padne ispod neke male vrijednosti (10^{-5}) primjenjuje se sljedeća pojednostavljena formula i to samo na pobjednički neuron:

$$\vec{w}_j(t+1) = \vec{w}_j(t) + \eta(t)(\vec{x}(t) - \vec{w}_j(t))$$

To ubrzava postupak $N = m \times n = 16$ puta, a ne unosi značajnije pogreške u sam postupak jer pri tako malim vrijednostima σ ispravci susjednih neurona su zanemarivi.

6. Zajedno sa popravljanjem težina računa se mjera promjene u mreži ϵ:

$$\vec{\epsilon} = (\epsilon_r, \epsilon_g, \epsilon_b) = \sum_{j=0\ldots N-1} \eta(t) h_{j,i}(t)(\vec{x}(t) - \vec{w}_j(t))$$

$$\epsilon = |\epsilon_r| + |\epsilon_g| + |\epsilon_b|$$

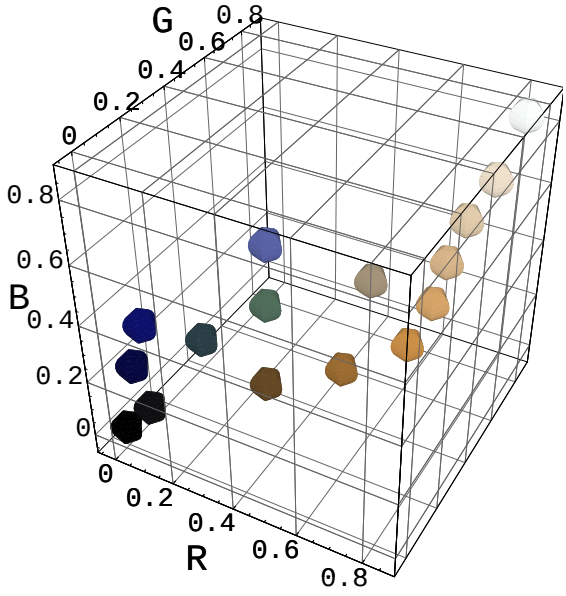
7. Ponavlja se korak 4 sve dok se ne obrade svi pikseli.

8. Nastavlja se od koraka 2 sve dok nema značajnijih promjena u mreži. Ovaj korak nije automatiziran već je program za učenje svakih 1000 iteracija ispisivao ϵ. Kada se 20 puta uzastopno desilo da je $\epsilon < 10^{-3}$ program je ručno zaustavljen i mreža je spremljena u datoteku. Učenje je trajalo nešto više od 3.1 miliona iteracija.

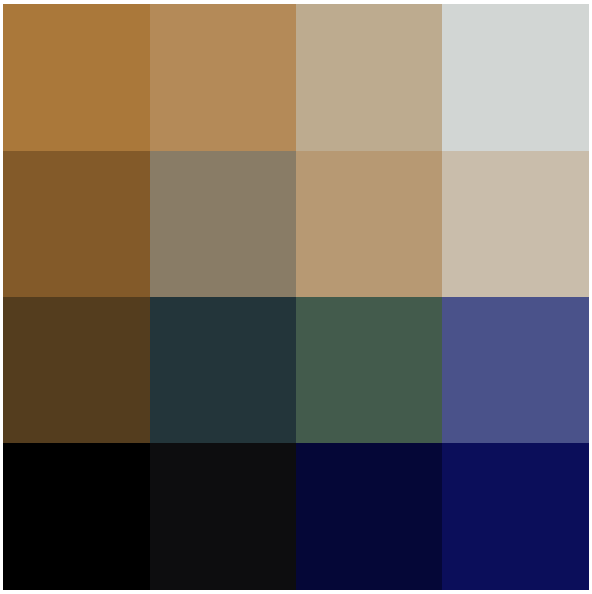
U nastavku je prikazano prvih 10000 iteracija u početku učenja te 10000 iteracija prije kraja učenja. Na temelju takvog ispisa programa odlučeno je u kom trenutku će se prekinuti učenje. Ispisivana je svaka 1000. iteracija. Redoslijed podataka je broj iteracije, 1d i 2d koordinate pobjedničkog neurona, ulazni podatak (piksel), težine neurona nakon ispravaka te mjera promjene ϵ.

```
1000:[0(0,0)][0.374203,13.0997](0,0,0,)(0.1126,0.116015,0.113656,)3.50891
2000:[0(0,0)][0.350069,10.7251](0,0,0,)(0.0205685,0.0206188,0.0207526,)0.809046
3000:[0(0,0)][0.327492,8.78099](0,0,0,)(0.0651961,0.066183,0.0848471,)2.12707
4000:[0(0,0)][0.306371,7.18926](0,0,0,)(0.02759,0.0213469,0.0549788,)0.836047
5000:[15(3,3)][0.286613,5.88607](0.843137,0.858824,0.862745,)(0.522651,0.52575,0.500214,)5.09907
6000:[0(0,0)][0.268128,4.81911](0,0,0,)(0.0487403,0.0493361,0.0483986,)1.21515
7000:[0(0,0)][0.250836,3.94555](0,0,0,)(0.00821355,0.00856737,0.00992791,)0.545122
8000:[11(2,3)][0.234658,3.23034](0.717647,0.54902,0.337255,)(0.592085,0.551072,0.481339,)0.740788
9000:[0(0,0)][0.219525,2.64478](0,0,0,)(0.00274434,0.00230124,0.00167677,)0.0978284
10000:[0(0,0)][0.205367,2.16536](0,0,0,)(0.000340157,0.000263374,0.000277175,)0.0231994

3192000:[15(3,3)][0.012,0](0,0,0,)(5.93974e-05,7.2874e-05,0.000100053,)2.82175e-06
3193000:[15(3,3)][0.012,0](0,0,0,)(0.000378337,0.00025371,0.000251983,)1.07372e-05
3194000:[15(3,3)][0.012,0](0,0,0,)(0.000299658,0.0002826,0.000269984,)1.03511e-05
3195000:[9(2,1)][0.012,0](0.25098,0.25098,0.301961,)(0.243569,0.348585,0.294718,)0.00136346
3196000:[15(3,3)][0.012,0](0.0117647,0.0156863,0.00392157,)(0.000240182,0.000211045,5.26763e-05,)0.000374923
3197000:[15(3,3)][0.012,0](0,0,0,)(0.000532119,0.000192324,0.000250807,)1.18451e-05
```

Ovom metodom učenja na jednoj slici sa tipičnom paletom (lijeva slika na 1.1) dobivena je paleta boja koja je numerički prikazana u tablici 2.1. Sve RGB vrijednosti normirane su na interval $[0, 1]$ tako da je bijela boja $(1, 1, 1)$ Slika 2.4 grafički prikazuje naučene boje i njihov položaj unutar kocke boja, kao i položaj boja unutar mreže (donja lijeva ćelija (neuron) ima koordinate $(0, 0)$, gornja desna $(3, 3)$).



Položaj u kocki boja



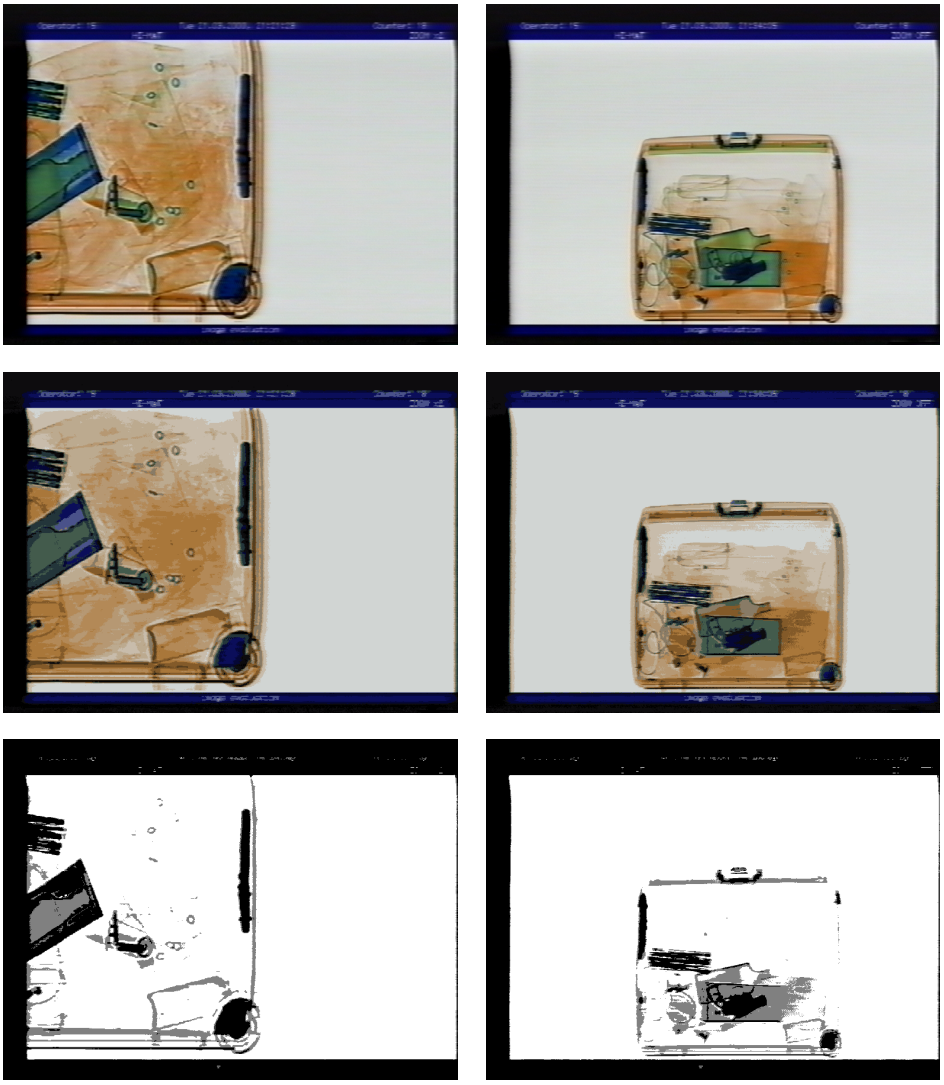
Položaj u mreži

Slika 2.4 Naučene boje

razred	R	G	B
0	$3.045338 \cdot 10^{-4}$	$9.326252 \cdot 10^{-5}$	$2.557287 \cdot 10^{-4}$
1	$5.276895 \cdot 10^{-2}$	$5.054473 \cdot 10^{-2}$	$5.872596 \cdot 10^{-2}$
2	$2.195709 \cdot 10^{-2}$	$3.062115 \cdot 10^{-2}$	$2.169867 \cdot 10^{-1}$
3	$4.577387 \cdot 10^{-2}$	$5.553664 \cdot 10^{-2}$	$3.548328 \cdot 10^{-1}$
4	$3.311855 \cdot 10^{-1}$	$2.401766 \cdot 10^{-1}$	$1.191730 \cdot 10^{-1}$
5	$1.384023 \cdot 10^{-1}$	$2.089930 \cdot 10^{-1}$	$2.302152 \cdot 10^{-1}$
6	$2.640369 \cdot 10^{-1}$	$3.587445 \cdot 10^{-1}$	$2.986328 \cdot 10^{-1}$
7	$2.921345 \cdot 10^{-1}$	$3.239503 \cdot 10^{-1}$	$5.418872 \cdot 10^{-1}$
8	$5.142536 \cdot 10^{-1}$	$3.553732 \cdot 10^{-1}$	$1.619957 \cdot 10^{-1}$
9	$5.368041 \cdot 10^{-1}$	$4.856691 \cdot 10^{-1}$	$4.003316 \cdot 10^{-1}$
10	$7.155607 \cdot 10^{-1}$	$5.997508 \cdot 10^{-1}$	$4.531101 \cdot 10^{-1}$
11	$7.877011 \cdot 10^{-1}$	$7.392312 \cdot 10^{-1}$	$6.707205 \cdot 10^{-1}$
12	$6.653483 \cdot 10^{-1}$	$4.694245 \cdot 10^{-1}$	$2.288481 \cdot 10^{-1}$
13	$7.049686 \cdot 10^{-1}$	$5.410360 \cdot 10^{-1}$	$3.438328 \cdot 10^{-1}$
14	$7.385663 \cdot 10^{-1}$	$6.680604 \cdot 10^{-1}$	$5.618610 \cdot 10^{-1}$
15	$8.212071 \cdot 10^{-1}$	$8.381093 \cdot 10^{-1}$	$8.307434 \cdot 10^{-1}$

Tablica 2.1 Naučene boje

Slika 2.5 prikazuje primjere obrade Kohonenovom mrežom. U prvom retku su originalne slike, drugi redak prikazuje sliku nakon što je svaka boja klasificirana u jedan od 16 razreda, dok treći redak prikazuje slike samo sa zanimljivim bojama opasnih predmeta. Razredi 0,1,2,3,5 i 7 sigurno pripadaju potencijalno opasnom predmetu te su obojane crno. Razredi 8-16 sigurno pripadaju pozadini, dok razredi 4 i 6 ponekad (ovisno o slici) pripadaju opasnim predmetima, a ponekad pozadini te su stoga obojane sivo.

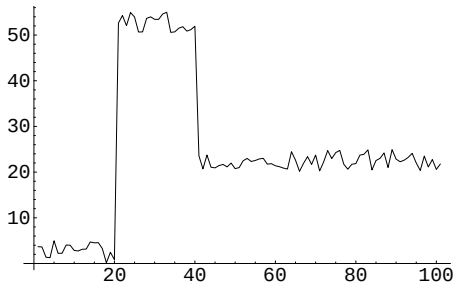


Slika 2.5 Primjeri obrade Kohonenovom mrežom

2.2 Detekcija rubova

Rub se u slici najčešće definira kao dovoljno nagla promjena svjetline. Cannyjev algoritam (opisan u nastavku) računa gradijent zamućene slike te gleda modul gradijenta kako bi odlučio koji su prijelazi dovoljno nagli da bi se mogli smatrati rubom. Svi parametri su određeni empirijski tako da na svim segmentiranim slikama daju dobre rezultate (nema šuma, a i svi bitni rubovi su prisutni).

Primjeri će biti prikazani na presjeku ruba u jednom retku slike. Na apscisi je položaj piksela unutar retka, a na ordinati je njegova svjetlina. Tipičan primjer ruba, uz prisustvo šuma, prikazuje slika 2.6.



Slika 2.6 Presjek ruba uz prisustvo šuma

2.2.1 Konvolucija

Diskretna konvolucija je vrlo česta operacija kod detekcije rubova (koristi se i u Cannyjevom algoritmu) pa će ovdje biti ukratko opisana. Prvo će se opisati jednodimenzionalna, a zatim i dvodimenzionalna konvolucija.

Neka su I i Q vektori duljine m_I, odn. m_Q. I obično čini jedan redak ili stupac slike. Neka je I_i vrijednost i-tog elementa vektora (analogno Q_i), te I' rezultat konvolucije. Tada je jednodimenzionalna konvolucija dva vektora definirana sa

$$I'_i = \sum_{k=0}^{m_Q} I_{i-k} Q_k \tag{2.4}$$

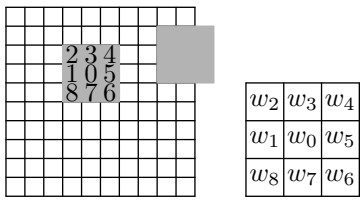
Tako npr. konvolucija nizova $I = (12345)$ i $Q = (-101)$ daje $(-2\ -2\ -2)$. Rezultat je manje dimenzije nego I zato što konvolucija nije definirana ukoliko je u gornjoj formuli $i - k < 0$.

Neka su sada I (*matrica slike*) i Q (*matrica maske*) matrice dimenzija $m_I \times n_I$ i $m_Q \times n_Q$. Obično je Q *kvadratna* matrica pa je $m_Q = n_Q$. Neka I_{ij} označava vrijednost piksela u i-tom retku i j-tom stupcu (analogno Q_{ij} za vrijednost maske).

Konvolucija slike I s maskom Q je slika I' gdje je vrijednost piksela I'_{ij} definirana kao:

$$I'_{ij} = \sum_{k=0}^{\frac{m_Q-1}{2}} \sum_{l=0}^{\frac{n_Q-1}{2}} I_{i+k-\frac{m_Q-1}{2}, j+l-\frac{n_Q-1}{2}} Q_{kl} \tag{2.5}$$

Obje dimenzije matrice Q moraju biti *neparne* da bi ove sume bile definirane.



Slika 2.7 Računanje konvolucije

Na slici 2.7 dan je primjer dijela slike za računanje konvolucije. Ako su vrijednosti piksela $x_0 \ldots x_8$, a vrijednosti maske $w_0 \ldots w_8$, tada je vrijednost x_0 nakon konvolucije dana kao:

$$x_0' = \sum_{i=0}^{8} x_i w_i$$

Vidi se da za rubne piksele konvolucija nije definirana jer maska izlazi izvan slike ($w_4 \ldots w_6$). U takvom slučaju se konvoluciju ili ne računa ili se pikseli ''izvan slike'' uzimaju jednaki 0.

Konvolucija se obično radi na sivoj slici. Ukoliko je slika u boji, ovisi o primjeni kako naparviti konvoluciju. Ukoliko se želi npr. zamutiti sliku, tada će se napraviti konvolucija na svakom kanalu slike pojedinačno.

2.2.2 Cannyjev postupak

Cannyjev algoritam za detekciju rubova je sljedeći:

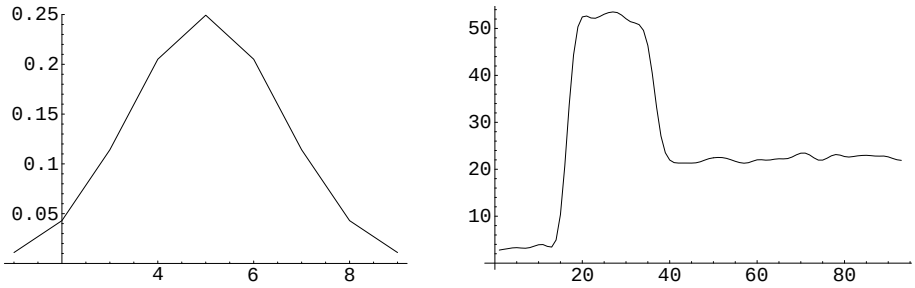
1. Konvolucija slike sa Gaussovim filterom standardne devijacije σ:

$$f(r) = \frac{1}{\sigma\sqrt{2\pi}} \exp \frac{-r^2}{2\sigma^2}$$

   U dvije dimenzije je $r^2 = x^2 + y^2$. Funkcija je prikazana na slici 2.10.

   Ovaj korak programski je izveden kao dvije jednodimenzionalne konvolucije (jedanput po retcima, jedanput po stupcima) prema formuli 2.4, umjesto kao dvodimenzionalna konvolucija prema fomuli 2.5. To je moguće zbog svojstva eksponencijalne funkcije.

   Ovaj korak izglađuje šum u ulaznoj slici i tako sprečava pojavu neželjenih rubova uzrokovanih šumom. Diskretna Gaussova funkcija (maska konvolucije) od 9 točaka sa $\sigma = 1.6$, kao i rezultat konvolucije, prikazuje slika 2.8.



Gaussova funkcija Rezultat konvolucije

Slika 2.8 Izglađivanje slike Gaussovom funkcijom

U programu se inače veličina konvolucijske jezgre odabire da bude jednaka $1+2.5\lceil\sigma\rceil$. Konstanta 2.5 je odabrana zato što je Gaussova funkcija na udaljenosti 2.5σ od maksimuma vrlo blizu 0 tako da je dobiveni rezultat konvolucije vrlo blizak onome koji bi se dobio konvolucijom sa beskonačnom jezgrom.

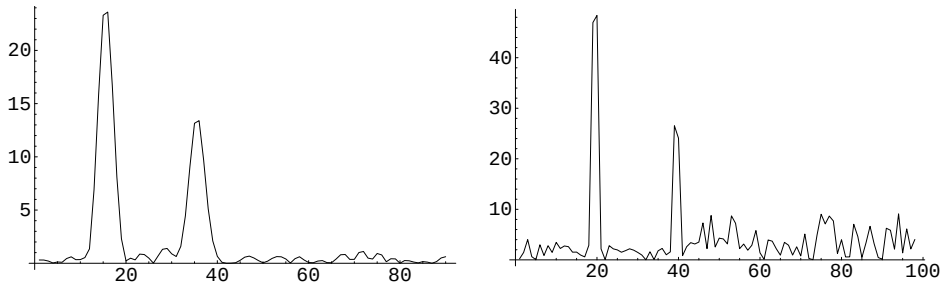
2. Računa se vertikalni i horizontalni gradijent konvolucijom sa maskama

$$G_x = (-1 \quad 0 \quad 1) \qquad G_y = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}$$

Gradijenti iz ovog koraka se sačuvaju zato što će biti potrebni za računanje smjera gradijenta kod detekcije objekata generaliziranom Houghovom transformacijom.

3. Iz vertikalne i horizontalne komponente se računa modul gradijenta: $G = \sqrt{G_x^2 + G_y^2}$. Ovo se računa za svaki piksel.

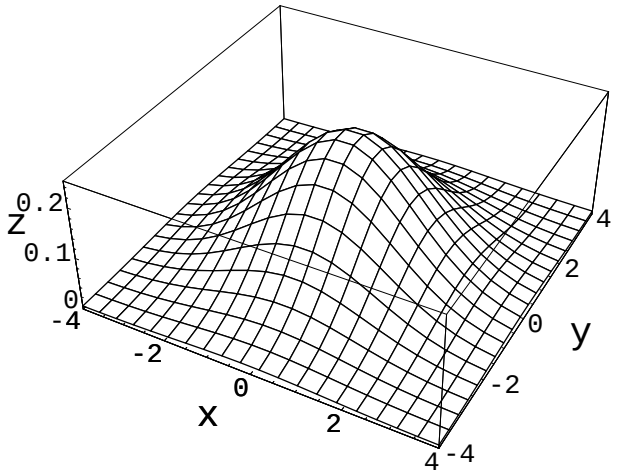
Rezultat gornja dva koraka prikazuje slika 2.9. Slika prikazuje usporedbu gradijenata dobivenih sa i bez izglađivanja ulaza.



Sa izglađivanjem Bez izglađivanja

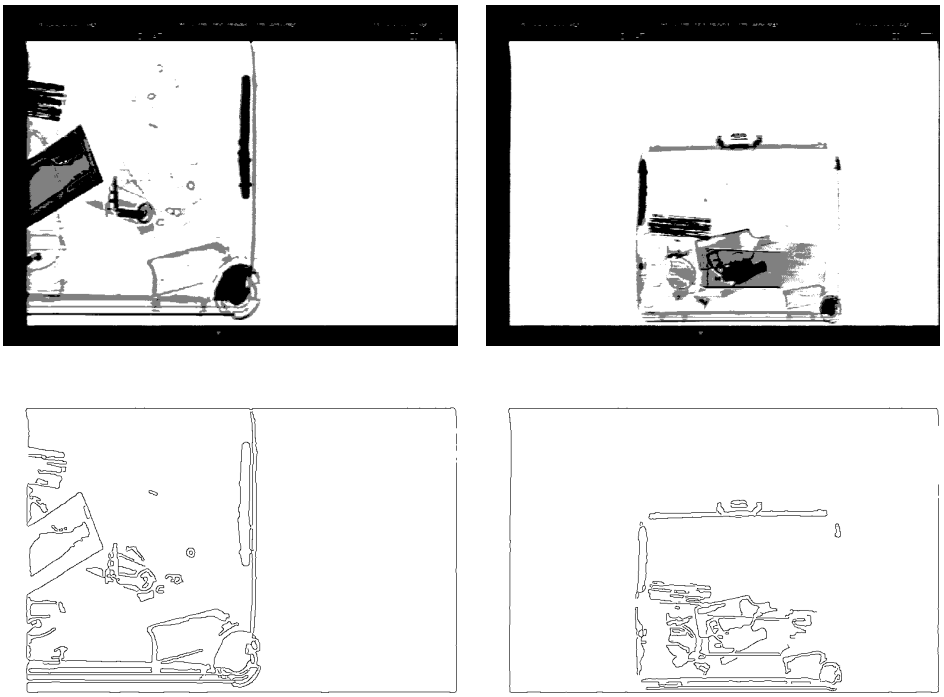
Slika 2.9 Gradijent slike

4. Potiskivanje ne-maksimuma. Time se osigurava da su rubovi široki 1 piksel.
5. Histereza rubova sa parametrima t_l i t_h. Time se dobiva binarna slika rubova. Odabrani su parametri $t_l = 0.25$ i $t_h = 0.4$. Ovaj korak pretvara sivu sliku rubova u crno-bijelu sliku.



Slika 2.10 Gaussova funkcija u dvije dimenzije

Slika 2.11 prikazuje primjenu Cannyjeve detekcije rubova na sliku sa klasificiranim bojama.



Slika 2.11

3 Računanje značajki i detekcija objekata

Problem je pronaći u sceni jedan ili više modela dopuštajući uz to djelomična preklapanja i afine transformacije.

Razmatralo se nekoliko različitih postupaka raspoznavanja invarijantnih na afine transformacije. Prva razmatrana metoda bili su Granlundovi koeficijenti koji se temelje na Fourierovoj transformaciji ruba na slici [14]. Ova metoda je odbačena zato što radi samo sa zatvorenim konturama objekata koje je teško dobiti na slikama gdje se objekti međusobno preklapaju. Sljedeća razmatrana metoda bile su momentne invarijante [14], a također je odbačena zbog problema sa preklapajućim objektima.

Posljednja, i vrlo ozbiljno razmatrana metoda, opisana je u [7]. Metoda je napravljena za raspoznavanje plosnatih objekata. Modeli se predstavljaju poligonalnim aproksimacijama, a također je potrebno i sve rubove u ulaznoj slici aproksimirati linijama. Linijski segment je predstavljen kao uređena četvorka (x, y, l, α) gjde su (x, y) koordinate polovišta linije, l je dužina, a α je nagib prema x-osi.

Ova metoda za svaki model generira i provjerava nekoliko hipoteza. Generiranje hipoteze uključuje predviđanje položaja objekta u sceni: to se ostvaruje poklapanjem privilegiranog segmenta modela (privilegirani segmenti su 10-ak najduljih segmenata modela) traženjem kompatiblinih segmenata u sceni. Kaže se da su segmenti modela M i scene S *kompatibilni* ukoliko je 1) kut između M i prethodnog susjeda od M približno jednak kutu između S i njegovog prethodnog susjeda, te 2) omjer duljina segmenata M i S je približno jednak a priori procjeni skaliranja modela u sceni. Nakon što se nađu kompatibilni segmenti procjenjuju se parametri transformacije koja model preslikava u objekt na slici.

Hipoteza se provjerava tako da se poklapaju dodatni segmenti modela i to počevši od segmenata najbližih M. Segmenti modela se transformiraju trenutnom transformacijom i za transformirani segment se traži poklapajući segment u sceni, te se računa mjera odudaranja. Ukoliko je mjera odudaranja mala, popravljaju se parametri transformacije. Postupak završava kada se nađe dovoljno (obično se zahtijeva da ukupna duljina poklopljenih segmenata bude barem 50% ukupne duljine modela) segmenata sa malom mjerom odudaranja.

Ova metoda nije upotrijebljena zato što je za računanje pozicije modela u sceni potrebno znati a priori statističke procjene srednjih vrijednosti i varijanci za rotaciju, translaciju i skaliranje objekta u sceni, što u ovom slučaju nije bilo moguće izračunati zbog malog uzorka.

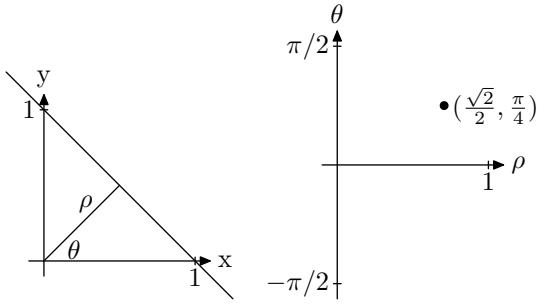
Na kraju je odabrana generalizirana Houghova transformacija kao općenita metoda invarijantna na rotaciju, translaciju i skaliranje, a također može detektirati i djelomično prekrivene objekte.

3.1 Houghova transformacija (HT)

Početno je Houghova transformacija nastala za detekciju *linija* u slici: za danu sliku mogu se odrediti parametri implicitno zadanog pravca [6]. Kasnije je transformacija generalizirana na detekciju jednostavnih *krivulja* koje se mogu opisati s malo parametara. Još kasnije je transformacija generalizirana na detekciju *proizvoljnih oblika* a može se načiniti i da detektira skalirane i zarotirane oblike.

3.1.1 Detekcija analitičkih oblika

Radi izlaganja ideje Houghove transformacije prvo će se opisati originalna Houghova transformacija (za detekciju linija i ostalih analitički opisivih krivulja s malo parametara kao što su kružnice i parabole). Osnovna ideja je računanje mjesta u parametarskom prostoru koje odgovara skupu krivulja koje prolaze kroz piksel -- element ruba u izvornoj slici.



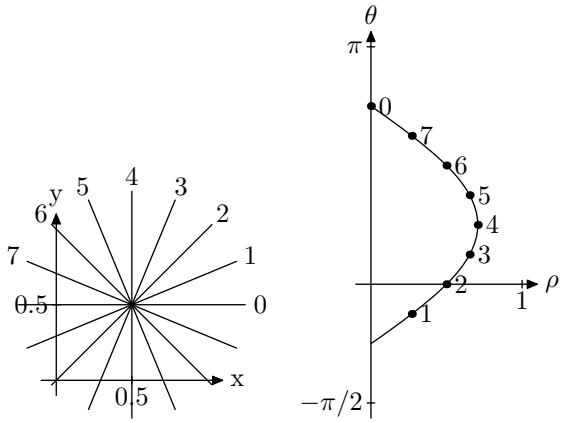
Slika 3.1 Parametri pravca i transformacija

Neka se za primjer transformira pravac iz Kartezijevog koordinatnog sustava u parametarski prostor. Prvo se zapisuje jednadžba pravca u implicitnom obliku

$$x\cos\theta + y\sin\theta - \rho = 0$$

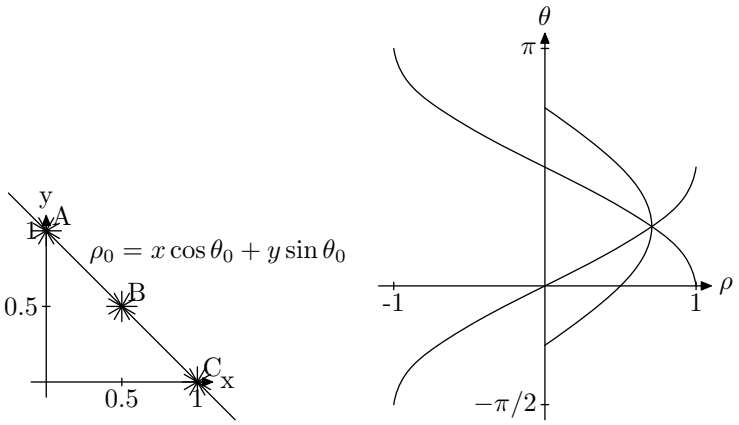
gdje je ρ udaljenost pravca od ishodišta, a θ kut što ga normala na pravac čini s osi x. Transformacija pravca je jedna *točka* u transformiranom prostoru. Tako se pravac $x + y - 1 = 0$ transformira u točku $(\sqrt{2}/2, \pi/4)$ (vidi sliku 3.1).

Neka je sada u Kartezijevom sustavu zadana točka (x', y'); treba naći u što se preslikava porodica pravaca koja prolazi kroz tu točku u parametarskom prostoru $\rho - \theta$. Kroz jednu točku prolazi porodica pravaca i svaki pravac zadovoljava jednadžbu $Ax' + By' + C = 0$. U parametarskom prostoru tome odgovaraju sve točke za koje vrijedi $\rho = x'\cos\theta + y'\sin\theta$. Tako npr. pravcu $y = 0.5$ koji prolazi točkom $(0.5, 0.5)$ odgovara točka 6 na krivulji, a pravcu $x = 0.5$ odgovara točka 2 (vidi sliku 3.2).



Slika 3.2 Primjer preslikavanja pravca

Houghova transformacija porodica pravaca koje prolaze kroz tri različite *kolinearne* točke A, B i C čine tri krivulje u transformiranom prostoru koje se *sijeku u točki* koja odgovara pravcu na kojem leže A, B i C (prikazano na slici 3.3). To se, uz dobar algoritam, može iskorisiti za detekciju pravaca na slici [2].

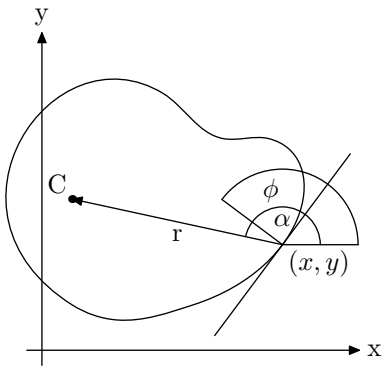


Slika 3.3 Određivanje parametara pravca

3.1.2 Generalizacija na proizvoljne oblike

Generalizaciju Houghove transformacije predložio je Ballard 1981. godine [3]. Neka silueta ima složeni analitički oblik (slika 3.4). Izabere se neka točka (x_c, y_c) u silueti i proglasi se *referentnom točkom* C. Povuče se dužina između C i nekog rubnog piksela (x, y). U tom pikselu izračunava se smjer gradijenta (kut između osi x i normale na krivulju).

Koordinate referentne točke mogu se izraziti kao funkcija smjera gradijenta (ϕ) i koordinata piksela na slici:



Slika 3.4 Parametri potrebni za GHT

$$\begin{aligned} x_c &= x + r\cos(\alpha) \\ y_c &= y + r\sin(\alpha) \end{aligned} \tag{3.1}$$

Koordinate referentne točke mogu se na jednak način izračunati za sve rubne piksele -- one su funkcija smjera gradijenta u tim točkama. Gradijent se dobije kao usputni rezultat prilikom detekcije rubova Cannyjevim algoritmom.

Za *poznati oblik* siluete, orijentaciju i veličinu objekta, neposredno prije traženja računa se tzv. tablicu R. Tablica R ima sljedeće elemente:

- Smjer gradijenta u svim pikselima siluete.
- Smjeru gradijenta pridružene dvojke (r, α). Pri tome se za svaki smjer gradijenta u tablici se može pojaviti više dvojki.

3.1.3 Opis GHT algoritma

Ovdje će se opisati varijanta algoritma koja uzima u obzir i moguću rotaciju i skaliranje objekta koji se traži u sceni. Moguć raspon vrijednosti skaliranja, i kuta rotacije, kao i rezolucije za ta dva parametra moraju biti a priori poznate. U ovom radu skaliranje nije uzeto u obzir (kasnije će se objasniti i zašto) a vrijednosti rotacije 0--360^o u 16 koraka (veličina koraka je 22.5^o).

1. Formira se tablica R za traženi oblik. Prilikom računanja, smjer gradijenta ϕ normaliziran je na interval $[0, \pi)$ zato što su gradijenti u intervalu $[\pi, 2\pi)$ istog smjera, ali suprotne orijentacije. Za potrebe raspoznavanja bitan je isključivo *smjer*.
2. Oblikuje se četverodimenzionalno polje brojila $C(x_1 : x_2; y_1 : y_2; s_1 : s_2; \theta_1 : \theta_2)$ koje odgovara mogućim položajima referentne točke. Vrijednost svih brojila inicijalizira se na 0.
3. Za svaku vrijednost rotacije θ i skaliranja s, te za svaki rubni piksel slike:
   - Računa se smjer gradijenta ϕ prema implicitnoj formuli

$$\tan\phi = \frac{g_y}{g_x}$$

gdje su g_x i g_y komponente gradijenta u x i y smjeru. Ukoliko ϕ izlazi iz intervala $[0, \pi)$, po potrebi se doda π da upadne u taj interval.

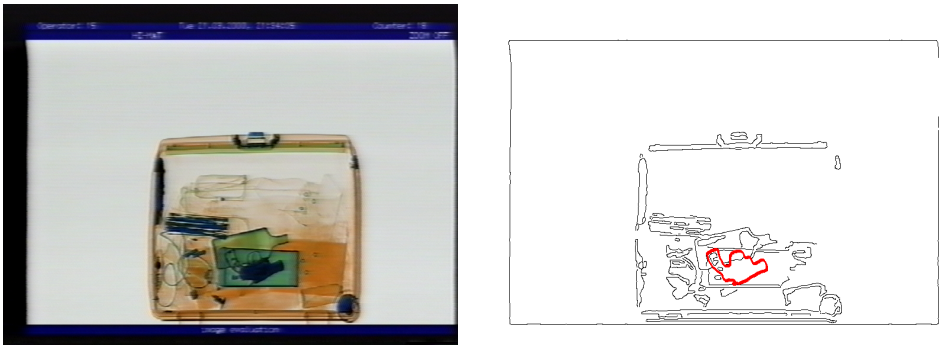
- Traže se retci tablice u kojima je vrijednost ϕ najbliža vrijednosti $\phi - \theta$.
- Za sve r i α (određene retcima iz prethodnog koraka) se na temelju modificiranih jednadžbi

$$x_c = x + s \cdot r\cos(\alpha + \theta)$$
$$y_c = y + s \cdot r\sin(\alpha + \theta)$$

izračunaju moguće koordinate referentne točke te se pripadajuća brojila $C(x_c, y_c, s, \theta)$ povećaju za 1.

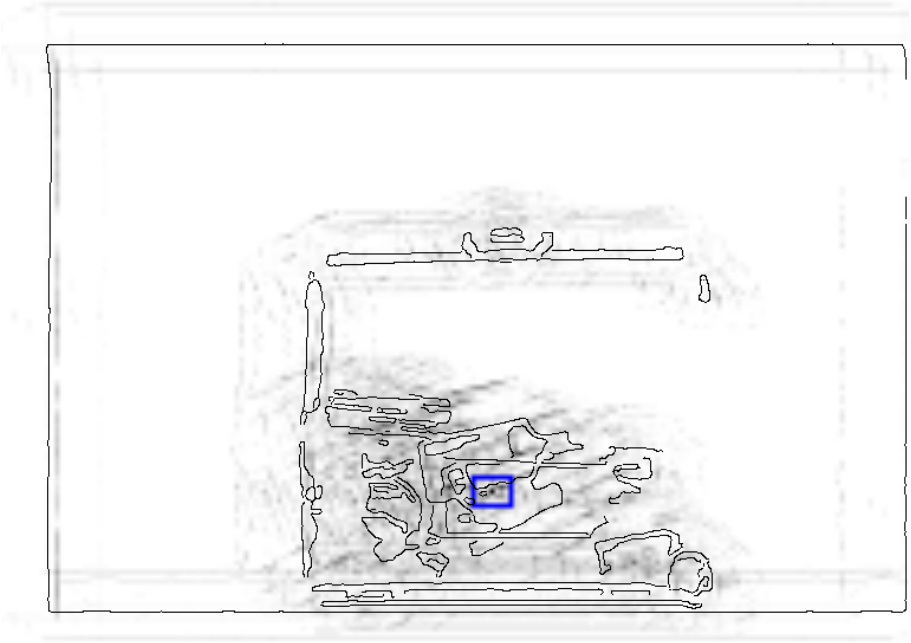
4. Moguće lokacije traženog oblika u slici određene su lokalnim maksimumima u polju C.

Dobar pregled ostalih istraživanja na području Houghove transformacije, kao i vrlo iscrpan izvor referenci, može se naći u [1]. Slika i rezultat raspoznavanja dani su na slici 3.5. Predmet na slici je skaliran za faktor 0.5 u odnosu na model.

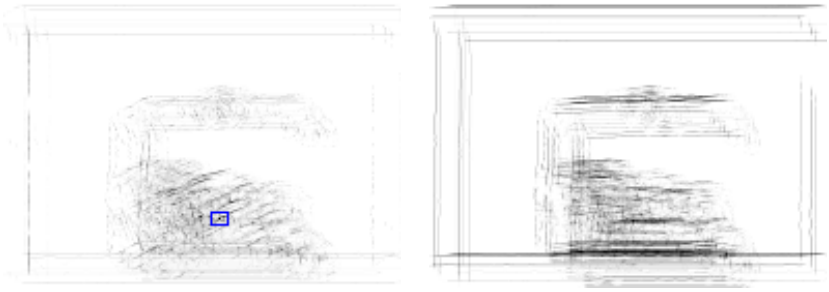


Slika 3.5 Uspješan rezultat raspoznavanja

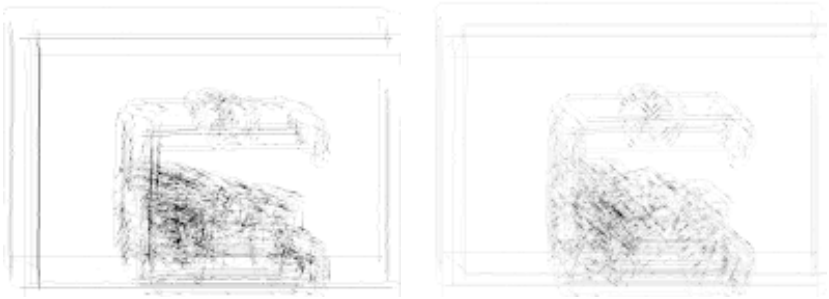
Slike 3.7 i 3.8 prikazuju sadržaje akumulatora generalizirane Houghove transformacije za iznos skaliranja 0.5 i 16 različitih kuteva rotacije. Tamnija područja imaju veće iznose akumulatora. Područje apsolutnog maksimuma je zbog slabe uočljivosti posebno označeno; u sredini kvadrata je jedan piksel koji odgovara položaju referentne točke. Slika 3.6 prikazuje superponiranu sliku rubova i akumulatora za iznos rotacije od 0^o. U ovom slučaju raspoznavanje se može smatrati uspješnim.



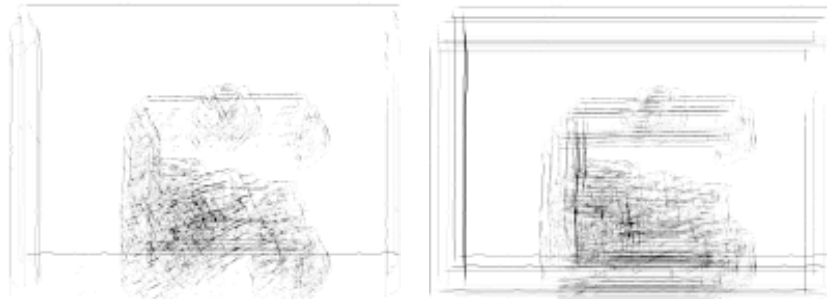
Slika 3.6 Uspješno raspoznavanje: maksimalni iznos akumulatora poklapa se sa referentnom točkom prisutnog objekta



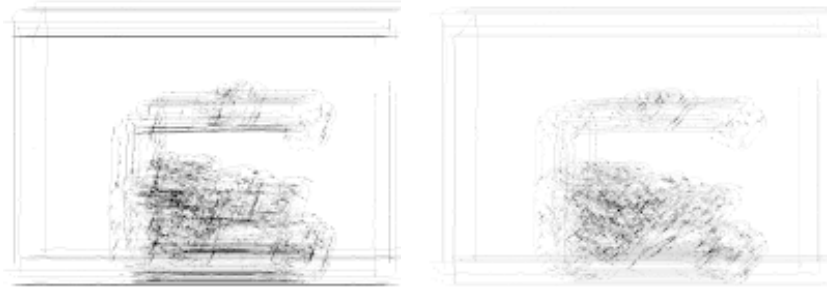
0^o 22.5^o



45^o 67.5^o

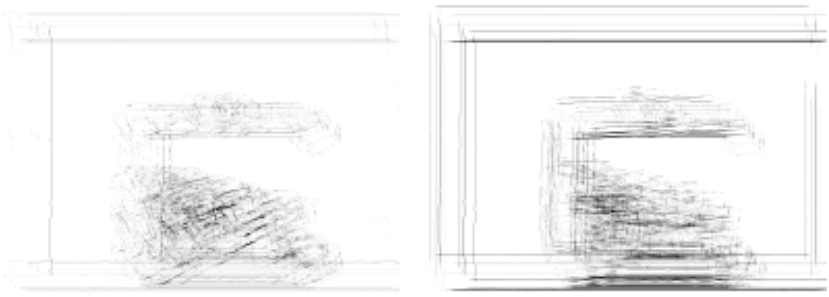


90^o 112.5^o

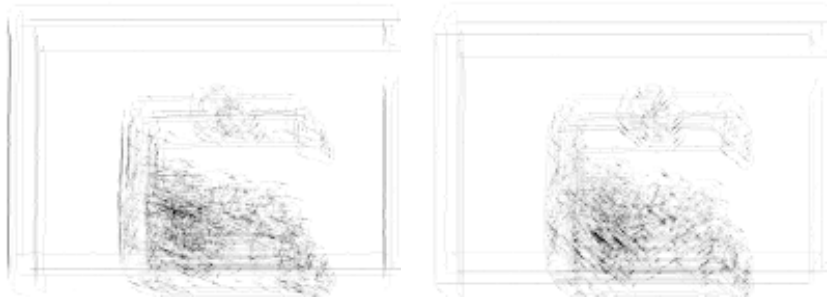


135^o 157.5^o

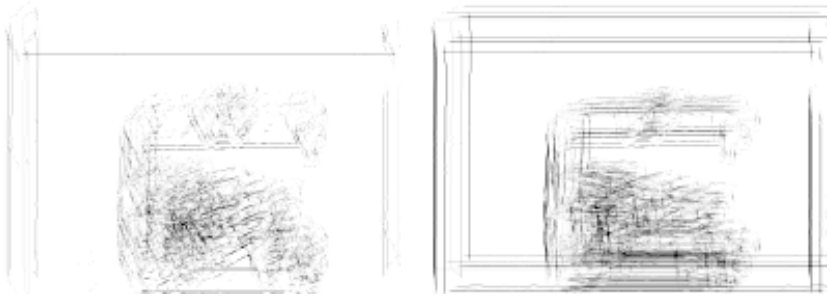
Slika 3.7



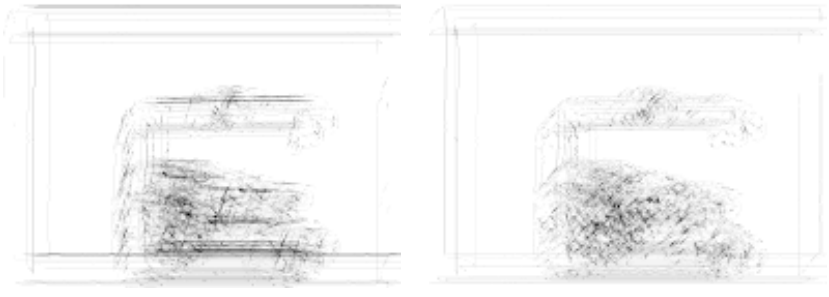
180^o 202.5^o



225^o 247.5^o



270^o 292.5^o



315^o 337.5^o

Slika 3.8

3.1.4 Provjera hipoteze

Lokalni maksimumi izračunati generaliziranom Houghovom transformacijom mogu se smatrati tek početnim hipotezama o položaju objekta u sceni. Eksperimentima je utvrđeno da GHT algoritam griješi (i lažno prijavljuje prisutnost modela u sceni), pogotovo kada model ima puno ravnih segmenata (3,5 i 6 model) zato što se oni poistovjećuju sa rubovima slike i rubovima ostalih nezanimljivih objekata u slici.

Zato je, za svaki lokalni maksimum, model rotiran, skaliran i pozicioniran na scenu te je izračunata srednja vrijednost intenziteta piksela unutar transformirane konture. Ukoliko je srednja vrijednost ispod neke granice (tipično ispod 96 za intenzitete 0-255, kao u ovom slučaju) tada se hipoteza prihvaća.

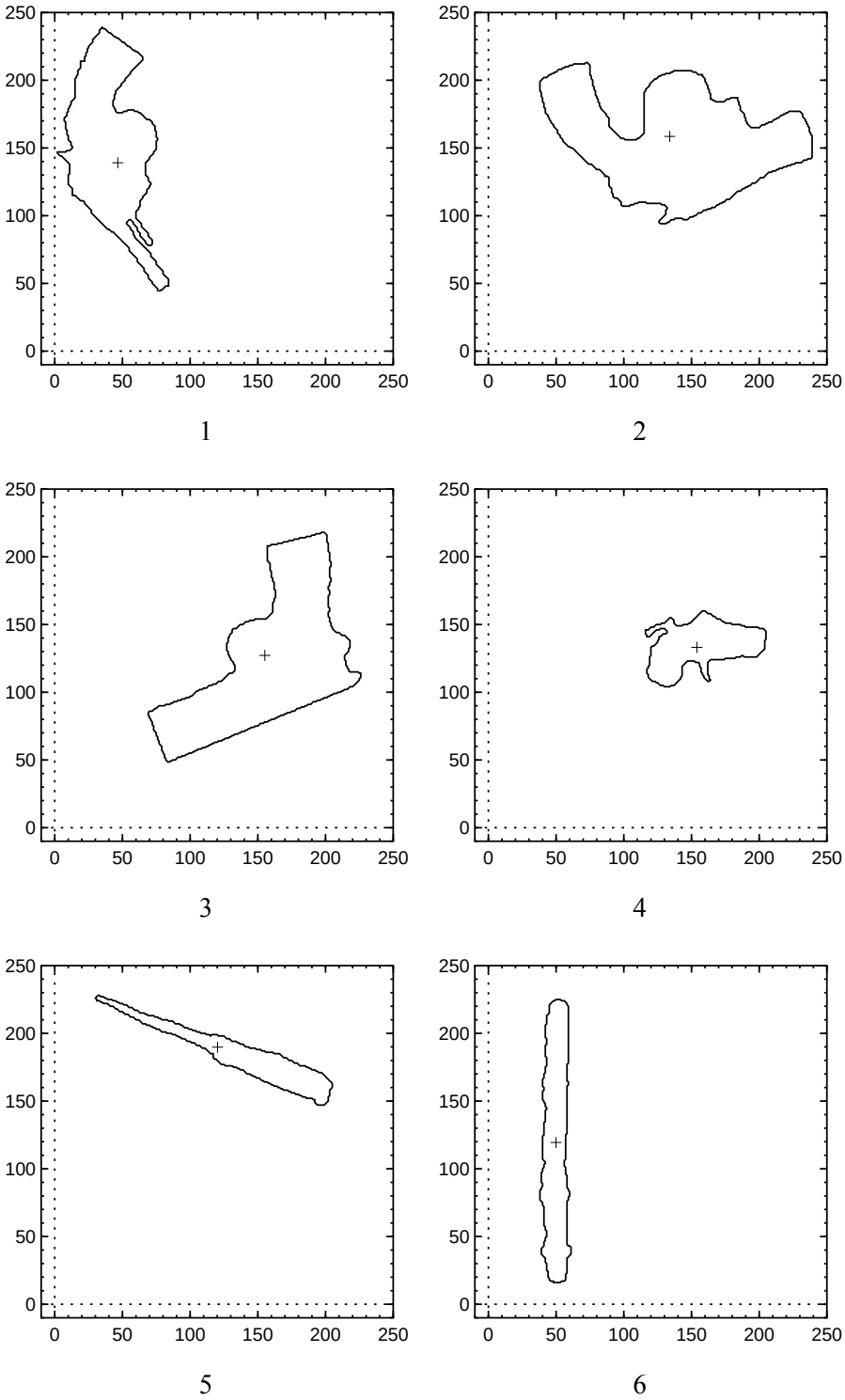
Bez provjere S provjerom

Slika 3.9 Utjecaj provjere na raspoznavanje

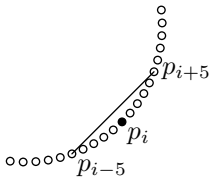
Slika 3.9 prikazuje, od programa predložene, položaje objekata prema 8 najvećih lokalnih maksimuma sa i bez provjere na originalnoj slici.

3.2 Modeli

Slika 3.10 prikazuje modele koji se koriste za raspoznavanje. Napravljeni su tako da su objekti snimljeni običnom kamerom na pozadini s kojom čine visoki kontrast. Na takve slike je primijenjena Cannyjeva detekcija rubova i rezultat je naknadno ručno obrađen tako da je ostao vidljiv samo *vanjski rub* objekta. Model 5 je nož, a model 6 je pištolj 2 postavljen vertikalno. Napisan je pomoćni program koji je pratio rub objekta te dobivenu listu točaka zapisao u datoteku. Program za raspoznavanje učitava točke ruba iz datoteke te na temelju njih stvara tablicu R. Referentna točka modela označena je križićem.



Slika 3.10 Modeli za raspoznavanje



Slika 3.11 Računanje gradijenta

Računanje gradijenta krivulje zadane nizom točaka, umjesto analitički, prikazano je na slici 3.11. Ukoliko se želi odrediti gradijent u točki p_i, povuče se sekanta koja prolazi kroz točke p_{i-5} i p_{i+5}. Nagib sekante aproksimira nagib tangente te dodavanjem $\pi/2$ dobije aproksimirani smjer gradijenta. Ukoliko krivulja ima veliku zakrivljenost u nekoj točki, treba povući sekantu kroz bliže susjede točki p_i.

Tablica u nastavku prikazuje tablicu točaka (x i y koordinate) za 4. model sa slike 3.10.

130 146	121 116	116 104	149 99	184 100	203 119	168 126	155 128
129 145	122 116	117 104	150 98	185 100	202 120	167 126	154 128
128 145	123 115	118 104	151 97	186 100	201 121	166 126	153 128
127 145	124 114	119 103	152 96	187 101	200 122	165 126	152 127
126 144	125 113	120 102	153 95	188 101	199 123	164 126	151 127
125 143	125 112	121 102	154 94	189 101	198 124	163 127	150 127
124 143	126 111	122 102	155 93	190 102	197 124	162 128	149 127
123 142	126 110	123 101	156 92	191 102	196 124	162 129	148 127
122 142	126 109	124 100	157 91	192 102	195 124	162 130	147 127
121 142	127 109	125 100	158 90	193 102	194 124	162 131	146 128
120 141	128 108	126 100	159 90	194 102	193 124	162 132	145 129
119 140	129 107	127 100	160 90	195 102	192 124	162 133	144 130
118 139	130 107	128 99	161 91	196 102	191 124	162 134	143 131
118 138	131 107	129 99	162 92	197 102	190 124	162 135	143 132
118 137	132 106	130 99	163 92	198 102	189 124	162 136	143 133
117 136	132 105	131 98	164 92	199 102	188 123	162 137	143 134
117 135	131 104	132 97	165 93	200 103	187 123	163 138	143 135
117 134	130 103	133 96	166 94	201 103	186 124	163 139	143 136
117 133	129 103	134 95	167 94	202 103	185 124	163 140	143 137
118 132	128 103	135 95	168 95	203 103	184 124	164 141	143 138
118 131	127 103	136 96	169 96	204 104	183 124	163 142	142 139
119 130	126 104	137 97	170 96	205 105	182 125	162 142	142 140
119 129	125 104	137 98	171 96	205 106	181 125	161 141	141 141
119 128	124 104	137 99	172 97	205 107	180 125	160 140	140 142
119 127	123 105	138 100	173 98	205 108	179 125	159 139	139 143
119 126	122 106	139 101	174 98	205 109	178 125	159 138	138 144
119 125	121 107	140 101	175 99	205 110	177 125	158 137	137 145
119 124	120 108	141 101	176 99	205 111	176 125	158 136	136 145
120 123	119 109	142 101	177 99	204 112	175 125	157 135	135 145
120 122	118 109	143 101	178 99	204 113	174 126	157 134	134 146
120 121	117 109	144 100	179 99	204 114	173 126	157 133	133 146
120 120	117 108	145 100	180 100	204 115	172 126	157 132	132 146
120 119	116 107	146 100	181 100	204 116	171 126	156 131	131 146
120 118	116 106	147 100	182 100	204 117	170 126	156 130	130 146
120 117	116 105	148 99	183 100	204 118	169 126	156 129	

Sljedeća tablica prikazuje tablicu R za isti model. Za referentnu točku uzeto je težište pištolja $(153.957, 116.907)$. Brojevi su redom ϕ (u radijanima), r (u pikselima) i α (u radijanima).

0 51.6518 2.98791	0.19635 34.4993 -0.177549	0.19635 23.8723 -1.9593
0 22.9499 -1.97581	0.19635 34.3368 -0.14888	0.19635 19.4691 -0.973041
0.19635 40.7185 -0.433213	0.19635 34.2028 -0.119962	0.19635 20.3035 -1.00076
0.19635 40.3089 -0.410695	0.19635 24.5167 0.461032	0.19635 21.1523 -1.02628
0.19635 38.9963 -0.397421	0.19635 39.7807 0.303971	0.19635 22.0138 -1.04981
0.19635 38.6202 -0.373544	0.19635 51.5082 3.0071	0.19635 22.8865 -1.07156
0.19635 37.3286 -0.358377	0.19635 51.3837 3.02638	0.392699 34.0976 -0.0908408
0.19635 36.9897 -0.333058	0.19635 50.283 3.04385	0.392699 34.0214 -0.0615646
0.19635 36.6749 -0.307287	0.19635 50.1953 3.06368	0.392699 33.9746 -0.0321824
0.19635 36.3849 -0.281087	0.19635 50.1274 3.08357	0.392699 25.0034 0.407395
0.19635 36.1203 -0.254485	0.19635 50.0793 3.10351	0.392699 50.0512 3.12347
0.19635 35.8816 -0.22751	0.19635 16.2268 -2.08939	0.392699 50.0431 -3.13973
0.19635 35.6694 -0.200194	0.19635 17.1025 -2.06041	0.392699 50.055 -3.11975

0.392699 14.5236 -2.1577
0.392699 15.3663 -2.12166
0.392699 24.8006 -1.94403
0.392699 18.651 -0.942862
0.392699 23.7693 -1.09171
0.392699 25.1213 -1.07473
0.589049 33.9571 -0.00274444
0.589049 32.9695 0.0275079
0.589049 31.9698 0.0283682
0.589049 31.0157 0.0615176
0.589049 30.0977 0.0967299
0.589049 28.5742 0.20822
0.589049 26.8514 0.377879
0.589049 25.9245 0.392111
0.589049 40.0914 0.327772
0.589049 49.0877 -3.09894
0.589049 48.1425 -3.0773
0.589049 13.7022 -2.19813
0.589049 16.4491 -0.920628
0.589049 17.8514 -0.909947
0.589049 26.0051 -1.09303
0.589049 27.3563 -1.07738
0.785398 29.2193 0.134108
0.785398 29.3698 0.167857
0.785398 28.7975 0.242201
0.785398 29.0536 0.275622
0.785398 28.0926 0.28531
0.785398 27.4426 0.330548
0.785398 39.1459 0.335996
0.785398 38.2033 0.344627
0.785398 20.9982 1.47746
0.785398 21.9277 1.52714
0.785398 22.9068 1.57267
0.785398 47.2208 -3.0548
0.785398 13.5517 -2.30137
0.785398 26.1026 -1.96574
0.785398 15.049 -0.933297
0.785398 28.7135 -1.0632
0.785398 30.076 -1.05031
0.981748 37.6875 -0.881915
0.981748 37.5776 -0.844447
0.981748 38.2491 -0.8249
0.981748 38.9346 -0.806035
0.981748 32.6191 0.406811
0.981748 33.1679 0.367182
0.981748 33.767 0.3289
0.981748 37.6217 0.378631
0.981748 37.0849 0.413661
0.981748 18.8717 1.24965
0.981748 18.5802 1.30074
0.981748 19.3164 1.36449
0.981748 20.1252 1.42333
0.981748 23.9295 1.6144
0.981748 24.9905 1.65264
0.981748 46.3239 -3.03142
0.981748 45.4533 -3.00713
0.981748 13.548 -2.40579
0.981748 12.2586 -0.967305
0.981748 13.6518 -0.948563
0.981748 31.4431 -1.03853
0.981748 32.8141 -1.02774
1.1781 31.7032 0.419292
1.1781 34.4138 0.292005
1.1781 35.1057 0.256518
1.1781 36.1715 0.424774
1.1781 35.2628 0.436461
1.1781 34.8046 0.474655
1.1781 34.3986 0.513808
1.1781 32.6716 0.5439
1.1781 31.8201 0.560163
1.1781 31.5344 0.603891
1.1781 30.7165 0.622379
1.1781 29.9097 0.641871
1.1781 29.7404 0.688952
1.1781 18.6857 1.13091
1.1781 18.2822 1.18042
1.1781 26.0849 1.68772
1.1781 44.6105 -2.98191
1.1781 43.6236 -2.97827
1.1781 15.0904 -2.49486
1.1781 14.305 -2.45273
1.1781 26.6729 -1.91669
1.1781 33.3419 -1.00206
1.1781 33.8909 -0.977191
1.1781 35.2802 -0.96955
1.37445 30.3328 0.476278
1.37445 30.7925 0.432513
1.37445 35.84 0.222444
1.37445 33.5313 0.528467
1.37445 29.6376 0.736465
1.37445 29.6022 0.7842
1.37445 19.3153 0.967611
1.37445 19.621 1.03855
1.37445 19.1329 1.08361
1.37445 27.2089 1.71994
1.37445 27.3753 1.75607
1.37445 42.6372 -2.97445
1.37445 41.6514 -2.97046
1.37445 40.6664 -2.96627
1.37445 33.6001 -2.95924
1.37445 32.8187 -2.92374
1.37445 31.8431 -2.91695
1.37445 30.869 -2.90974
1.37445 29.8967 -2.90205
1.37445 29.1875 -2.86063
1.37445 28.2281 -2.8508
1.37445 27.2716 -2.84029
1.37445 26.3183 -2.82902
1.37445 25.3686 -2.81689
1.37445 24.4229 -2.80383
1.37445 23.4818 -2.78972
1.37445 22.5457 -2.77443
1.37445 22.0093 -2.71568
1.37445 21.1027 -2.6961
1.37445 20.2049 -2.67477
1.37445 19.3171 -2.65147
1.37445 18.4408 -2.62594
1.37445 16.73 -2.56696
1.37445 15.8999 -2.53277
1.37445 10.8411 -1.19717
1.37445 11.2447 -1.11428
1.37445 11.72 -1.03762
1.37445 35.8554 -0.94655
1.5708 36.8161 0.216451
1.5708 29.6343 0.83194
1.5708 27.5771 1.79172
1.5708 37.716 -2.9524
1.5708 36.7343 -2.94728
1.5708 35.7537 -2.94188
1.5708 34.584 -2.96448
1.5708 17.5778 -2.59789
1.5708 36.4489 -0.924286
1.5708 37.0599 -0.902747
1.76715 29.4477 0.491847
1.76715 28.9703 0.85746
1.76715 20.5161 0.887243
1.76715 19.8996 0.926215
1.76715 26.8471 1.83624
1.76715 26.1733 1.88315
1.76715 27.5534 2.38536
1.76715 29.1828 2.48097
1.76715 29.9787 2.50144
1.76715 30.7865 2.52084
1.76715 31.0496 2.56579
1.76715 31.893 2.58286
1.76715 32.7452 2.59905
1.76715 33.6056 2.61442
1.76715 34.4735 2.62901
1.76715 35.3484 2.64289
1.76715 36.2298 2.65609
1.76715 36.6724 2.69294
1.76715 37.576 2.70448
1.76715 38.4843 2.71548
1.76715 39.004 2.74943
1.76715 39.9299 2.759
1.76715 40.8593 2.76814
1.76715 41.792 2.77687
1.76715 42.7277 2.78522
1.76715 43.6663 2.79321
1.76715 44.6075 2.80086
1.76715 45.5512 2.8082
1.76715 46.4973 2.81524
1.76715 47.4456 2.82199
1.76715 48.0974 2.84827
1.76715 39.6821 -2.96188
1.76715 38.6986 -2.95726
1.76715 26.3507 -1.88098
1.76715 10.5174 -1.2858
1.76715 37.6875 -0.881915
1.9635 38.979 -0.733424
1.9635 39.7275 -0.716573
1.9635 28.5701 0.508378
1.9635 37.7933 0.210769
1.9635 27.5599 0.861154
1.9635 26.15 0.865247
1.9635 21.8343 0.816163
1.9635 21.1619 0.850596
1.9635 26.4976 1.91907
1.9635 25.8796 2.05482
1.9635 26.3599 2.08841
1.9635 26.9733 2.25473
1.9635 26.8669 2.30712
1.9635 26.8347 2.35978
1.9635 27.6308 2.43654
1.9635 28.3998 2.45936
1.9635 49.0555 2.85416
1.9635 11.1344 -1.48474
1.9635 10.2812 -1.37928
2.15985 39.8498 -0.681161
2.15985 40.6315 -0.665662
2.15985 27.7008 0.525951
2.15985 38.0151 0.236495
2.15985 25.397 0.839712
2.15985 24.6616 0.812635
2.15985 23.2479 0.814292
2.15985 26.8554 1.95408
2.15985 26.3341 2.00352
2.15985 26.0214 2.14084
2.15985 25.7563 2.19451
2.15985 26.3529 2.22531
2.15985 50.0153 2.85983

2.15985 25.1015 -1.8552
2.15985 11.0933 -1.57467
2.35619 38.9311 -0.769708
2.35619 41.4226 -0.650752
2.35619 41.636 -0.617087
2.35619 41.8961 -0.583803
2.35619 26.3365 0.512174
2.35619 50.9766 2.86528
2.35619 51.6806 2.88918
2.35619 23.8708 -1.82674
2.35619 22.6614 -1.79521
2.35619 12.2646 -1.73816
2.35619 11.1421 -1.66454
2.55254 42.2021 -0.550965
2.55254 41.6873 -0.530525
2.55254 52.4134 2.91242
2.55254 21.6877 -1.80547
2.55254 14.2405 -1.71476
2.55254 13.2516 -1.72558
2.74889 39.2285 0.255305
2.74889 52.1953 2.93108
2.74889 20.4959 -1.76936
2.74889 19.5166 -1.77947
2.74889 18.3473 -1.73742
2.74889 16.3784 -1.75768
2.74889 15.3969 -1.76974
2.94524 41.1903 -0.509583
2.94524 41.5977 -0.476865
2.94524 24.9776 0.496896
2.94524 51.9955 2.94989
2.94524 51.8143 2.96884
2.94524 17.9911 -2.03427
2.94524 18.8909 -2.0106
2.94524 19.8003 -1.98909
2.94524 20.7181 -1.96949
2.94524 21.6432 -1.95155
2.94524 17.3619 -1.74698
3.14159 41.1483 -0.455272
3.14159 39.4929 0.279808

3.3 Eksperimentalni rezultati

Za svaku od 28 slika program je računao GHT za svaki od 6 modela sa slike 3.10, tražio lokalne maksimume (uz provjeru na originalnoj slici, odjeljak 3.1.4) te odabrao najvećih 8 maksimuma između svih modela. Lokalni maksimum je prihvaćen kao kandidat za prisutnost objekta u slici ako je iznosio barem 55% duljine modela (u pikselima) i ako je provjera na originalnoj slici dala srednju vrijednost intenziteta ≤ 96. Ove granice određene su empirijski.

Sažetak parametara za sve korake algoritma:

1. Kohonenova mreža
   - Učenje: Veličina 4×4, $\eta_0 = 0.4$, $\sigma_0 = 16$, $\tau_1 = 5000$, $\tau_2 = 15000$.
   - Raspoznavanje: Razredi 0, 1, 2, 3, 5 i 7 obojani su crno, 4 i 6 sivo a ostali bijelo.
2. Detekcija rubova: $\sigma = 1.6$, $t_l = 0.25$, $t_h = 0.4$.
3. GHT: Mogući kutevi rotacije $[0, 2\pi]$ kvantiziraju se u 16 koraka. Mogući položaji referentne točke kvantizirani su u omjeru 4:1 (4×4 područje ulazne slike ima 1 zajednički akumulator). Detekcija modela prihvaćena je ukoliko je detektirana duljina ruba iznosila barem 55% duljine konture modela (uključujući skaliranje) te ukoliko srednja vrijednost piksela unutar transformiranog modela nije prelazila 96 (maksimalna moguća vrijednost 255). Moguće vrijednosti skaliranja uvijek su bile postavljene na 1.

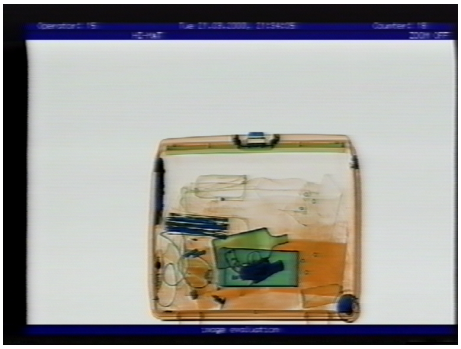
Tablica prikazuje rezultate raspoznavanja za svaku pojedinu sliku. Za svaki od 8 maksimuma naveden je predloženi model M, iznos akumulatora A kao omjer duljine konture objekta te položaj referentne točke T. Iznos akumulatora može iznositi i više od 1 zato što jednoj ćeliji doprinosi 16 piksela sa slike, kao i okolni šum. Ukoliko je neki model ispravno detektiran na slici, tada je on naveden podebljano. Broj iza slike navodi model stvarno prisutan na slici ili - ukoliko nema zanimljivih objekata. Slovo iza modela je V ukoliko je objekt ‘‘veliki’’ ili M ukoliko je ‘‘mali’’ (ovisno o stupnju povećanja koji je odabrao operater). Slike označene sa * su u bitno različitoj paleti, za koju neuronska mreža

nije bila trenirana. Na slikama označenima sa # nalaze se modeli u raznim položajima (ne frontalnim), a koji nisu među modelima za traženje (bilo zato što nisu cijeli, bilo zato što su premali da bi se napravio dobar model).

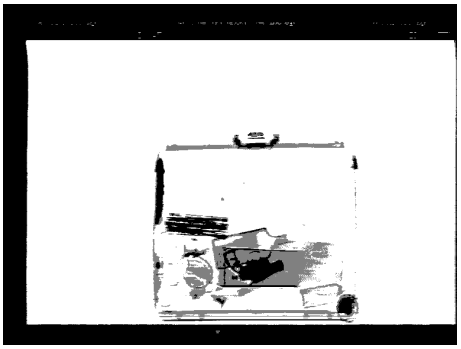
slika	1	2	3	4	5	6	7	8
s01.png/4M	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.66	M6 A0.66
	T(236,52)	T(704,52)	T(696,52)	T(684,52)	T(252,52)	T(224,52)	T(672,52)	T(668,52)
s02.png/4V	M5 A1.23	M5 A1.15	M5 A1.02	**M4 A1.00**	M5 A0.84	M5 A0.84	M5 A0.82	M5 A0.82
	T(376,276)	T(396,272)	T(412,268)	**T(460,388)**	T(420,264)	T(384,288)	T(436,264)	T(368,296)
s03.png/4V	M5 A1.19	M5 A1.19	M5 A0.96	M6 A0.8	M5 A0.78	**M4 A0.77**	M5 A0.76	M5 A0.75
	T(372,284)	T(384,280)	T(396,316)	T(108,136)	T(376,292)	**T(460,388)**	T(360,304)	T(364,332)
s04.png/4V	**M4 A0.95**	M6 A0.68	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.66
	T(100,388)	T(180,52)	T(704,52)	T(696,52)	T(684,52)	T(252,52)	T(224,52)	T(240,52)
s05.png/-	M5 A0.82	M5 A0.8	M5 A0.77	M5 A0.75	M5 A0.72	M5 A0.71	M5 A0.71	M5 A0.7
	T(396,452)	T(472,400)	T(420,440)	T(452,408)	T(372,464)	T(404,444)	T(464,404)	T(380,460)
s06.png/-	M5 A1.42	M5 A1.35	M5 A1.31	M5 A1.29	M5 A1.25	M5 A1.25	M5 A1.23	M5 A1.21
	T(132,268)	T(124,272)	T(140,264)	T(104,312)	T(112,276)	T(192,288)	T(184,292)	T(148,260)
s07.png/-	M5 A1.32	M5 A1.23	M5 A1.2	M5 A1.19	M5 A1.18	M5 A1.17	M5 A1.14	M5 A1.09
	T(136,508)	T(152,500)	T(124,516)	T(116,520)	T(144,504)	T(128,512)	T(108,524)	T(160,496)
s08.png/3M	M6 A0.88	M6 A0.82	M6 A0.81	M6 A0.81	M6 A0.78	M5 A0.72	M6 A0.72	M5 A0.71
	T(356,504)	T(368,500)	T(348,508)	T(344,508)	T(376,496)	T(400,424)	T(336,512)	T(328,528)
s09.png/3V	M5 A1.49	M5 A1.43	**M3 A1.42**	M5 A1.37	M5 A1.34	M5 A1.3	M5 A1.28	M5 A1.26
	T(352,308)	T(360,304)	**T(372,408)**	T(324,300)	T(372,300)	T(336,296)	T(344,312)	T(316,304)
s10.png/3V	M5 A1.51	M5 A1.41	**M3 A1.41**	M5 A1.4	M5 A1.4	M5 A1.37	M5 A1.36	M5 A1.35
	T(352,308)	T(360,304)	**T(372,408)**	T(372,300)	T(360,332)	T(380,296)	T(380,324)	T(344,312)
s11.png/6M	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.66	M6 A0.66	M6 A0.66
	T(236,52)	T(696,52)	T(684,52)	T(252,52)	T(224,52)	T(672,52)	T(668,52)	T(664,52)
s12.png/6V	M5 A1.05	M5 A0.87	M5 A0.86	M6 A0.68	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.67
	T(404,224)	T(408,148)	T(404,196)	T(496,52)	T(236,52)	T(696,52)	T(684,52)	T(516,52)
s13.png/6V	M5 A0.84	M5 A0.81	M5 A0.78	M5 A0.75	M6 A0.68	M6 A0.67	M6 A0.67	M6 A0.67
	T(408,128)	T(404,196)	T(404,216)	T(412,212)	T(492,52)	T(236,52)	T(704,52)	T(696,52)
s14.png/-	M5 A0.72	M6 A0.67	M5 A0.67	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.66
	T(372,388)	T(236,52)	T(452,320)	T(704,52)	T(696,52)	T(684,52)	T(252,52)	T(672,52)
s15.png/1M	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.66	M6 A0.66	M6 A0.66
	T(236,52)	T(696,52)	T(684,52)	T(252,52)	T(224,52)	T(704,52)	T(672,52)	T(668,52)
s16.png/1V	M5 A1.14	M5 A1.13	M5 A1.12	M5 A1.09	**M1 A0.87**	M5 A0.83	M5 A0.81	M5 A0.79
	T(352,212)	T(376,232)	T(360,212)	T(340,212)	**T(432,396)**	T(340,356)	T(404,216)	T(396,216)
s17.png/1*	M5 A1.03	M5 A0.97	M5 A0.96	M5 A0.96	M5 A0.92	M5 A0.92	M5 A0.88	M6 A0.76
	T(364,212)	T(388,216)	T(396,216)	T(348,224)	T(340,212)	T(352,212)	T(368,224)	T(368,212)
s18.png/1M	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.66	M6 A0.66	M6 A0.66
	T(236,52)	T(696,52)	T(684,52)	T(252,52)	T(224,52)	T(672,52)	T(668,52)	T(664,52)
s19.png/1#	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.66	M6 A0.66	M6 A0.66
	T(236,52)	T(696,52)	T(684,52)	T(252,52)	T(224,52)	T(672,52)	T(668,52)	T(664,52)
s20.png/2M	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.66	M6 A0.66
	T(236,52)	T(704,52)	T(696,52)	T(684,52)	T(252,52)	T(224,52)	T(672,52)	T(668,52)
s21.png/2V	**M2 A1.35**	M5 A1.19	M5 A1.19	M5 A1.11	M5 A1.08	M5 A0.9	M5 A0.89	M5 A0.82
	T(440,336)	T(256,220)	T(264,220)	T(228,216)	T(260,180)	T(272,184)	T(280,184)	T(288,220)
s22.png/#	M6 A0.68	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.66	M6 A0.66
	T(376,384)	T(236,52)	T(696,52)	T(684,52)	T(252,52)	T(224,52)	T(672,52)	T(668,52)
s23.png/5V#	M5 A1.07	M5 A0.98	M5 A0.98	M5 A0.96	M3 A0.95	M5 A0.93	M5 A0.92	M3 A0.9
	T(232,248)	T(460,328)	T(452,328)	T(240,232)	T(492,244)	T(232,232)	T(212,208)	T(508,244)
s24.png/5V#	M5 A1.17	M5 A1.01	M5 A0.98	M5 A0.97	M5 A0.9	M5 A0.9	M6 A0.87	**M5 A0.86**
	T(236,492)	T(228,472)	T(472,68)	T(260,492)	T(268,492)	T(444,64)	T(476,528)	**T(100,232)**
s25.png/5V*#	M5 A1.13	M5 A1.13	M5 A0.98	M5 A0.9	M5 A0.89	M5 A0.88	M3 A0.88	M3 A0.84
	T(240,232)	T(228,232)	T(300,216)	T(288,212)	T(276,212)	T(356,208)	T(492,244)	T(484,244)
s26.png/5V	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.67	M6 A0.66	M6 A0.66	M6 A0.66
	T(236,52)	T(696,52)	T(684,52)	T(252,52)	T(224,52)	T(672,52)	T(668,52)	T(664,52)
s27.png/5V	**M5 A2.44**	M5 A1.92	M5 A1.89	M5 A1.41	M5 A1.39	M5 A1.18	M5 A1.06	M5 A1.05
	T(324,280)	T(336,284)	T(316,276)	T(308,272)	T(344,288)	T(232,440)	T(232,424)	T(236,420)
s28.png/5V	**M5 A2.44**	M5 A1.85	M5 A1.83	M5 A1.47	M5 A1.35	M5 A1.35	M5 A1.31	M5 A1.27
	T(324,280)	T(336,284)	T(316,276)	T(224,444)	T(344,288)	T(308,272)	T(212,448)	T(236,440)

U ovom eksperimentu skaliranje nije uzeto u obzir zato što zbog prisutnosti tri modela s mnogo ravnih segmenata program uvijek lažno prijavljuje prisutnost objekata 3, 5 ili 6 čak

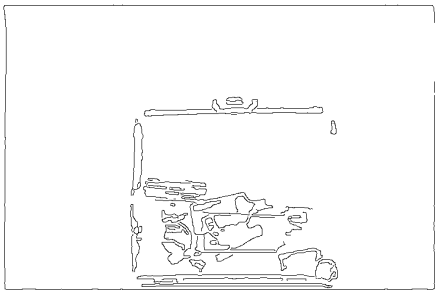
i kada je neki drugi objekat prisutan u sceni (bilo umanjen ili u prirodnoj veličini, vidi sliku 3.12). To se događa zato što postoji veliki strukturirani šum u slici (koji se sastoji od ravnih segmenata) i ti ravni segmenti se bolje poklapaju s umanjenim i uvećanim verzijama modela 3, 5 i 6 nego sa stvarnim modelom (kod kojega se, ako je mali, izgubi i dio rubova). Kako ti ravni segmenti također pripadaju metalnim predmetima,



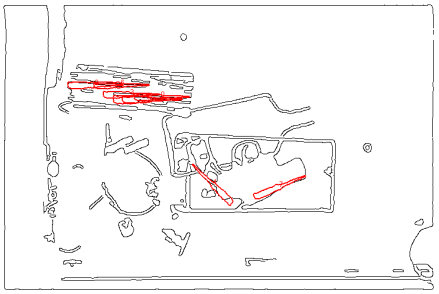
Ulazna slika



Segmentirana slika



Nakon detekcije rubova



Nakon raspoznavanja: predloženi modeli su u crvenoj boji i vidi se da se niti jedan ne poklapa sa modelom koji je stvarno u slici.

Slika 3.12 Slabo raspoznavanje sa skaliranjem modela

Vrijeme obrade je variralo (proporcionalno broju i duljini detektiranih rubova) između 50 i 170 sekundi na računalu s 233MHz procesorom.

4 Zaključak

U uvodu su opisani problem i cilj rada, način dobivanja eksperimentalnih snimaka te je rečeno nešto o tehnici snimanja prtljage. Budući da se obrađuje slika u boji, izneseni su neki osnovni pojmovi o boji, opisana su dva bitna modela (RGB koji je prilagođen stroju te HSI koji je blizak ljudskom poimanju boje) te je opisana konverzija iz RGB u HSI model. Također su dane reference na literaturu koja se podrobnije bavi aspektima obrade slike u boji.

Zatim je opisana segmentacija slike, nužan korak prije računanja značajki za raspoznavanje. Pokazana je nepouzdanost H komponente nakon pretvorbe iz RGB u HSI model zbog čega se odustalo od amplitudne segmentacije na područja temeljene na HSI modelu. Zbog toga je odlučeno da se segmentacija na područja radi Kohonenovom neuronskom mrežom (odjeljak 2.1). Opisana je mreža, modificirani algoritam učenja, prikazane su neke iteracije u postupku učenja te su dani primjeri segmentacije.

Sliku segmentiranu na područja obrađuje se Cannyjevim detektorom rubova koji je detaljno opisan u odjeljku 2.2.2. Prije samog opisa dane su potrebne matematičke osnove (konvolucija) te su pojedini koraci algoritma pokazani na primjeru jednodimenzionalnog ruba, radi lakšeg uočavanja pojedinih detalja.

Segmentaciju slijedi raspoznavanje. Opisano je nekoliko postupaka od kojih se odustalo te su navedeni i razlozi. Metoda izbora je generalizirana Houghova transformacija koja je invarijantna na rotaciju, translaciju i skaliranje, a u slučaju preklapajućih objekata degradacija raspoznavanja je postupna (ovisi o vidljivosti objekta). Prikazani su nedostaci GHT sa zadanim skupom modela te načini ispravljanja nekih (odjeljak 3.1.4).

Posebno su nacrtani modeli, a kao primjer su za jedan model navedene i tablice na temelju kojih radi GHT algoritam. Također je opisano računanje gradijenta krivulje koja nije zadana analitički pa se ne može upotrijebiti derivacija.

U dodacima je opisan način instalacije programa, te su dokumentirane klase za obradu slike, Kohonenovu mrežu i vektorske objekte koje se mogu upotrijebiti i u drugim projektima.

Eksperimentalni rezultati daju tablicu sa rezultatima raspoznavanja. Također je dan sažetak svih parametara za pojedine korake algoritma pomoću kojih se došlo do tih rezultata. Opisano je i zašto skaliranje nije uzeto u obzir. Čak i bez skaliranja, na nekim slikama zbog okolnog šuma objekt nije prepoznat niti u prirodnoj veličini.

Može se zaključiti da je traženje i raspoznavanje objekta uspješno jedino ukoliko je objekt dovoljno velik te nije previše zaklonjen drugim objektima. Također, postupak raspoznavanja jako ometa strukturirani šum koji je sličan nekom modelu (ravne linije). Inače, GHT algoritam je pouzdan i uspješno nalazi objekt ukoliko se *unaprijed* zada što treba tražiti (slika 3.5).

Treba istaknuti da je kvaliteta eksperimentalnih rezultata ograničena upravo načinom dobivanja eksperimentalnih snimaka prtljage. Rezultati bi bili puno bolji da je bio moguć pristup digitalnim podacima s 3D snimke prtljage (tada bi bio potreban i drugi postupak segmentacije). Dakle, relativno slabi rezultati raspoznavanja mogu se objasniti sljedećim činjenicama:

- Loša kvaliteta ulaza (uzrokovana konverzijom 3D digitalna slika→2D projekcija kodirana bojama→VHS snimka →digitalizacija VHS snimke).
- Nepotpuna segmentacija zbog loše kvalitete ulaza.
- Strukturirani šum (ravne linije: rubovi slike, kišobran) koji se bolje poklapa sa modelima koji imaju puno ravnih segmenata nego stvarni model (slika 3.12).

Razvijeni sustav još ne može služiti niti kao pomoćno sredstvo za skretanje pažnje na potencijalno opasne objekte zato što ne može detektirati opasne umanjene predmete. Da bi se podigla razina pouzdanosti sustava istraživanje se treba nastaviti u 4 smjera:

- Sučelje programa i sklopovlja koje snima prtljagu kako bi se izbjegle nepotrebne konverzije podataka čime se jako gubi na kvaliteti.
- Druge metode segmentacije koje bi u 3D snimci prtljage mogle otkriti objekte načinjene od zanimljivih materijala kako bi se raspoznavanje moglo fokusirati na ta područja.
- Računanje značajki trodimenzionalnih objekata.
- Pogodan sustav za predstavljanje znanja: trenutna metoda s ‘‘rječnikom’’ modela je nefleksibilna i raspoznaje samo modele koji se nađu u ‘‘rječniku’’.

Student: Željko Vrba

5 Literatura

1. J.Illingworth, J.Kittler: A Survey of the Hough Transform. Computer Vision, Graphics, and Image Processing, 44, 1988, 87-116.

2. R.D.Duda, P.E.Hart: Use of the Hough transform to detect lines and curves in pictures. Communications of the Association of Computing Machinery, 15, 1972, 11-15.

3. D.H.Ballard: Generalizing the Hough transform to detect arbitrary shapes. Pattern Recognition, 3, 1981, 245-256.

4. Simon Haykin: Neural Networks: A Comprehensive Foundation. 2nd ed., Prentice Hall, 1999.

5. Rafael C. Gonzalez, Paul Wintz: Digital Image Processing. 2nd ed., Addison-Wesley, 1987.

6. P.V.C. Hough: A method and means for recognizing complex patterns. U.S. Patent 3,069,654, 1962.

7. N.Ayache, O.D.Faugeras: HYPER: A New Approach for the Recognition and Positioning of Two-Dimensional Objects. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1, 1988, 44-54.

8. T.Kohonen: Self-Organization and Associative Memory. Springer-Verlag, 1984.

9. T.Kohonen, K.Masisara and T.Saramaki: Phonotopic Maps -- Insightful Representation of Phonological Features for Speech Representation. Proceedings IEEE 7th Inter. Conf. on Pattern Recognition, Montreal, Canada, 1984.

10. J.Canny: A Computational Approach to Edge Detection. Pattern Analysis and Machine Intelligence, 6, 1986, 679-698.

11. Specifikacija PNG formata. Može se naći na internetu na sljedećim adresama: ftp: //ftp.uu.net/graphics/png/documents ili http://www.w3.org/TR/REC.png.html.

12. Specifikacija i programi za rad s PNM formatom slike. Može se naći na interentu na adresi http://www.acme.com/software/pbmplus.

13. Richard P. Lippman: An Introduction to Computing with Neural Nets. IEEE ASSP Magaizne, 3, 1987, 4-22.

14. Slobodan Ribarić, N. Pavešić, G. Gyergyek: Uvod u raspoznavanje uzoraka. Tehnička knjiga, Zagreb, 1998.

Dodatak A Sadržaj CD-a i instalacija programa

Program je pisan u C++ za Linux operacijski sustav. Grafičko sučelje i rutine za obradu slike koriste Qt GUI biblioteku, a sam program podržava učitavanje i spremanje slika u PNG i PNM formatima. Tablica u nastavku prikazuje sadržaj CD-a.

`common`	izvorni kod zajednički za sve programe
`ghough`	izvorni kod programa za interaktivan prikaz HT
`images`	ulazne slike
`models`	modeli
`recogn`	izvorni kod programa za raspoznavanje
`program`	izvršni kod programa i podaci potrebni za rad
`text`	tekst diplomskog rada
`util`	izvorni kod pomoćnih programa

Za rad programa nije potrebna posebna instalacija. Treba ući u direktorij `program` i pokrenuti program `recgon` ili `ghough`. Programi se *moraju* pokrenuti iz tog direktorija jer se u tom direktoriju nalaze i datoteke koje programi trebaju za ispravan rad.

Direktorij `util` sadrži izvorni kod pomoćnih programa koji su korišteni za učenje neuronske mreže (programi `train`, `quant` i `map`) te za vektorizaciju modela (pretvaranje crno bijele slike ruba modela u listu točaka; program `trace`).

U toku pisanja programa razvijene su i klase za prikaz geometrijskih objekata (točke i konture) te sustav za njihovo učitavanje i spremanje u datoteke.

Dodatak B Klase za obradu slike

U nastavku će biti opisane sve bitne klase za obradu slike, organizirane prema header datotekama u kojima se nalaze. Dokumentirane su samo `public` i `protected` članovi. Slika B.2 prikazuje hijerarhiju klasa za obradu slike.

B.1 refcnt.h

Ova datoteka implementira klasu koja omogućava jednostavno brojanje referenci klasama koje ju naslijede (ukoliko se pridržavaju nekih konvencija). Te mogućnosti koristi klasa `Raster`.

```
class RefCount {
protected:
  bool DESTROY();
  int ASSIGN(const RefCount&);
  bool DETACH();
public:
  RefCount();
  RefCount(const RefCount&);
  virtual ~RefCount();
};
```

`DESTROY()` poziva naslijeđena klasa u destruktoru. Ukoliko metoda vrati `true`, tada je to zadnja instanca objekta te destruktor treba osloboditi dijeljenu memoriju.

`ASSIGN()` se brine o brojanju referenci kada se radi *plitka* kopija. Metodu treba pozvati (i provjeriti povratnu vrijednost) naslijeđena klasa svaki put prije nego što napravi plitku kopiju. Vrijednosti koje metoda vraća su sljedeće:

0 Pokušano je dodjeljivanje objekta samom sebi
1 Normalan slučaj; može se jednostavno obaviti plitko kopiranje.
2 Dodjeljivanjem se gubi zadnja referenca na dijeljenu memoriju koju drži ciljni objekt dodjeljivanja. Prije nego što obavi dodjeljivanje, ciljni objekt treba izbrisati dijeljenu memoriju.

`DETACH()` naslijeđena klasa poziva prije nego što napravi duboku kopiju (deep copy). Ukoliko metoda vrati `false`, objekt još ne dijeli memoriju ni sa jednim drugim objektom te zapravo (možda skupu) operaciju dubokog kopiranja ne treba raditi.

B.2 imgbase.h

Ova header datoteka definira konstante te dvije osnovne klase za obradu slike: `Raster` (reprezentacija slike u boji) te `Transform` (razne transformacije slike).

B.2.1 Konstante

Definirane su sljedeće konstante

```
typedef unsigned char Pixel;

#define PIXEL_MAX 255

enum Colorspace { MONO = 0, RGB, HSI };

enum Channel
{
       RED = 0, GREEN, BLUE,
       HUE = 0, SATURATION, INTENSITY,
       GRAY = 0
};
```

`Pixel` je tip koji se koristi za reprezentaciju piksela. Može se promijeniti u npr. `float` ako su potrebne neke posebne operacije. Pripadajući makro je `PIXEL_MAX` koji definira najveću svjetlinu koja se može pojaviti u slici; za ovaj slučaj je to 255 (najviše koliko stane u jedan bajt).

`Colorspace` je konstanta koja govori u kojem formatu je spremljena slika u memoriji (siva slika, RGB ili HSI model); koristi se i u konstruktoru `Raster` klase.

`Channel` definira simboličke konstante pomoću kojih se može nezavisno pristupati pojedinim kanalima slike.

B.2.2 Raster

Ovo je osnovna klasa u kojoj je spremljena slika. Klasa implementira brojanje referenci radi štednje memorije i brzine prilikom korištenja operatora = i konstruktora kopiranja. Zbog takve implementacije, konstruktor kopiranja i operator= kopiraju samo nekih 40-ak bajtova neovisno o veličini slike, umjesto potencijalnih megabajta (ovisno o veličini slike).

```
class Raster : public RefCount {
public:
  Colorspace c;
  int w, h;
```

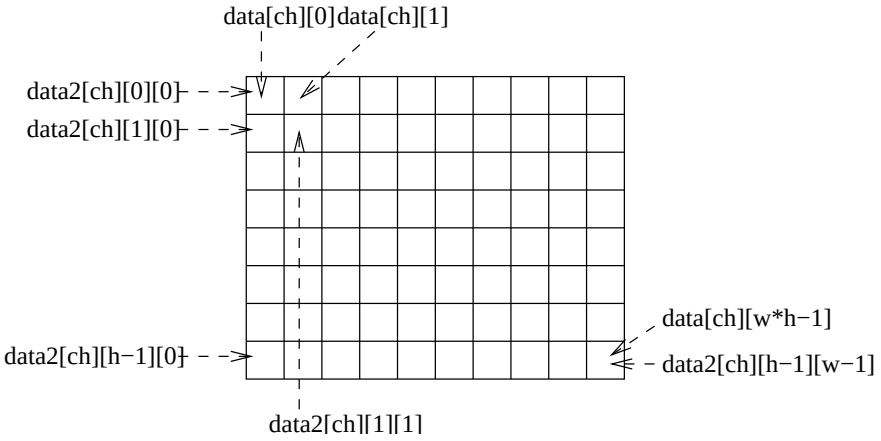
```
  Pixel *data[3];
  Pixel **data2[3];

  Raster();
  Raster(Colorspace _c, int _w, int _h);
  Raster(const Raster&);
  Raster(const QImage &im);
  Raster(const Raster &im, Colorspace ch);
  ~Raster();
  QImage image() const;
  void replace(const Transform &it);
  Raster copy(const Transform &it) const;
  void detach();
  Raster &operator=(const Raster&);
};
```

B.2.2.1 Varijable

`w,h` Širina (w) i visina (h) slike u pikselima.

`c` Sadrži model boje. Za MONO model boje koristi se (i alocira) samo nulta ravnina dok RGB i HSI modeli koriste sve 3 ravnine. Ravninu bira prvi indeks u data i data2 nizovima.

`data,data2` Nizovi koji sadrže stvarne podatke o slici; za organizaciju podataka i odnose među tim nizovima pogledati sliku B.1. data2 je niz od h pointera od kojih svaki pokazuje na početak svog retka.



Slika B.1 Organizacija jedne ravnine piksela

B.2.2.2 Konstruktori

`Raster()` Ovo je default konstruktor koji stvori sliku dimenzija 0×0 i *ne alocira* memoriju. Potreban je radi kasnijih metoda koje prvo deklariraju varijablu, a zatim naknadno alociraju sliku (privatnim metodama).

`Raster(Colorspace _c, int _w, int _h)` Stvara sliku prema zadanom modelu boja i dimenzijama (w: širina, h: visina u pikselima) te alocira memoriju.

`Raster(const Raster&)` Konstruktor kopiranja. Radi plitku kopiju.

`Raster(const QImage &im)` Stvara sliku iz postojećeg `QImage` objekta (dio Qt biblioteke). `QImage` objekt može učitavati PNG i PNM slike iz datoteka, te je zapravo ovim konstruktorom omogućeno direktno kreiranje `Raster` klasa iz postojećih slika, npr:

```
Raster r(QImage("slika.png"))
```

`Raster(const Raster &im, Colorspace ch)` im *mora* biti slika u boji (RGB ili HSI). Novostvorena slika je *siva* i dijeli memoriju s `im` objektom. Sadržaj nove slike je izabrani kanal originalne slike.

B.2.2.3 Transformacije

`void replace(const Transform&)` Radi zadanu transformaciju na slici i zamijeni trenutnu sliku sa rezultatom.

`Raster copy(const Transform&) const` Radi zadanu transformaciju na slici i vrati rezultat. Početna slika ostaje nepromijenjena.

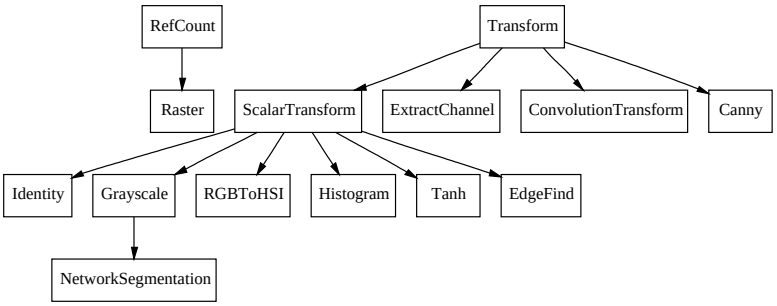
B.2.2.4 Ostale metode

`QImage image() const` Pretvara sliku u QImage. To je najčešće potrebno radi prikaza na ekranu ili spremanja u datoteku.

`void detach()` Ukoliko slika dijeli memoriju s nekom drugom slikom, kopira taj dio memorije na posebno mjesto. Ta metoda se poziva ukoliko treba osigurati da promjena trenutne slike neće izazvati promjenu druge slike s kojom ova (možda) dijeli dio memorije.

`Raster &operator=(const Raster&)` Operator dodjeljivanja. Radi plitko kopiranje (shallow copy).

B.2.3 Transform



Slika B.2 Hijerarhija klasa za obradu slike

Transform je klasa koju trebaju naslijediti sve klase koje rade neku transformaciju slike. Kompletna hijerarhija implementiranih klasa prikazana je na slici B.2 i one će biti opisane u poglavlju B.3.

```
class Transform {
public:
  virtual ~Transform();
  virtual void replace(Raster &r) const;
  virtual void copy(Raster &dst, const Raster &src) const;
};
```

`void replace(Raster &r)` Ova metoda se zove kada je sliku treba *zamijeniti* sa njenom transformacijom; original se gubi.

`void copy(Raster &dst, const Raster &src) const` Ova metoda se zove kada treba stvoriti *novu* transformiranu sliku (`dst`), a original treba ostati neizmijenjen (`src`). Memoriju za odredišnu sliku (dst) mora alocirati metoda koja poziva copy. To je najčešće `Raster::copy`.

Implementacijski detalj: klasa *nije* apstraktna zato što naslijeđena klasa ne mora implementirati obje metode (`copy` i `replace`). Tada bi se, u slučaju apstraktne osnovne klase, kompajler bunio o pokušaju instanciranja apstraktne klase. Zato ovdje obje metode naprave `abort()`.

B.3 imgtransform.h

Ova datoteka implementira sve klase za transformaciju slike.

B.3.1 ScalarTransform

Mnogo transformacija može se opisati kao jednostavna funkcija koja uzima jedan piksel i vraća jedan piksel (sivi ili od 3 komponente). Sve takve transformacije nazvao sam *skalarnima*, analogno skalarnim funkcijama u matematici. Ovu klasu također treba smatrati

apstraktnom, iako to nije u smislu C++ jezika (slično kao kod Transform klase, sve metode pozovu `abort()`).

```
class ScalarTransform : public Transform {
protected:
  virtual void fc(Pixel *r, Pixel *g, Pixel *b,
                  Pixel r0, Pixel g0, Pixel b0) const;
  virtual void fr(Pixel *r, Pixel *g, Pixel *b) const;
  virtual void fc(Pixel *p, Pixel p0) const;
  virtual void fr(Pixel *p) const;
public:
  virtual ~ScalarTransform();
  virtual void replace(Raster&) const;
  virtual void copy(Raster&, const Raster&) const;
};
```

Klasa definira metode `copy` i `replace` Metode su implementirane da odgovaraju definiciji ‘‘skalarne’’ transformacije, pri tome pozivajući jednu od 4 `protected` funkcije (ovisno o vrsti slike) koje moraju implementirati naslijeđene klase. Te funkcije implementiraju funkciju koju treba napraviti na slici. U nastavku slijedi (malo pročišćena) implementacija metoda `copy` i `replace`:

```
void ScalarTransform::replace(Raster &id) const
{
  if(id.c != MONO) for(int i = 0; i < id.w * id.h; i++) {
    fr(id.data[0]+i, id.data[1]+i, id.data[2]+i);
  } else for(int i = 0; i < id.w * id.h; i++) {
    fr(id.data[MONO]+i);
  }
}

void ScalarTransform::copy(Raster &dst,
                           const Raster &src) const
{
  if(src.c != MONO) for(int i = 0; i < src.w*src.h; i++) {
    fc(dst.data[0]+i, dst.data[1]+i, dst.data[2]+i,
    src.data[0][i], src.data[1][i], src.data[2][i]);
  } else for(int i = 0; i < src.w*src.h; i++) {
    fc(dst.data[MONO]+i, src.data[MONO][i]);
  }
}
```

Kao što se vidi iz implementacije, fr i fc metode pozivaju se ovisno o tome da li funkcija treba zamijeniti (r, replace) vrijednost postojećeg piksela ili ga kopirati (c, copy) u novi.

Postoje dvije varijante za svaku od tih funkcija: jedna varijanta prima komponente piksela ukoliko je u pitanju slika u boji, a druga varijanta prima samo svjetlinu piksela ukoliko je u pitanju siva slika. Također, vidi se da se za sliku u boji zadana operacija obavlja *na svakom kanalu posebno*.

Donja tablica daje sažetak postojećih skalarnih transformacija zajedno sa pripadajućim konstruktorom.

Konstruktor	Objašnjenje
`Identity()`	Identiteta: izlazna slika identična je ulaznoj.
`Grayscale()`	Pretvara sliku u boji u sivu sliku prema formuli 1.1. akođer promijeni i c član pripadajuće slike, te dealocira ostale (za sivu sliku nepotrebne) kanale.
`RGBToHSI()`	Pretvara sliku u RGB modelu u HSI model prema formulama 1.3.
`Tanh(float a)`	Računa funkciju tanh(ax). Prije transformacije svi pikseli se skaliraju u interval $[0, 1]$, a nakon transformacije se vraćaju u interval $[0, PIXEL\_MAX]$.
`Histogram()`	Vidi tekst.
`NetworkSegmentation(Mode)`	Vidi tekst.

B.3.1.1 Histogram

`Histogram` klasa računa histogram zadane sive slike. Specifična je po tome što dodaje dvije nove metode: `void reset()` koja postavlja histogram na 0 te `const int *getStatistics() const` koja vraća niz od `PIXEL_MAX` vrijednosti.

B.3.1.2 NetworkSegmentation

`NetworkSegmentation` klasa segmentira sliku Kohonenovom neuronskom mrežom. Izvedena je iz klase `Grayscale` jer također, ovisno o argumentu konstruktora, može stvoriti sivu sliku.

```
class NetworkSegmentation : public Grayscale {
public:
  enum Mode { QUANTIZE, CLASSIFY };
public:
  NetworkSegmentation(Mode m = CLASSIFY);
};
```

Ukoliko konstruktor dobije `CLASSIFY` argument, tada se slika klasificira u nekoliko unaprijed izabranih klasa; tada je rezultat siva slika gdje nijansa sive označava pripadnost određenoj klasi. Ukoliko je argument konstruktora `QUANTIZE`, tada se slika vektorski kvantizira.

Ime datoteke iz koje se učitava mreža, kao i opasne klase boja direktno su kodirani u implementaciju i ne mogu se dinamički mijenjati.

B.3.2 ExtractChannel

Ova transformacija iz slike u boji stvara sivu sliku tako što iz početne slike izolira izabrani kanal (koji se navodi u konstruktoru). Ova klasa implementira samo `copy()` metodu, a nova slika dijeli memoriju s početnom.

```
class ExtractChannel : public Transform {
public:
  ExtractChannel(Channel _ch);
  virtual void copy(Raster&, const Raster&) const;
};
```

B.3.3 ConvolutionTransform

Odjeljak 2.2.1 detaljno opisuje matematičku pozadinu konvolucije. Ova klasa je generička implementacija, slično kao što je `ScalarTransform` generička implementacija skalarne transformacije.

```
class ConvolutionTransform : public Transform {
protected:
  int nr, nc;
  mutable const Raster *src;

  virtual Pixel f(int r, int c) const = 0;
public:
  ConvolutionTransform(int r, int c) : nr(r), nc(c) { }
  virtual void copy(Raster&, const Raster&) const;
};
```

B.3.3.1 Varijable

`nr,nc` Veličina konvolucijske maske; broj redaka i stupaca.

`src` Funkcija f treba piksele od originalne slike, ali pointer na originalnu sliku nemamo u konstruktoru već samo u copy metodi iz koje se ova varijabla inicijalizira. Mora biti `mutable` kako bi se u temporary objektu iz `const` metode mogla dodijeliti vrijednost.

B.3.3.2 Metode

`virtual Pixel f(int, int) const` Poziva se za računanje svakog transformiranog piksela. Garantira se da će se uvijek pozvati s koordinatama (i, j) tako da je $(i, j) -$

$(nr/2, nc/2) \geq 0$ i (i, j)+$(nr/2, nc/2) < (H, W)$ gdje su W i H širina i visina ulazne slike (to garantira copy metoda). Funkcija vraća vrijednost transformiranog piksela. Parametri funkcije su r i c (redak i stupac transformiranog piksela).

`ConvolutionTransform(int, int)` Konstruktor. Parametri su broj redaka (r) i stupaca (c) konvolucijske maske.

`virtual void copy(Raster&, const Raster&) const` Implementira mehanizam konvolucije. Za svaki transformirani piksel (r_t, c_t) računa funkciju $f(r_t, c_t)$.

B.3.4 Canny

```
class Canny : public Transform {
protected:
  virtual void copy(Raster &dst, const Raster &src) const;
public:
  Canny(float sigma, float tlow, float thigh);
};
```

Ova klasa implementira Cannyjev postupak detekcije rubova i radi samo na sivim slikama. Parametri konstruktora određuju parametre Cannyjevog postupka i definirani su u odjeljku 2.2.2.

B.4 kohnwork.h

Ova datoteka definira klase Neuron i KohonenNetwork koje zajedno implementiraju osnovne operacije potrebne za treniranje i raspoznavanje pomoću Kohonenove neuronske mreže.

B.4.1 Neuron

Ova klasa implementira osnovne funkcije potrebne za implementaciju algoritma učenja, kao npr. inicijalizacija slučajnim brojevima, Euklidsku udaljenost te korak učenja. Također su implementirane operacije sa streamovima, za ulaz i izlaz. Format je binarni, za detalje pogledati izvorni kod u `kohnwork.cc`.

```
struct Neuron {
  float *vec;
  int dim;

  Neuron();
  Neuron(int d);
  Neuron(const Neuron &n);
```

```
  ~Neuron();
  float d(float *inp) const;
  float update(float eta, float h, float *inp);

  friend istream &operator>>(istream &is, Neuron &n);
  friend ostream &operator<<(ostream &os, const Neuron &n);
};
```

B.4.1.1 Varijable

`vec` Vektor težina neurona.

`dim` Dimenzija vektora vec.

B.4.1.2 Konstruktori

Default konstruktor postavi vec i dim na 0, dakle ovo je prazan neuron. Potreban je zbog STL klasa koje zahtijevaju default konstruktor. Copy konstruktor radi duboku kopiju.

Drugi konstruktor alocira memoriju za neuron s d dimenzija i postavi ga na slučajne vrijednosti u intervalu $[-0.5, 0.5)$.

B.4.1.3 Metode

`float d(float*) const` inp je vektor od dim elemenata. Metoda vraća Euklidsku udaljenost između trenutnih težina neurona vec i ulaza inp.

`float update(float, float, float*)` Metoda ažurira težine neurona prema koraku 5 učenja u odjeljku 2.1. Parametre eta=$\eta(t)$ i h=$h(t)$ računa pozivajući program.

B.4.2 KohonenNetwork

Ova klasa implementira algoritme učenja i klasificiranja kako su opisani u odjeljku 2.1.

```
class KohonenNetwork {
public:
  KohonenNetwork(int r, int c, int dim, float eta0,
                 float sigma0, float tau1, float tau2);
  KohonenNetwork(const char *fname);
  ~KohonenNetwork();
  float train(float *inp);
  int run(float *inp) const;
  int width() const;
```

```
  int height() const;
  const Neuron &neuron(int i) const;
  bool read(const char *fname);
  bool write(const char *fname) const;
};
```

B.4.2.1 Konstruktori

Prvi konstruktor stvara mrežu geometrije r redaka i c stupaca sa slučajno inicijaliziranim neuronima dimenzije dim. Ostali parametri su početni parametri učenja i vremenske konstante opisani u odjeljku 2.1.

B.4.2.2 Metode

`float train(float*)` Za dani ulaz inp (dimenzije dim) provodi algoritam učenja opisan u odjeljku 2.1. Vraća sumu apsolutnih vrijednosti svih ispravaka nad pojedinačnim komponentama svih neurona. Ta vrijednost se koristi kao ‘‘mjera promjene’’ mreže i služi za zaustavljanje procesa učenja.

`int run(float*) const` Vraća indeks ‘‘najboljeg’’ neurona (najmanja udaljenost) za ulaz inp.

`int width() const` Vraća broj stupaca (širinu) mreže.

`int height() const` Vraća broj redaka (visinu mreže). Ukupan broj neurona u mreži je `width()*height()`.

`const Neuron &neuron(int) const` Vraća vrijednost i-tog neurona u mreži.

`bool read(const char*)` Učitava mrežu iz datoteke imena fname. Ukoliko uspije, vrati `true`. Inače vrati `false` i uzrok pogreške ispiše na `stderr`.

`bool write(const char*) const` Zapisuje mrežu u datoteku imena fname na disk. Ukoliko uspije, vrati `true`. Inače vrati `false` i uzrok pogreške ispiše na `stderr`.

Dodatak C Klase za vektorske objekte

Sve pomoćne klase potrebne za vektorizaciju i stvaranje modela nalaze se u datoteci `vectorize.h`.

C.1 Point

Ovo je vrlo jednostavna klasa koja predstavlja točku u dvodimenzionalnom pravokutnom koordinatnom sustavu:

```
struct Point {
  float x, y;
  Point(int _x = 0, int _y = 0) : x(_x), y(_y) { }

  friend ostream &operator<<(ostream &os, const Point &p);
  friend istream &operator>>(istream &is, Point &p);
};
```

Definirane su i dvije funkcije za zapis, odnosno čitanje Point klase sa nekog streama. Izlazni prikaz klase je par brojeva odvojenih razmakom, npr. `-11.3 5.6`.

C.2 Path

```
class Path : public vector<Point> {
  friend istream &operator>>(istream&, Path&);
  friend ostream &operator<<(ostream&, const Path&);
};
```

Ova klasa predstavlja listu dvodimenzionalnih točaka. U datoteku se sprema kao niz točaka sa po jednom točkom (par koordinata odvojenih razmakom) u svakom redu. Lista točaka završava jednim praznim redom.

C.3 PathList

```
class PathList : public list<Path> {
  friend istream &operator>>(istream&, PathList&);
  friend ostream &operator<<(ostream&, const PathList&);
};
```