SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1510

PRAĆENJE OBJEKATA U SLIKOVNIM SEKVENCAMA NA TEMELJU INFORMACIJE O RUBOVIMA

Darko Jurić

Zagreb, srpanj 2010.

Zahvaljujem mentoru prof. dr. sc. Zoranu Kalafatiću na strpljenju, velikoj pomoći, pruženoj literaturi i izvrsnim savjetima prilikom izrade rada kao i mojoj obitelji na podršci.

Sadržaj

1. Uvod

Područje praćenja objekata je jedno od najotvorenijih u računalnom vidu. Nažalost, ne postoji univerzalna tehnika praćenja, stoga se konstanto razvijaju novi postupci. Područje iako je još u povojima toliko je razvijeno da danas poznajemo nekoliko kategorija u koje možemo razvrstati algoritme za praćenje. Obično se dijele na:

- **praćenje kontura** – (*eng. contour tracking*) podrazumijeva određivanje granice objekta (npr. aktivne konture)
- **praćenje nakupine** – (*eng blob tracking*) podrazumijeva segmentaciju samog objekta (npr. ekstrakcija nakupine)
- **praćenje temeljeno na nekoj jezgri** – (*eng. kernel based*) podrazumijeva postojanje iterativnog postupka za lokalizaciju objekta (npr. KLT tracker, Mean-shift tracker, itd.)
- **različite kombinacije vrsta prve tri točke**

U ovom radu implementiran je algoritam koji se temelji na praćenju rubova objekta. Najprije nizom metoda izluče se rubovi koji se zatim nastoje usporediti i pratiti pomoću metode transformacije udaljenosti. Poteškoća pri praćenju je samo mijenjanje objekta, njegovo skaliranje i nesavršeno izlučivanje rubova koje može dosta odudarati od pojedine susjedne slike u videu. Rješenje koje je implementirano,a koje se odnosi na rješenje ovog problema je redovito ažuriranje kontura objekta. Dodatna poteškoću je pozadina koja u većini slučajeva nije statična Za usporedbu i obnavljanje dvaju područja praćenja u susjednim slikama razvijen je algoritam koji se temelji na pronalasku najbližeg susjednog piksela i širenja ruba na kojem leži takav piksel.

Takvo rješenje problema se pokazalo dosta uspješnim, ali postoje velike poteškoće koje se ponajprije javljaju u promijeni veličine objekta. Naime, zbog niske uspješnosti metoda koje obavljaju prilagodbu prostora praćenja veličini objekta, veličina prostora praćenja nije mijenjana.

2. Strukture i algoritmi korišteni u postupku praćenja

Za samo izlučivanje rubova potrebno je bilo upotrijebiti nekoliko metoda odnosno filtara. Dio slike od kojeg je bilo potrebno načiniti sliku rubova se najprije pretvorio u sliku sivih tonova (*eng. grayscale*). Tada se moglo pristupiti izlučivanju rubova. Rubovi su se izlučivali Sobelovim detektorom rubova. Iako se Cannyev detektor rubova više koristiti za ovakve svrhe Sobelova metoda se pokazala kao bržom i boljom kao što je navedeno u poglavlju 2.2.2. Slika dobivena takvim postupkom može sadržavati 255 vrijednosti sivih nijansi, a računanje transformacije udaljenosti obavlja se nad binarnim slikama. Binarizacija je vršena širenjem greške (*eng. dithering*) Stuckijevom matricom. Razlozi su navedeni u poglavlju 2.3.4. Naposljetku računanje transformacije udaljenosti (*eng. distance transform*) obavlja se pomoću algoritma za računanje Chamferovih (poglavlje 2.4.2) udaljenosti.

2.1. Pretvaranje boja slike u sivu paletu

Konverzija u sive tonove će biti objašnjena na RGB prostoru boja. Pretpostavlja se da su ulazne slike u boji čiji pikseli sadrže informaciju o boji zapisanu u obliku te tri komponente. Veličine u bitovima tih komponenta mogu varirati ili one mogu biti predstavljene tablicom što je tipično za 4-bitne i 8-bitne slike u boji.

Nasuprot slikama u boji slike nijanse jedne boje nose samo informacije o intenzitetu nijanse takve boje. Za njih se još kaže da su monokromatske. Takve slike se razlikuju od crno-bijelih slika u kojima postoje samo dvije nijanse sive boje: crna i bijela.

Često se dobivaju kao rezultat mjerenja intenziteta svjetla jednoga kanala elektromagnetskog spektra (npr. spektar vidljive svjetlosti, infracrveni spektar…). Također mogu biti sintetizirane od slike koja sadrže boje.

2.1.1. Numerički prikaz

Intenzitet piksela je izražen u rasponu od neke minimalne do maksimalne vrijednosti. Često se vrijednost iskazuje realnim brojevima između 0 i 1. Takva notacija je česta u stručnoj literaturi.

Iako se vrijednosti često iskazuju realnim brojevima pikseli slika pohranjene u binarnom zapisu nose cjelobrojne vrijednosti. Često je intenzitet pohranjen u 8-bitova (1 bajt), što znači da slika ukupno sadrži do $2^8 = 256$ intenziteta sive boje.

Za neka područja djelatnosti koje iziskuju visokokvalitetan prikaz takav broj intenziteta je neprihvatljiv. Ponajprije su to slike dobivene pomoću različitih uređaja, npr. u medicini CT. U takvim slikama broj prihvatljivih nijansi je oko 2000 te je za pohranu potrebno oko 10-12 bitova. Za primjenu na računalima pogodnije je uzeti 16 bitova (2 bajta).

Bez obzira koja je dubina (broj bitova) korištena vrijednost 0 uvijek označava crnu, dok maksimalna vrijednost u tom zapisu (255 za 8 bitova po pikselu, 65535 za 16 bitova po pikselu…) označava bijelu nijansu.

2.1.2. Pretvaranje boja u sivu paletu

Pretvaranje boja u sivu paletu nije jednoznačno. Zastupljenost crvene, zelene odnosno plave komponente varira od uređaja kojim se snimi slika. Uobičajena strategija

pri pretvaranju je pogoditi što sličniju svjetlinu slike sivih tonova sa svjetlinom boja originalne slike.

Prvi korak pri konverziji jest ekstrakcija informacije piksela, odnosno njegove R (*eng. red*), G (*eng. green*) i B (*eng. blue*) komponente. Tada se nova vrijednost piksela dobiva pomoću:

$$novaVrijednostPiksela = 0.3 * R + 0.59 * G + 0.11 * B \quad (2.1)$$

Kako bi se izračunavanje ubrzalo preporučuje se uporaba cjelobrojnih vrijednosti, pa se također upotrebljava:

$$novaVrijednostPiksela = (11 * R + 16 * G + 5 * B)/32 \quad (2.2)$$

Slika 2.1 Početna slika (lijevo) i slika sivih nijansi (desno)

Posebnosti implementirane metode

Radi bržeg izvođenja sve pretvorbe su vršene na 32-bitnom zapisu, pa se bijela nijansa za crno-bijelu sliku pogrešno prikazuje. Ovdje su tonovi sive boje prikazani kao tonovi plave boje, gdje svjetlija nijansa plave odgovara svjetlijoj nijansi sive boje.

2.2. Sobelova metoda detekcije rubova

Detekcija rubova je osnova u većini postupaka obrade slike ili postupku praćenja. Vrlo često se upotrebljava kako bi se dobila informacija o objektima prije segmentacije i ekstrakcije značajki. Također upotrebljava se kako bi poboljšao izgled zamućenih slika iz slikovnih sekvenci videa.

Ovdje se upotrebljava kao jedna od faze pripreme objekta za praćenje. Ona pokušava detektirati rubove objekta, odnosno granice između objekta ili granice između pozadine i objekta.

Osnova rada

Provjerava se razlika između pojedinih slikovnih elementa, te ako je ona veća od unaprijed određene granice uzima se da je to rub. Ovdje je samo razlika dviju piksela pisana kao vrijednost novog, nije određivana neka granica.

2.2 Vertikalna i horizontalna promjena (boje) - gradijent

Kako bi se dobio vertikalni rub gleda se promjena u horizontalnom smjeru za neki piksel, dok za horizontalne gleda se vertikalni smjer. Jačina gradijenta dobiva se pomoću:

$$\nabla f = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \tag{2.1}$$

Zbog brzine računanja pokazuje se da je relativno dobra aproksimacija:

$$\nabla f = \left|\frac{\partial f}{\partial x}\right| + \left|\frac{\partial f}{\partial y}\right| \tag{2.2}$$

Gradijent $\frac{\partial f}{\partial x}(x,y)$ možemo zapisati kao razliku među susjednim horizontalnim pikselima u odnosu na neki piksel, dok $\frac{\partial f}{\partial y}(x,y)$kao razliku među pikselima u vertikalnom smjeru u odnosu na neki piksel.

Takav zapis je opravdan jer vrijedi:

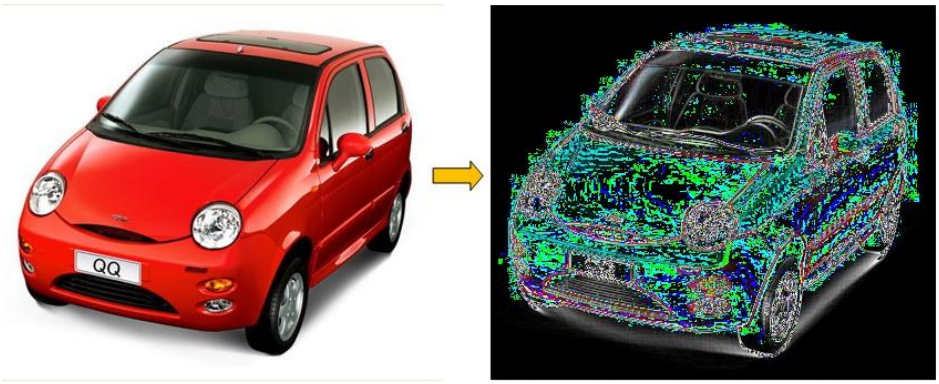
$$\frac{\partial f}{\partial x}(x,y) \approx f(x+1,y) - f(x,y) \text{ i } \frac{\partial f}{\partial y}(x,y) \approx f(x,y+1) - f(x,y) \tag{2.3}$$

Dakle gradijente možemo zapisati kao $Gx = [-1 \quad 0 \quad 1]$ i $Gy = [-1 \quad 0 \quad 1]^T$, gdje Gx i Gy označava vektor-matricu za detekciju vertikalnih, odnosno horizontalnih rubova.

Algoritam:

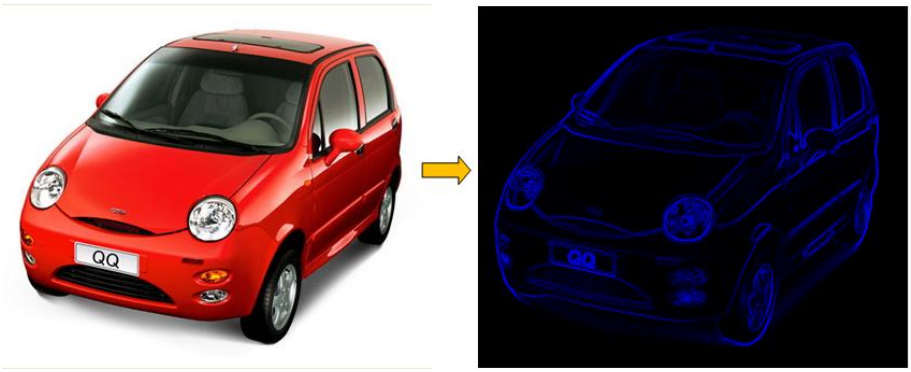
1. Pretvoriti boje slike u tonove sive boje
2. Za svaki piksel izračunati $\frac{\partial f}{\partial x}$ i $\frac{\partial f}{\partial y}$ pomoću matrica Gx i Gy pomnoživši vrijednosti piksela sa odgovarajućim brojevima matrice i na kraju te vrijednosti zbrojivši.
3. Izračunati ukupnu jačinu gradijenta pomoću izraza 1 ili 2
4. Ograničiti izračunatu jačinu (obično su graničnici 0 i 255 za slike sivih tonova)
5. Zapisati takvu vrijednost kao novu vrijednost onog piksela za koji se provodi proračun jačine gradijenta (središnji element matrica Gx i Gy).

Važno je napomenuti važnost prvog koraka. Ukoliko se ne bi napravio dobila bi se neodgovarajuća slika.



Slika 2.3 Detekcija rubova na sliku u boji

Ovako opisanim postupkom i takvim matricama bi se već mogao načiniti jednostavan detektor rubova.



Slika 2.4 Detekcija rubova dobivena gore opisanim postupkom

Iako nisu u obzir uzete vrijednosti nikakvih drugih okolnih piksela takav detektor je uspio detektirati rubove koji su pod nekim kutom u odnosu na horizontalu ili vertikalu. Takvi rubovi se detektiraju samo za intenzivne granice među bojama (npr. na slici 2.3 spojevi

karoserije automobila), dok ostali manje intenzivni prijelazi nisu detektirani kako rub (slika 2.4. prijelazi među spojevima sjedala, te dijelovi karoserije).

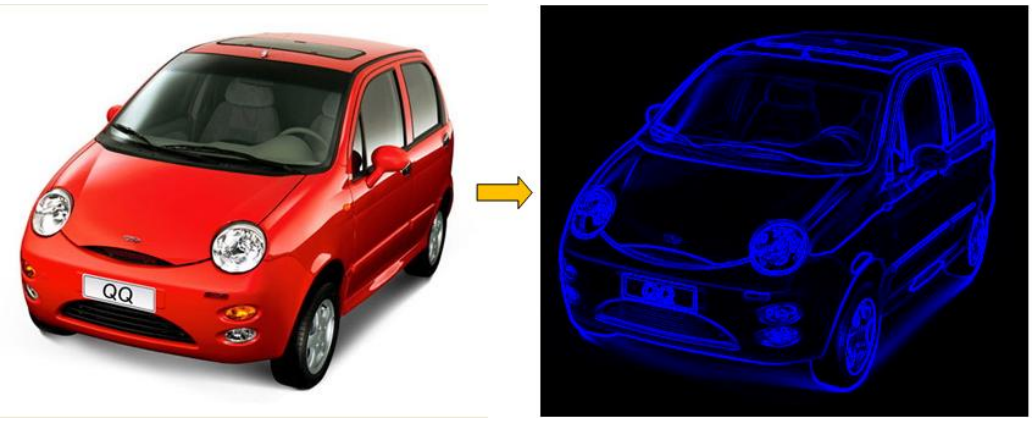
Kako rubovi mogu biti u različitim smjerovima, te postoje manje intenzivni prijelazi koje bismo također željeli detektirati kao rubove potrebno je gledati okolinu piksela; najčešće su to njegovi neposredni susjedi.

Tako matrice Gx i Gy prelaze u:

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \text{ i } Gy = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Takve matrice zovu se Prewittov horizontalni i vertikalni operator, a detekcija pomoću njih gore opisanim postupkom detekcija rubova pomoću Prewitta.

Detekcija rubova dobivena takvim matricama je:



Slika 2.5 Detekcija rubova dobivena gore napisanim matricama Gx i Gy

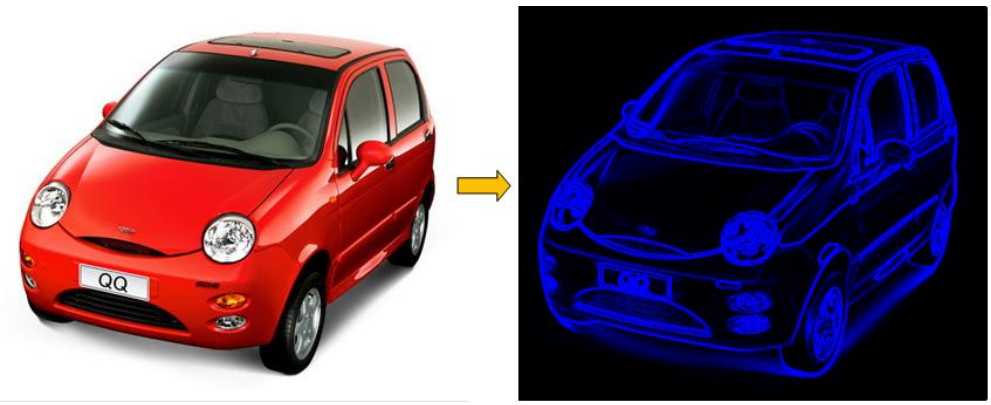
Na slici (desno) dobivenoj detekcijom rubova zamjetno je poboljšanje u odnosu na prethodnu detekciju pomoću matrica-vektora (Slika 2.4). Neki prijelazi također još uvijek nisu prepoznati kao rubovi (prijelazi između komponenti sjedala). Tome se može doskočiti tako da se gradijent $\frac{\partial f}{\partial x}$ i $\frac{\partial f}{\partial y}$ bolje aproksimiraju što je postignuto davanjem određenih težina okolnim pikselima. Konkretno:

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{ i } Gy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Matrica 2.1 Sobelov horizontalni i vertikalni operator

Matrice množi $\frac{1}{8}$. Množenje sa $\frac{1}{8}$ je važno samo u slučaju dobivanja ispravne vrijednosti gradijenta, pa se izostavlja prilikom detekcije rubova.

Detekcija pomoću ovakvih matrica gore opisanim postupkom zove se detekcija rubova pomoću Sobela. Detekcija rubova dobivenih pomoću ovakvih matrica je:

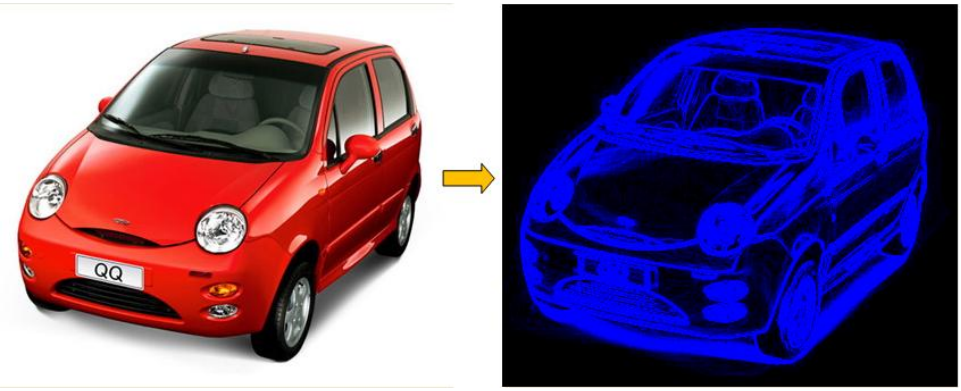


Slika 2.6 Detekcija rubova pomoću Sobela

Postoji modifikacija takvih matrica koja je osjetljiva na manje izražene razlike među vrijednostima piksela.

$$Gx = \begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix} \text{ i } Gy = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix}.$$

Takve matrice zovu se Sharrov horizontalni i vertikalni operator, a detekcija pomoću njih gore opisanim postupkom detekcija rubova pomoću Sharra. Detekcija rubova dobivenih pomoću ovakvih matrica je:



Slika 2.7 Detekcija rubova pomoću Scharra

2.2.1. Dobivanje smjera gradijenta

Smjer gradijenta može se dobiti pomoću:

$$\varphi = tan^{-1}\left(\frac{\partial f}{\partial y}/\frac{\partial f}{\partial x}\right)$$

2.2.2. Svojstva i ostale metode

Pozitivno svojstvo takve metode je brzina računanja, te dobri rezultati u gotovo svim slikama.
Negativno je što je opisana metoda nije otporna na šumove, pa se u nekim slučajevima prije same ekstrakcije rubova koriste filtri za glađenje slike kao što je Gaussov filter, no njegovom uporabom nepovratno se gube detalji slike.

Od ostalih metoda izdvaja se Cannyeva metoda ekstrakcije rubova. Prednost je otpornost na šum, dobri dobiveni detalji rubova, ali tu su i negativna svojstva poput puno skupljeg računanja u odnosu na objašnjenu metodu, te samu značajku što je takav postupak parametriziran.

Za ovaj rad izabran je Sobelova metoda detekcije rubova zbog brzine računanja, dobrih rezultata, ali i zbog toga što je neparametrizirana metoda za razliku od Cannyeve metode što uvelike olakšava ovaj postupak.

Uporaba Cannyevog postupka na objekte koji se u vremenu udaljavaju i približavaju ili koji su teže razlučljivi od okoline iziskuje nalaženje optimalnih parametara. Izborom jednih parametara dobivaju se dobri rezultati za jedan dio objekata dok za drugi ne. Sobelov postupak se pokazao vrlo dobrim za sve slučajeve koje su se pojavili prilikom praćenja objekta.

2.3. Stucki dithering

2.3.1. Dijeljenje greške (*eng. Error diffusion*)

Dijeljenje greške je metoda kojom se pri kvantizaciji slike greška proširuje na okolne piksele pri čemu se postiže bolja vidljivost slikovnih elemenata. Veoma je raširena uporaba prilikom ispisa slika u boji na pisaču koji podržava samo crno-bijelu tehniku.

U slijedećim odlomcima objasnit će se princip rada dijeljenja greške na slikama. Ako nije drugačije rečeno podrazumijevat će se da pikseli slika imaju vrijednosti palete sivih tonova.

Algoritam – za jednu dimenziju

Algoritam kreće s lijeva na desno. Uzima se vrijednost trenutnog piksela i uspoređuje se sa $(vrijednostBijelogTona - vrijednostCrnogTona)/2$. Ako je vrijednost piksela veća od te vrijednosti nova vrijednost postaje bijela, inače piksel poprima vrijednost crne nijanse. Nova vrijednost piksela je crna ili bijela nijansa pa očito postoji greška. Greška se izražava kao $staraVrijednostPiksela - novaVrijednostPiksela$ te se raspršuje na okolne piksele. Zatim se prelazi na sljedeći piksel i postupak se ponavlja iterativno.

Algoritam – za dvije dimenzije

Algoritam koji širi grešku samo na njemu susjedne piksele u istom retku gdje je i on sam ima za posljedicu pojavu artefakata u slici koji se očituju kao vertikalne linije. Algoritam za dvije dimenzije širi grešku i na donje susjedne piksele tako da taj efekt nestaje. Npr. jedan od načina širenja greške jest: 50 % greške dodajemo desnom susjedu, a po 25 % greške dodaje se susjedu ispod i dolje-desno. Matrično bismo to zapisali:

$$\frac{1}{4}\begin{bmatrix} \# & 2 \\ 1 & 1 \end{bmatrix}$$

Matrica 2.2 Primjer matrice za disipaciju greške

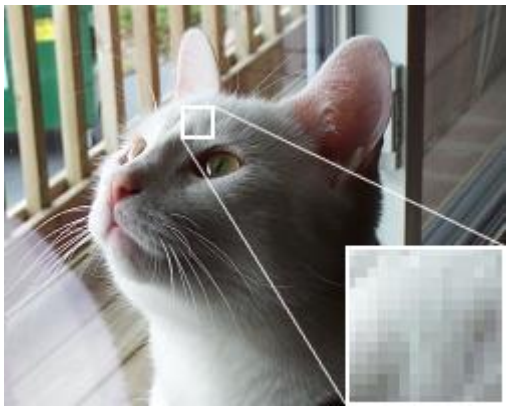
Nazivnik u razlomku dobije se kao suma brojeva matrice. Znak „#“ označava trenutni piksel koji se obrađuje. Radi jednostavne implementacije dvodimenzionalnim poljem stavlja se vrijednost 0 ili bilo koja druga jer se ionako taj položaj u matrici ne gleda.

Algoritam primijenjen na slike u boji

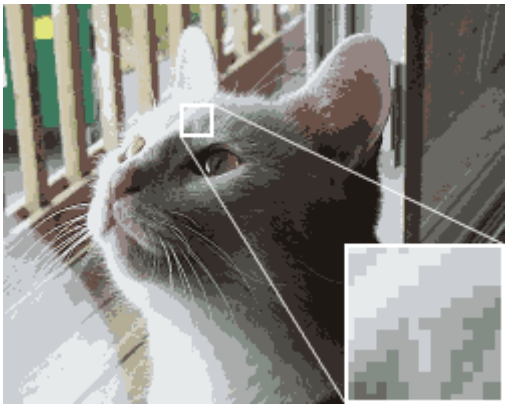
Algoritam se također može primijeniti na slike u boji, no umjesto odabira dvije krajnje vrijednosti odabire se neka najbolje odgovarajuća zamjena za staru boju. Npr. za pretvaranje 8-bitnih boja u 4-bitne može se upotrijebiti sljedeća formula:

$$novaVrijednostPiksela = \frac{\left(staraVrijednostPiksela + \frac{2^2}{2}\right)}{2^2}$$

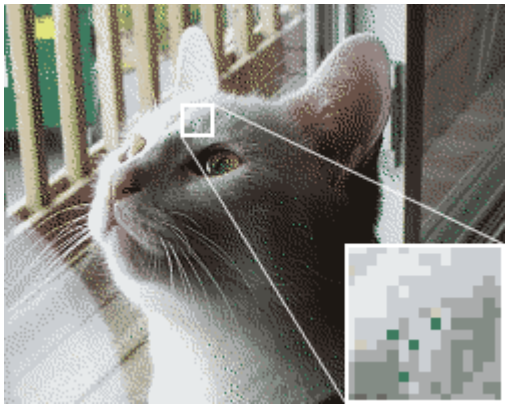
Primjer pretvaranja 8-bitne (256 boja) slike u 4-bitnu (16 boja):



a)



b)



c)

Slika 2.8 a) Originalna slika u 256 boja. b) Kvantizirana slika u 16 boja bez prenašanja greške. c) Kvantizirana slika u 16 boja sa prenašanjem greške.

2.3.2. Stuckijev model prenašanja greške

Prenošenje greške pomoću Stuckijeve matrice je vrsta prenošenja greške koja grešku dijeli na više manjih vrijednosti (obzirom na gore navedenu matricu) te ih time bolje raspodjeljuje među susjednim pikselima.

Stuckijeva matrica:

$$\frac{1}{42}\begin{bmatrix} - & - & \# & 8 & 4 \\ 2 & 4 & 8 & 4 & 2 \\ 1 & 2 & 4 & 2 & 1 \end{bmatrix}$$

Matrica 2.3 matrica za disipaciju greške pomoću Stuckijeovg modela

Znak " –„ označava piksele koji su već obrađeni, ali radi jednostavnije implementacije pomoću dvodimenzionalnog polja njihove vrijednosti se mogu staviti na bilo koju vrijednost jer se ionako ne koriste.

Algoritam za prenošenje greške pomoću Stuckijeve matrice:

```
For each y from top to bottom
      For each x from left to right
            oldPixel := pixel[x, y]
            newPixel := FindClosestPaletteColor(oldPixel)
            pixel[x, y] := newPixel
            quantError := oldPixel - newPixel
            //rasipanje greške na desne piksele
            pixel[x+1, y] := pixel[x+1, y] + 8/42 * quantError
            pixel[x+2, y] := pixel[x+2, y] + 4/42 * quantError
            //rasipanje greške na donje piksele
            pixel[x-2, y+1] := pixel[x-2, y+1] + 2/42 * quantError
            pixel[x-1, y+1] := pixel[x-1, y+1] + 4/42 * quantError
            …
            pixel[x+1, y+2] := pixel[x+1, y+2] + 2/42 * quantError
            pixel[x+2, y+2] := pixel[x+2, y+2] + 1/42 * quantError
```

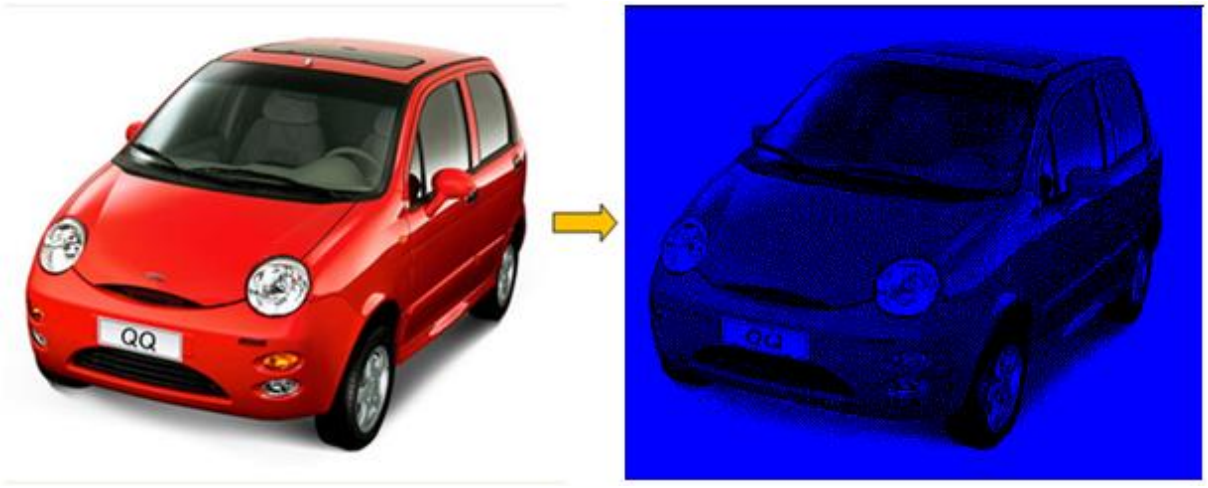
2.3.3. Binarizacija pomoću Stuckijeve metode

Za binarizaciju vrijedi gore navedeni algoritam, samo funkcija *FindClosestPaletteColor* vraća vrijednost bijelog tona ukoliko je greška veća od $(vrijednostBijelogTona - vrijednostCrnogTona)/2.$(*vrijednostBijelogTona* u slikama čiji elementi sadrže vrijednosti sivih nijansi 255, a *vrijednostCrnogTona* 0), a vrijednost crnog ukoliko uvjet nije zadovoljen.

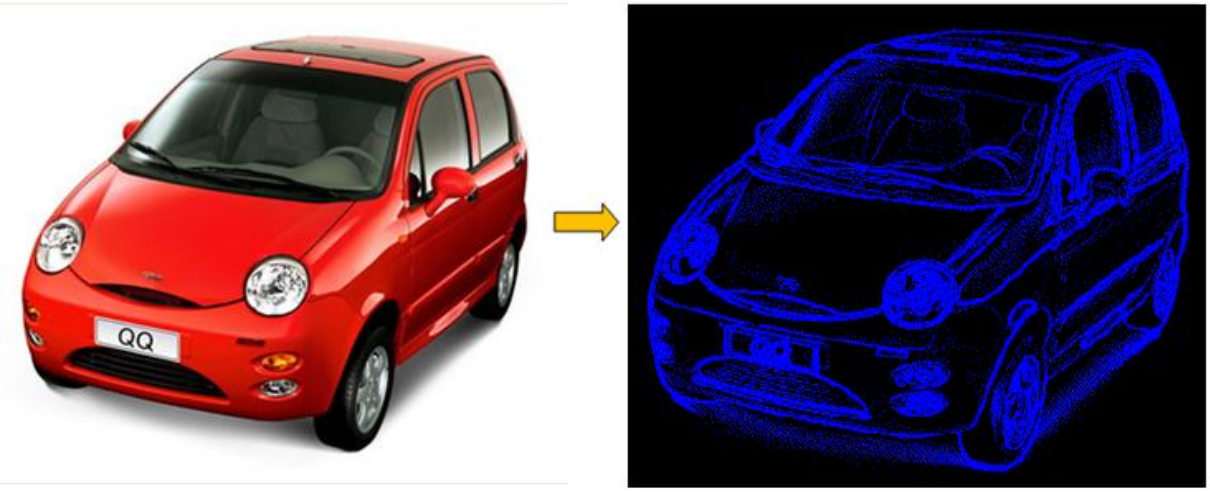
2.3.4. Usporedba Stuckijevog i ostalih modela

Razvijeno je nekoliko modela disipacija grešaka, a neki od njih su: Floyd-Steinberg, Sierra, Atkinson…

Ova metoda u implementaciji praćenja se provodila nakon detekcije rubova, te je pažnja bila usmjerena na očuvanje kvalitete rubova. Iako se Floyd-Steinbergov model često rabi, za potrebe praćenja se pokazuje da najbolje odgovara Stuckijev model koji zbog najveće matrice raspodjele grešaka najbolje rasipa greške na okolne piksele.



Slika 2.9 Primjer binarizacije Stuckijevim modelom

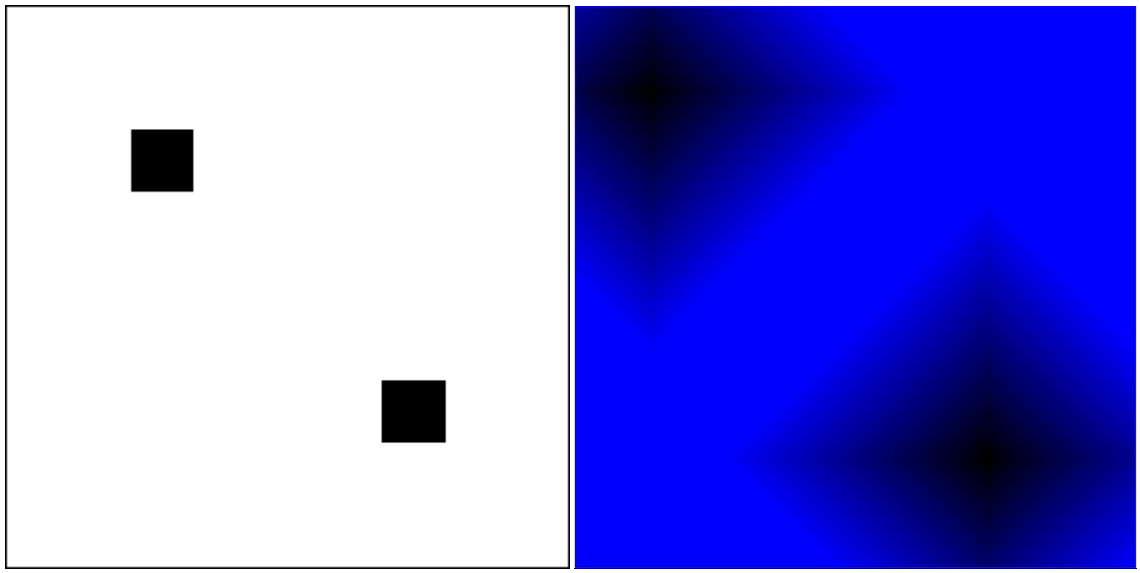


Slika 2.10 Primjer dobre očuvanosti rubova nakon njihove detekcije (Sobelova metoda)

2.4. Transformacija udaljenosti

Poznata je pod engleskim nazivom „*distance transform*“, skraćeno DT.

Neka imamo crno bijelu sliku nekih dimenzija. Crni pikseli neka imaju vrijednost 0 dok bijeli neku vrijednost veću od 0 (npr. 255). Transformacija udaljenosti je metoda kojom se izračunava udaljenost između svakog označenog elementa matrice. Postupak će biti prikazan na donjoj slici.



Slika 2.11 Crno-bijela slika dimenzija 9x9 i njezina pripadna transformacija udaljenosti

Crno-bijelu sliku možemo promatrati kao matricu koja sadrži samo dvije vrijednosti. Slika 2.11 predstavlja matricu dimenzija 9x9. Elementi za koje se izračunava transformacija udaljenosti označeni su crno. Nazivaju se prednji pikseli (*eng. foreground pixels*). Elementi označeni bijelo su pozadinski (*eng. background pixels*) . S desne strane je prikazana slika nakon obavljanja transformacije udaljenosti. Svaki prošli pozadinski element je poprimio određenu vrijednost koja predstavlja udaljenost do najbližeg prednjeg elementa. Metrika je u ovom slučaju „Manhattan“ koja je objašnjena u slijedećem poglavlju.

Važno je napomenuti da se transformacija udaljenosti radila na ekvivalentu desnog dijela gornje slike puno veće rezolucije kako bi se dobio dovoljno dobar prikaz.

2.4.1. Metrike udaljenosti

U uvodu je prikazana DT slika čije su vrijednosti dobivene „Manhattan“ metrikom. Postoji veći broj metrika udaljenosti,a bit će navedene neke češće korištene. Za svaku metriku prikazat će se 2D Kartezijev koordinatni sustav na kojem će biti označene dvije točke (crvena i zelena). Crvena predstavlja ishodište i ujedno je točka od koje se računaju udaljenosti. Udaljenosti će se prikazati za okolne točke, a posebno će se istaknuti udaljenost između dvije navedene točke.

Euklidska udaljenost

$$D_{Euclid} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{2.4}$$

Udaljenost koja je možda najčešća u uporabi. Ona je najkraća udaljenost između dviju točaka.

4	$\sqrt{32}$	$\sqrt{25}$	$\sqrt{20}$	$\sqrt{17}$	4				
3	$\sqrt{25}$	$\sqrt{18}$	$\sqrt{13}$	$\sqrt{10}$	3				
2	$\sqrt{20}$	$\sqrt{13}$	$\sqrt{8}$	$\sqrt{5}$	2		…		
1	$\sqrt{17}$	$\sqrt{10}$	$\sqrt{3}$	$\sqrt{2}$	1	$\sqrt{2}$			
0	4	3	2	1	**0**	1			
-1				$\sqrt{2}$	1	$\sqrt{2}$			
-2			…				…		
-3									
-4									
	-4	-3	-2	-1	0	1	2	3	4

Slika 2.12 Udaljenosti za Euklidsku mjeru. Označena je udaljenost između dviju točaka.

Manhattan udaljenost

$$D_{CityBlock} = |x_2 - x_1| + |y_2 - y_1| \quad (2.5)$$

Ova udaljenost pretpostavlja da od jedne do druge točke možemo jedino doći mičući se po koordinatnim osima (horizontali i vertikali). Dijagonalni pomaci nisu dopušteni. Znana je još i kao „City-block“ udaljenost.

4	8	7	6	5	4				
3	7	6	5	4	3				
2	6	5	4	3	2		…		
1	5	4	3	2	1	2			
0	4	3	2	1	0	1			
-1				2	1	2			
-2			…				…		
-3									
-4									
	-4	-3	-2	-1	0	1	2	3	4

Slika 2.13 Udaljenosti za 'City-block' mjeru. Označene su udaljenosti između dviju točaka. One su jednakovrijedne.

„Chessboard“ udaljenost

$$D_{Chess} = \max\left(|x_2 - x_1|,\ |y_2 - y_1| \right) \quad (2.6)$$

Udaljenost pretpostavlja da od jedne do druge točke možemo doći mičući se jedino horizontalno i vertikalno (kao Manhattan udaljenost). Uzima se najveća razlika udaljenosti u pojedinom smjeru.

4	4	4	4	4	4				
3	4	3	3	3	3				
2	4	3	2	2	2		…		
1	4	3	2	1	1	1			
0	4	3	2	1	0	1			
-1				1	1	1			
-2			…				…		
-3									
-4									
	-4	-3	-2	-1	0	1	2	3	4

Slika 2.14 Udaljenosti za 'Chessboard' mjeru. Označene su udaljenosti između dviju točaka. One su jednakovrijedne

2.4.2. Transformacija udaljenosti - Chamfer

Chamferova transformacija udaljenosti je takva transformacija koja pretpostavlja dobivanje vrijednosti udaljenosti nekoga piksela na temelju vrijednosti njegovih susjeda koji su na isti način izračunati. Znači pretpostavljena je propagacija udaljenosti kroz sliku. Takva je pretpostavka točna za regularne metrike za koje vrijedi:

Za sve p i q takve da je $dist_M(p,q) \leq 2$, postoji r drugačiji od p i q takav da je $dist_M(p,q) = dist_M(p,r) + dist_M(r,q)$.

Takve metrike su „City-block“, „Chessboard“, ali ne i Euklidska. Prema tome za metrike koje nemaju ovakvo svojstvo može se u najboljem slučaju načiniti samo aproksimacija.

Izračunavanje

Trivijalni algoritam bi bio pronaći udaljenost za svaka dva označena piksela (obično pikseli rubova slike). Očito takva metoda nije efikasna jer bismo imali složenost O(bxb), gdje je b broj takvih elemenata slike. Pokazuje se da postoje bolji načini za njezino izračunavanje.

Algoritam izračunavanja DT slike upotrebljava određenu masku s kojom prolazi 2 puta po slici. Primjer izračunavanja bit će pokazan na primjeru maske 3x3 koja implementira 'Manhattan' metriku udaljenosti.

b	a	b
a	0	a
b	a	b

2	1	2
1	0	1
2	1	2

Slika 2.15 Generička maska 3x3. Maska 3x3 za 'Manhattan' udaljenost

Piksel za koji se računa transformacija udaljenosti odgovara centralnoj poziciji na maski. Udaljenost od njega samog je naravno 0. Za njegove neposredne susjede udaljenost se izračunava prema formuli za 'Manhattan' udaljenost (izraz 2.5).

Maska se najprije razbije na dva dijela (Slika 2.15). Prvi dio je obojen crveno, drugi zeleno. Centralni element pripada i jednom i drugom dijelu pa je obojen mješavinom dvaju boja. Maska se može razlomiti i vertikalno, no tada se mijenja redoslijed prolaza kroz sliku. Ovdje je maska podijeljena horizontalno. Slijede objašnjenje koraka algoritma koji se sastoji od dvaju prolaza po slici.

Priprema slike

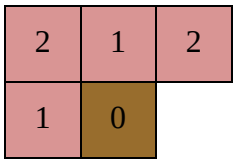
Slika se najprije pretvori u crno-bijelu ako već nije. Rubovi se postave na vrijednost 0, a svi ostali elementi na što veću vrijednost (npr. 999). Ovdje je velika vrijednost označena simbolom za beskonačnu vrijednost.

∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	0	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	0	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞

Slika 2.16 Početni izgled pripremljene slike (matrice).

Prvi prolaz

Uzima se dio maske, obojen crveno (Slika 2.15).

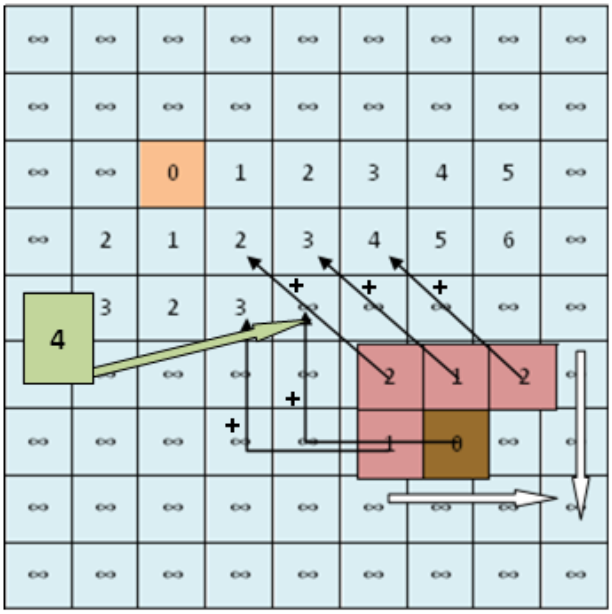


Slika 2.17 Dio maske za prvi prolaz

Maska se postavi na prvi element (piksel) koji to dozvoljava. Tako za masku 3x3 odgovara elementu na koordinati (1, 1) slike brojeći od 0 i od gornjeg lijevog kuta. Uzimaju se vrijednosti njegovih susjeda (i njega samog) te se uspoređuju sa odgovarajućim vrijednostima maske. Uzima se takva najmanja vrijednost.

$$d0 = \min(d + D(i), d0), \qquad (2.7)$$

gdje je *d0* vrijednost piksela za kojeg se udaljenost računa (odgovara srednjem elementu maske), *D(i)* i-ti element maske, a *d* vrijednost piksela koja odgovara određenom elementu maske. Važno je da pikseli do kojih se računa udaljenost imaju vrijednost 0. Kako se uvijek uzima minimum vrijednosti ne može se dogoditi da takav piksel poprimi bilo koju drugu vrijednost. Nakon postavljanja nove vrijednosti maska se pomiče desno, a nakon svakog obrađenog retka za jedno mjesto dolje. Postupak se dalje ponavlja dok se god ne dođe do pozicije na koju se maska ne može smjestiti. Za masku 3x3 posljednja valjana pozicija brojeći od gornjeg lijevog kuta je (r-1, s), gdje je r posljednji indeks retka, a s posljednji indeks stupca slike. Za sliku prikazanu dolje posljednja koordinata je (7,8) prema prethodnom navedenom načinu mjerenja. Dio postupka je predočen slikom Slika 2.18.



Slika 2.18 Prikaz dijela postupka obilaženja slike gornjom polovicom maske

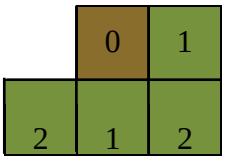
Na kraju prvog prolaza imamo:

∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	0	1	2	3	4	5	∞
∞	2	1	2	3	4	5	6	∞
∞	3	2	3	4	5	6	7	∞
∞	4	3	4	5	6	7	8	∞
∞	5	4	5	6	7	0	1	∞
∞	6	5	6	7	2	1	2	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞

Slika 2.19 Vrijednosti piksela nakon prvog prolaza

Drugi prolaz

Drugi dio maske se postavlja na donji desni rub slike.



Slika 2.20 Dio maske za drugi prolaz

Prva valjana koordinata je (n-2, k-2) gdje su n i k dimenzije slike. Broji se od (n-1, k-1). Ponavlja se isti postupak kao i za prvi prolaz samo što se sada maska giba prema lijevo i nakon svakog obrađenog reda gore. Na kraju drugog prolaza kao konačni rezultat imamo:

∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	2	1	2	3	4	5	6	∞
∞	1	0	1	2	3	4	5	∞
∞	2	1	2	3	4	3	4	∞
∞	3	2	3	4	3	2	3	∞
∞	4	3	4	3	2	1	2	∞
∞	5	4	3	2	1	0	1	∞
∞	6	5	4	3	2	1	2	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞

Slika 2.21 Vrijednosti piksela nakon obavljenog postupka

Maske drugih veličina

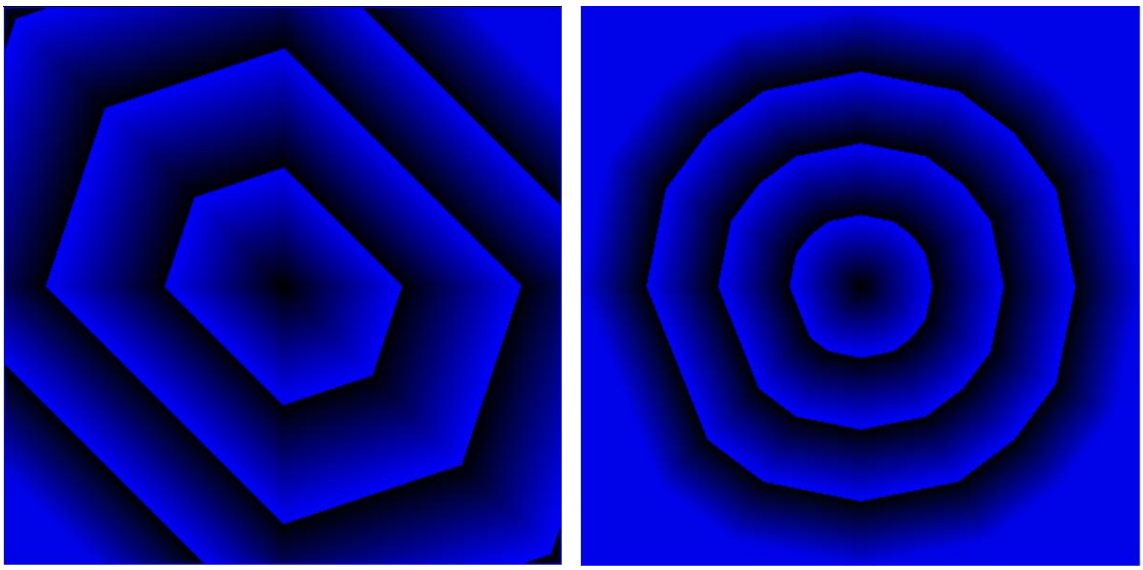
Često korištena metrika jest Euklidska. Rečeno je kako pomoću ovakvog računanja transformacije udaljenosti ne možemo točno izračunati Euklidsku udaljenost između piksela nego samo njezinu aproksimaciju. Utjecaj veličine maske ima ogroman učinak na točnost aproksimacije. Dobra strana maske veličine 3x3 jest brzina izračunavanja, ali za veće udaljenosti pogreška prilikom izračunavanja Euklidske udaljenosti se uvećava te se uvode maske drugih veličina. U implementaciji je korištena maska veličine 5x5.

2b	c	2a	c	2b
c	b	a	b	c
2a	a	0	a	2a
c	b	a	b	c
2b	c	2b	c	2b

Slika 2.22 Generička maska veličine 5x5

Vrijednosti elemenata maske koje nisu obojeni nije potrebno gledati pa se prilikom implementacije izostavljaju.

Razlike u kvaliteti rezultata prikazani su donjim slikama. Transformacija je obavljena za sliku koja ima označen samo središnji piksel.



Slika 2.23 Transformacija udaljenosti za središnji piksel pomoću maske 3x3 (lijeva slika) i maske 5x5 (desna slika)

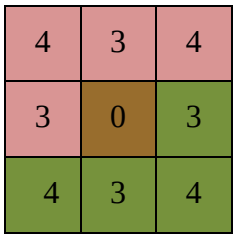
Vrijednosti elemenata maske

Maska veličine 3x3 za računanje aproksimacije Euklidske udaljenosti je:

$\sqrt{2}$	1	$\sqrt{2}$
1	0	1
$\sqrt{2}$	1	$\sqrt{2}$

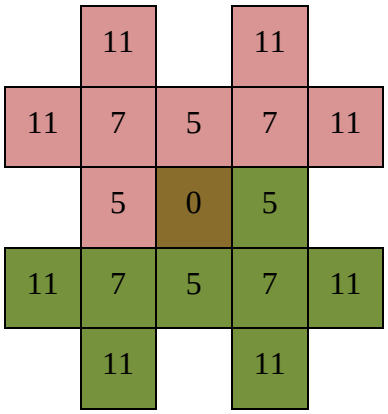
Slika 2.24 Vrijednosti piksela nakon obavljenog postupka

Izračunavanje na računalu se puno brže odvija nad cijelim nego nad decimalnim brojevima. Stoga se radi brzine računanja uvode cjelobrojne vrijednosti maske. Pokazuje se da su najbolje cjelobrojne vrijednosti koje zamjenjuju brojeve 1 i $\sqrt{2}$ su 3 i 4 jer se dijeljenjem vrijednosti udaljenosti elemenata slike sa manjom vrijednošću (u ovom slučaju 3) dobivaju približno iste vrijednosti kao da se koriste gore prikazana maska.



Slika 2.25 Maska veličine 3x3 koja sadrži aproksimacijske cjelobrojne vrijednosti

Za masku veličine 5x5 to su 5, 7 i 11. Dolje prikazana maska koristi se i u implementaciji algoritma za praćenje.



Slika 2.26 Maska veličine 5x5 koja sadrži aproksimacijske cjelobrojne vrijednosti

Greške učinjene pri korištenju maski

Greške se računaju u odnosu na stvarnu euklidsku udaljenost (*eng. Euclidean Distance Transform - EDT*).

Za maske veličine 3x3 navedene su u tablici. Simbol a i b označavaju elemente maske (slika 1. lijevi dio – generička 3x3 maska).

a	b	Vrsta udaljenosti	Greška
1	1	Chess-board	41.41%
1	2	City-block	29.29%
1	$\sqrt{2}$	Euklidska	7.61%
3	4	Euklidska	6.07%

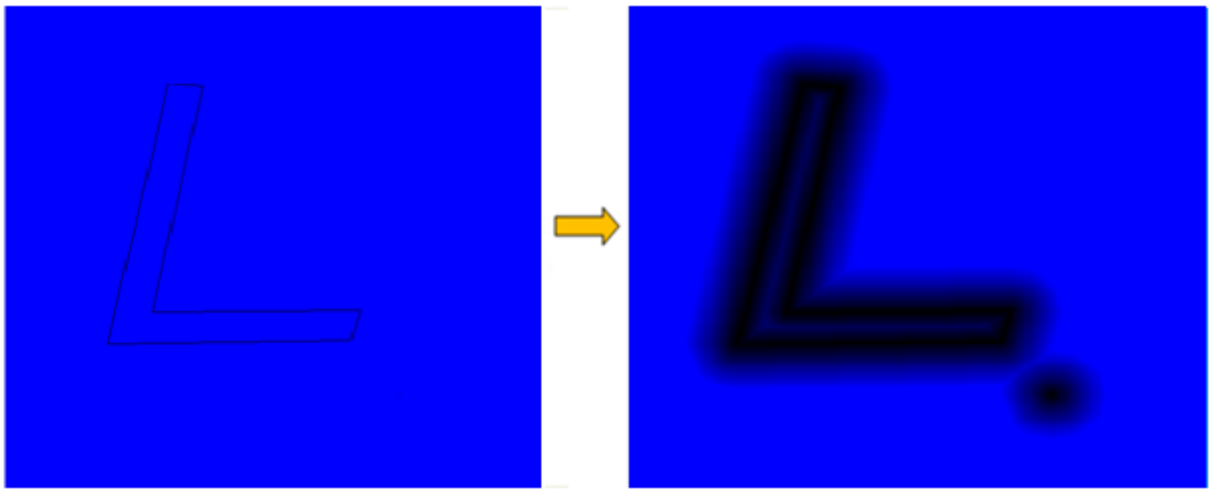
Slika 2.27 Greške maske 3x3 za različite vrijednosti

Korištenje aproksimacijskih cjelobrojnih vrijednosti za Euklidsku udaljenost (5,7,11) u masci 5x5 snižava grešku na manje od 2%.

Ovakva pozitivna strana je uvelike došla do izražaja prilikom točnosti praćenja objekta.

Primjeri

Primjer je načinjen pomoću maske 5x5 koja sadrži aproksimacijske cjelobrojne vrijednosti za Euklidsku metriku.



Slika 2.28 Primjer transformacije udaljenosti

3. Postupak praćenja

Praćenje se sastoji od tri faze. Faze su:

1. Faza izdvajanja objekta
2. Faza praćenja
3. Faza obnavljanja predloška objekta.

Faza izdvajanja objekta počinje korisnikovim odabirom objekta za praćenje. Definira se područje traženja. Traže se rubovi u dijelu slike koju je korisnik odabrao. Naposljetku pamti se slika dobivena izlučivanjem rubova. Takva slika predstavlja predložak objekta koji se mora naći u idućoj slici (*eng. template*). Time faza izdvajanja objekta završava.

Faza praćenja objekta kombinira 1. i 3. fazu. Traže se rubovi područja traženja koje je definirano u 1. fazi. Radi se transformacija udaljenosti tako dobivene slike. Nakon toga pretražuje se područje pretrage s ciljem pronalaska prošlog definiranog predloška. Nakon njegova pronalaska radi se njegovo ažuriranje. Bitno je naglasiti kako se radi ažuriranje nakon svake uzastopne slike (*eng. frame*) videa veza sa početnim izgledom objekta se gubi. To će imati za posljedicu da postupak praćenja nikad neće prijaviti da objekt nije pronađen, nego će predložak uvijek biti ažuriran na najbolje pronađen ekvivalent za stari predložak. Predložak se opet sprema za kasniju usporedbu.

Faza obnavljanja predloška objekta koristi se svaki put u fazi praćenja (3. faza). Kako i samo ime kaže zadatak joj je „obnoviti“ praćeni objekt. Pretpostavka je da se objekt mijenja, bilo kroz njegov oblik ili prikaz njegove veličine (skaliranje). Metoda je objašnjena u poglavlju 3.3. Iako metoda daje dobre rezultate, njezina nestabilnost se očituje kod promjene veličine objekta ako je objekt u bliskom kontaktu sa drugim. Zbog toga je isključena opcija obnavljanja veličine predloška iako u samoj implementaciji postoji podrška (metode) za to.

Sve tri faze čine jednu cjelinu. Postupak praćenja može se provesti ukidanjem 3. faze, ali samo kada objekt vrlo malo mijenja veličinu. Naravno za sve druge slučajeve rezultati su puno slabiji nego sa uključenom 3. fazom.

3.1. Faza izdvajanja objekta

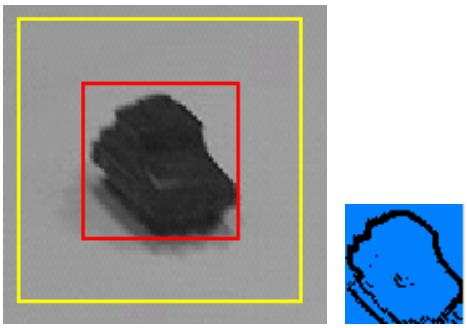
Postupak izdvajanja rubova objekta odvija se u nekoliko faza.

1. Korisnički odabir objekta
2. Definiranje područja pretrage
3. Pretvaranje slike predloška (korisnički odabir) u sliku sivih tonova (*eng. grayscale)*
4. Uporaba ekstrakcije rubova: Sobel
5. Uporaba binarizacije: Stucki
6. Spremanje dobivenog predloška

Koraci 3.-5. Spadaju u jednu cjelinu te se mogu nazvati ekstrakcija rubova.

3.1.1. Korisnički odabir objekta

Faza izdvajanja objekta je početna faza u praćenju objekta. Objekt se označava od strane korisnika pomoću kursora. Ovo je ujedno i najosjetljivija faza jer greške učinjene u odabiru objekta rezultirat će nemogućnošću njegova praćenja. U nastavku je prikazano nekoliko slika odabira objekta te pripadni rubovi odabranog dijela slike. Svaki prikaz je komentiran. Crveni pravokutnik predstavlja praćeni objekt kojega na početku zadaje korisnik. Područje pretrage objašnjeno u poglavlju 3.1.2 je prikazano žutim pravolutnikom.

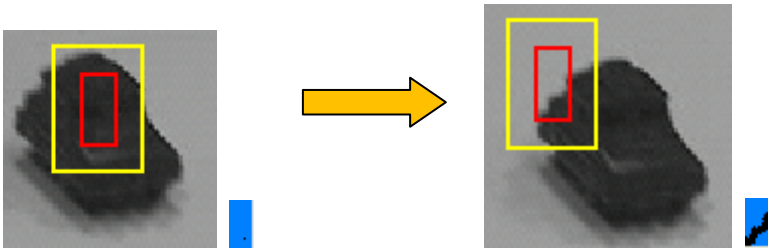


Slika 3.1 Primjer dobrog odabira objekta

Slika 3.1 prikazuje primjer dobrog odabira objekta. Objekt je u cijelosti odabran te sadrži dovoljnu količinu jedinstvenih rubova koje omogućavaju praćenje.

Također praćenje će se također ispravno odvijati ukoliko se odabere nešto veća površina oko objekta.

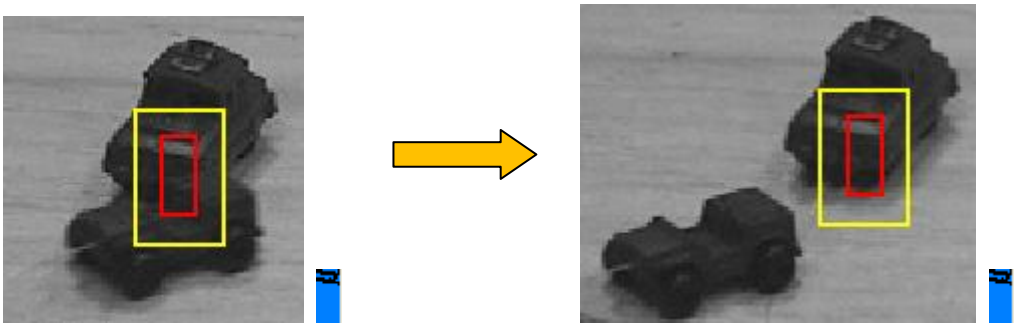
Donja slika prikazuje nepravilan odabir dijela za praćenje.



Slika 3.2 Primjer lošeg odabira objekta sa mogućnošću praćenja

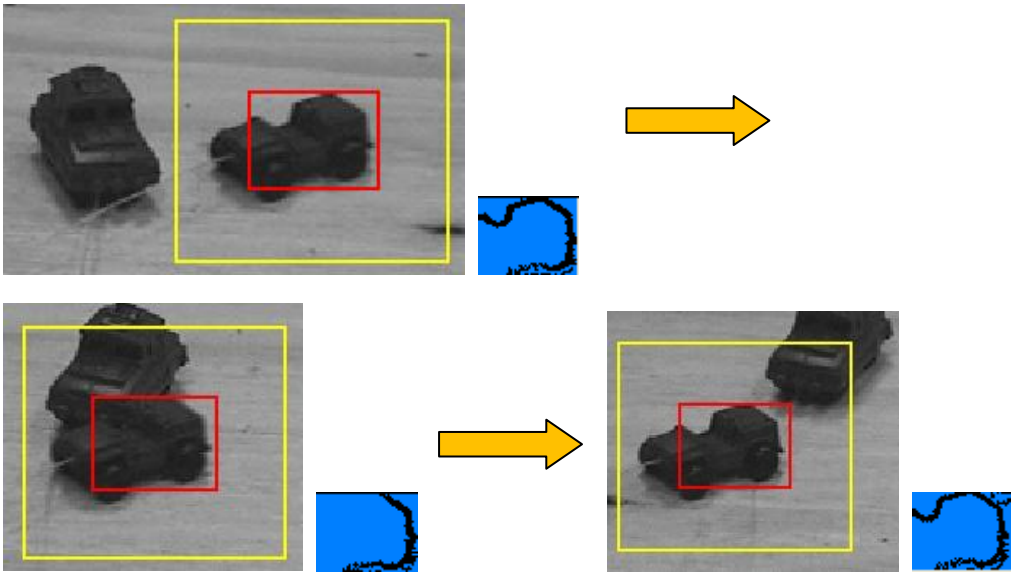
Odabran je dio objekta koji nema dovoljno značajki da bi se odvojio od okoline praćenja. Zbog toga se javlja nepravilnost u praćenju. Naime pokušat će se naći njemu najsličniji dio. Njegov ekvivalent može biti bilo što. Stoga će se pokušat pratiti prvi dio objekta ili okoline koji ima dovoljno značajki da se izdvoji. Kako se objekt kreće prema dolje-desno prvi dio koji će pratiti jest gornji-lijevi dio objekta kako je prikazano desnim dijelom gornje slike.

U slučaju da je objekt bio spojen sa drugim objektom te da se odabrao dio koji nema značajki koji ga izdvaja od okoline praćenje ne bi bilo valjano, odnosno ovisilo bi od slučaja do slučaja. Slika dolje prikazuje neispravan odabir dijela za praćenje. U ovom slučaju praćenje nije ispravno.



Slika 3.3 Primjer lošeg odabira objekta bez mogućnosti praćenja.

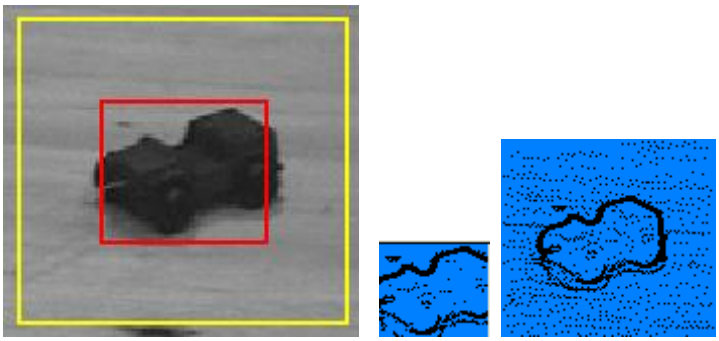
U slučaju da se objekt ispravno odabrao praćenje bi bilo moguće iako objekti nakratko postaju jedna cjelina.



Slika 3.4 Primjer valjanog praćenja objekta uz dobar odabir i sudaranje objekata.

3.1.2. Definiranje područja pretrage

Kako se objekt kreće očito je da će se pomaknuti u neku stranu. Kako bi ga se moglo uspješno pratiti treba pretražiti neku okolinu oko onoga dijela kojeg se prati. Područje pretrage na trenutnoj slici praćenja je prikazano žuto obojenim pravokutnikom.



Slika 3.5 Sa lijeva na desno. Slika praćenja objekta koju vidi korisnik. Slika rubova objekta. Slika rubova područja traženja.

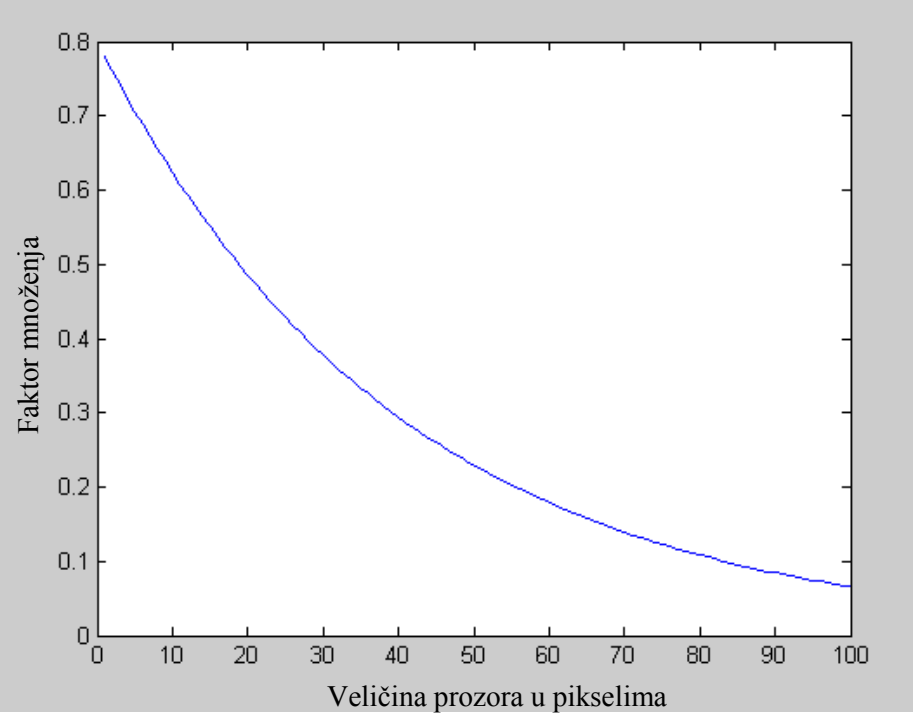
Veličina područja pretrage dobiva se proširivanjem područja praćenja za umnožak nekog faktora i neke od veličina područja praćenja (ovdje je odabrana njegova širina).

$$newSearchArea = trackingArea + mulFactor * trackingAreaWidth \quad (3.1)$$

Kako je za manje veličine prozora traženja poželjan što veći *mulFactor* (tipično 0.5-0.8), a za veće veličine može se smanjiti na oko 0.2 - 0.4, bolje je koristiti neku funkciju pomoću koje će se računati *mulFactor* nego rabiti konstantu. Ovdje je zbog navedenog razloga korištena padajuća eksponencijalna funkcija koja je i prikazana donjom slikom.

$$mulFactor = 0.9 * e^{-0.015*x} \quad (3.2)$$

Parametar x je neka od veličina prozora praćenja. U implementaciji se uzima minimum njegove duljine i širine.



Slika 3.6 Funkcija za računanje *mulFactor*-a

3.1.3. Ekstrakcija rubova

U ekstrakciju rubova spada:

1. Pretvaranje boja slike u sivu paletu
2. Ekstrakcija rubova Sobelovom metodom
3. Binarizacija slike raspršivanjem greške pomoću Stucki matrice

Sama ekstrakcija rubova odvija se Sobelovom metodom kako je pokazano u poglavlju 2.2. Da bi se ona mogla obaviti najprije se vrši pretvaranje u skalu sivih tonova što pokazuje poglavlje 2.1. Binarizacija ne spada u ekstrakciju rubova, ali ovdje je uvrštena jer računanje slike transformacije udaljenosti zahtijeva kao ulaz binarnu sliku. Razlog odabrane vrste binarizacije i njezina provedba opisane su u poglavlju 2.3.

Važno je napomenuti da se ekstrakcija rubova u fazi izdvajanja objekta vrši na dijelu slike odabranom od strane korisnika. U fazi praćenja ekstrakcija rubova se vrši na području pretrage koje je nešto veće od samog predloška objekta (poglavlje 3.1.2).

3.1.4. Spremanje dobivenog predloška

Predložak dobiven korisničkim odabirom pohranjuje se u varijablu radi buduće usporedbe. Prilikom spremanja ne provjerava se nikakva njegova valjanost kao npr. minimalni broj točaka rubova, veličina i sl. Jedino prilikom korisničkog odabira gleda se veličina odabrane površine. Najmanja površina koja se može odabrati jest veličine 4x4. Ovaj korak osim što se izvršava u 1. fazi također se svaki put izvršava i u 3. fazi. Kako je zbog nestabilnosti izuzeta mogućnost prilagođavanja veličine predloška objektu njegova veličina je uvijek stalna. Ipak prilikom svakog spremanja njegov prikaz se ažurira na trenutno praćeni dio slike. Ovo ima jednu negativnu posljedicu, a to je da se postepeno može izgubiti sličnost sa prvobitnim odabranim objektom ako objekt nestaje pa tako predložak može pokazivati na sasvim drugi dio slike, odnosno kroz vrijeme bit će prilagođen najsličnijem dijelu slike. Kroz njegovo stalno ažuriranje nemoguće je detektirati kada je objekt nestao jer veza s njime nije očuvana.

3.2. Faza praćenja

U poglavlju 2.4.2 detaljno je opisana metoda transformacije udaljenosti koja je i implementirana. Maska koja se koristi je veličine 5x5 sa cjelobrojnim koeficijentima koji aproksimiraju Euklidsku udaljenost.

U nastavku će biti opisana metoda kojom se uspoređuje i pronalazi predložak. Objašnjenje će biti popraćeno primjerima.

3.2.1. Loše strane binarne korelacije

Slikom dolje predstavljena je neka binarna slika koja se sastoji od dva crna piksela. Desno je neki uzorak koji želimo naći u takvoj slici. Očito je da takav isti uzorak ne postoji pa tražimo njegov najbolji ekvivalent.

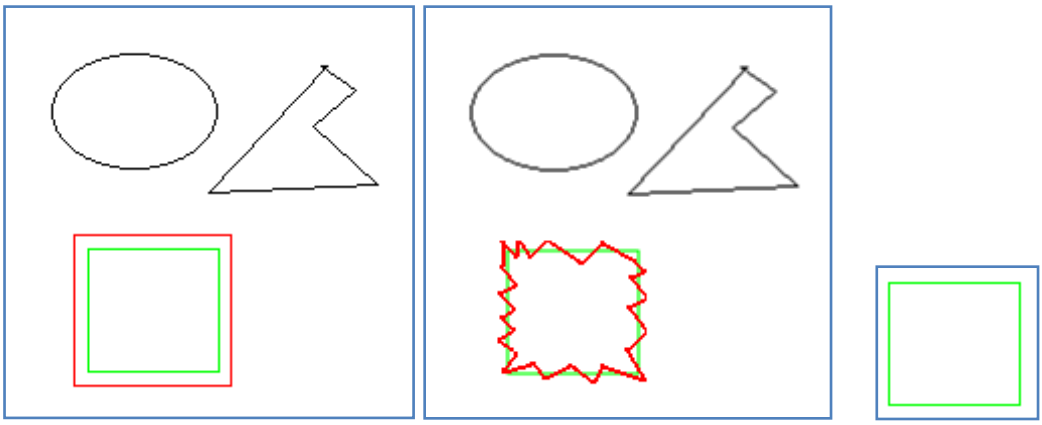
Ako bismo tražili binarnom korelacijom tada bismo nekim obilaskom slike uspoređivali piksele koje imaju vrijednost 0 te bismo u obzir uzeli ona područja koja najbolje odgovaraju traženom uzorku. Donja slika pokazuje realniji slučaj odnosno djelomično nepoklapanje uzorka i dijela slike koji mu odgovara. Ipak binarna korelacija pokazala bi se uspješnom u ovom slučaju.

255	255	255	255	255	255	255	255	2555
255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255
255	255	255	0	0	255	255	255	255
255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	2555
255	255	255	255	255	255	255	255	255

255	255	255	255
255	0	255	255
255	255	0	255

Slika 3.7 Slika (lijevo i sredina) i uzorak koji tražimo (desno)

Međutim vrlo često u praćenju objekata sam objekt se neće samo translatirati nego i promijeniti veličinu bilo skaliranjem (umanjivanjem ili uvećavanjem) ili promjenom oblika. Jedna od situacija koja bi bila vrlo loša za binarnu korelaciju, a koja se vrlo često događa je prikazana slikom Slika 3.8.



Slika 3.8 Slika (lijevo i sredina) i uzorak koji tražimo (desno). Na slici (lijevo i sredina) zeleno je označeno najbolje podudaranje uzorka.

Gornja slika također prikazuje nepoklapanje uzorka, ali ovaj puta puno gori slučaj barem što se tiče binarne korelacije. Naime najbolje poklapanje i onaj rezultat što će dati binarna korelacija neće biti isti. Za lijevi dio slike rezultat će biti neka točka kvadrata koja će ovisiti o smjeru kako smo obišli sliku. Za srednji dio uzorak će se možda bolje poklopiti sa nekim drugim objektom s kojim će imati više zajedničkih točaka. Lijevi dio odgovarao bi skaliranju objekta dok sredina nesavršenoj ekstrakciji rubova ili promijeni oblika objekta. Ovakve situacije u praćenju su vrlo česte te kako je pokazano binarna korelacija nije pogodna za rješavanje problema nalaženja uzorka u takvim slikama.

U poglavlju 2.4.2 detaljno je opisana transformacija udaljenosti. Ako bismo od slike načinili sliku transformacije udaljenosti te na neki način usporedili sa uzorkom dobili bi smo puno veću robusnost i točnost u odnosu na binarnu korelaciju. Osjetljivost na promjenu skaliranje objekta bila bi uvelike smanjena, a bile bi dopuštene puno veće tolerancije na promjenu u veličini i obliku objekta u odnosu na binarnu korelaciju. Način usporedbe opisan je slijedećim poglavljem.

3.2.2. Usporedba sličnosti pomoću transformacije udaljenosti (Chamfer)

Usporedba slika i uzorka bit će objašnjen kroz primjer.

Ako od slike Slika 3.7 (lijeva) načinimo sliku transformacije udaljenosti dobit ćemo sliku kakva je prikazana dolje. Radi jednostavnosti upotrjebljena je matrica dimenzija 3x3 za „Manhattan“ udaljenost. Uzorak je slika Slika 3.7 (desna).

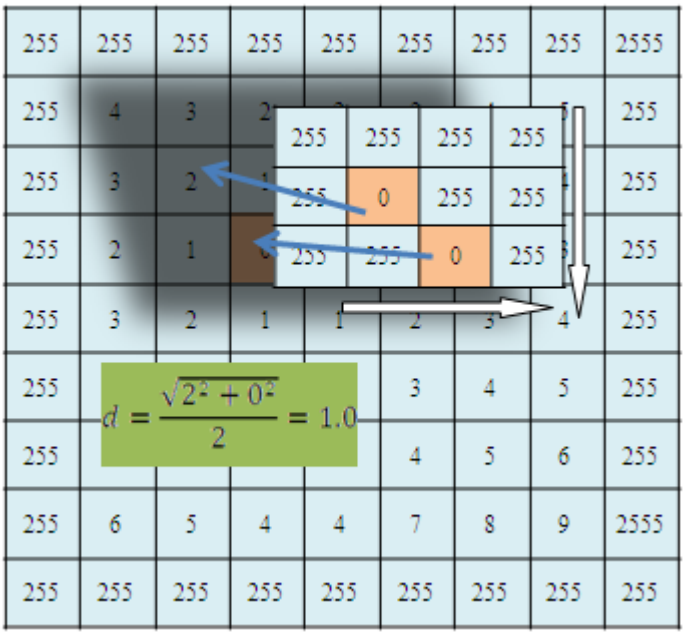
255	255	255	255	255	255	255	255	2555
255	4	3	2	2	3	4	5	255
255	3	2	1	1	2	3	4	255
255	2	1	0	0	1	2	3	255
255	3	2	1	1	2	3	4	255
255	4	3	2	2	3	4	5	255
255	5	4	3	3	4	5	6	255
255	6	5	4	4	7	8	9	2555
255	255	255	255	255	255	255	255	255

Slika 3.9 Transformacija udaljenosti slike Slika 3.7

Slika se obilazi na takav način da se obiđu svi pikseli. Rješenje koje je implementirano obilazi sliku desno i prema dolje. Ako je tako definirano obilaženje slike uzorak se na početku stavlja u gornji lijevi rub slike. Uzimaju se samo one vrijednosti piksela slike čija je odgovarajuća vrijednost piksela u uzorku 0. Sličnost toga dijela slike sa uzorkom se izražava preko

$$d = \frac{\sqrt{\sum_{i=o}^{n} {v_i}^2}}{n} \tag{3.3}$$

gdje je v_i vrijednost piksela čiji mu je odgovarajući piksel u uzorku 0, a n broj takvih piksela. Ako se rabi maska veličine 3x3 tada rubni vrijednosti rubnih piksela slike nisu izračunati. Da bi se postupak ubrzao potrebno je provjeravati samo 1 do (n-1) elementa reda, te 1 do (k-1) elementa stupca slike, gdje su n i k dimenzije slike. Početni korak postupka prikazuje donja slika.



Slika 3.10 Početni korak obilaženja slike pri traženju najsličnijeg područja uzorku. Sličnost je računata prema izrazu 3.3.

Sličnost uzorka i dijela slike pokrivenim prvim korakom jest 1.0. Kada se obiđe cijela slika ustanovit će se da se najveća sličnost dobiva za koordinate (2, 2) brojeći od gornjeg-lijevog ruba slike od nule gdje prvi broj predstavlja pomak u desno, a drugi pomak prema dolje. Vrijednost d će tada iznositi $d = \frac{\sqrt{0^2+1^2}}{2} = 0.5$. Može se primijetiti da je određeni dio slike tim što više sličniji uzorku što je njegova vrijednost niža. Za savršeno poklapanje uzorka i određenog dijela slike bi iznosila 0. Kako ovdje uzorak u potpunosti ne odgovara niti jednom dijelu slike ta vrijednost neće nikad iznositi 0.

3.3. Faza obnavljanja predloška objekta

Faza obnavljanja predloška je ujedno najzahtjevnija i najkritičnija. Prilikom praćenja objekta ako se pretpostavi da se objekt mijenja bilo promjenom veličine, oblika ili zastupljenosti u slici (blijeđenje, izlazak iz okvira) tada da bi njegovo praćenje bilo uspješno podaci o njegovom izgledu moraju biti ažurirani. Isto tako da bi praćenje bilo uspješno te promjene moraju biti malene. Neke veće promjene kao što je naglo translatiranje objekta između susjednih praćenih slika mogu se tolerirati povećanjem područja pretrage dok naglo skaliranje ili promjena oblika ne. Nesavršena ekstrakcija rubova dodatno otežava ovaj zadatak. Realan je slučaj da objekt naglo stupi u koliziju s nekim drugim objektom te u tom slučaju njihovi rubovi se stapaju. Takve slučajeve treba tolerirati. Razvijen je originalan pristup ovome problemu koji se pokazao djelomično uspješnim kako će biti pokazano u nastavku.

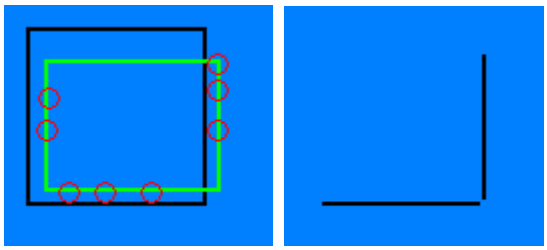
3.3.1. Opis obnove predloška

Obnova predloška izvodi se pomoću tri algoritma. Prvi je pronalaženje ekvivalentnog piksela, drugi širenje ruba kojemu pripada pronađeni ekvivalenti piksel, a treći ograničavanje veličine predloška na vanjske rubova obnovljenog objekta.

Nakon što se objekt pronađe u slijedećoj slici videa njegove granice su nešto promijenjene. Za svaki piksel starog predloška pokušava se pronaći ekvivalenti piksel. Kada bi se upotrebljavao samo taj algoritam bez drugoga (širenje rubova) kvaliteta predloška bi se gubila na način da bi mu se broj piksela smanjivao ne oviseći o promijeni njegove veličine. Da bi se takva mogućnost uklonila uvodi se širenje rubova koje uzima kao parametar koordinatu pronađenog piksela te ukoliko se on nalazi na rubu, rub se zapisuje u novi predložak. Rubom se smatra kontinuirana neprekinuta krivulja. Ipak kako objekt može biti spojen s drugim objektom ne mora se uzeti njegova cijela duljina nego samo onaj dio koji je u granici maksimalnog odstupanja od rubnih koordinata starog ekvivalentnog ruba. Takvom predlošku se još mora promijeniti veličina koja odgovara rubnim koordinatama vanjskih rubova. Taj zadatak obavlja treći algoritam. Zbog problema koji postoje i koji dovode do nestabilnosti ako se provede ovakav postupak veličina objekta nije mijenjana. Razlozi će biti navedeni u slijedećim poglavljima.

Pronalaženje ekvivalentnog piksela

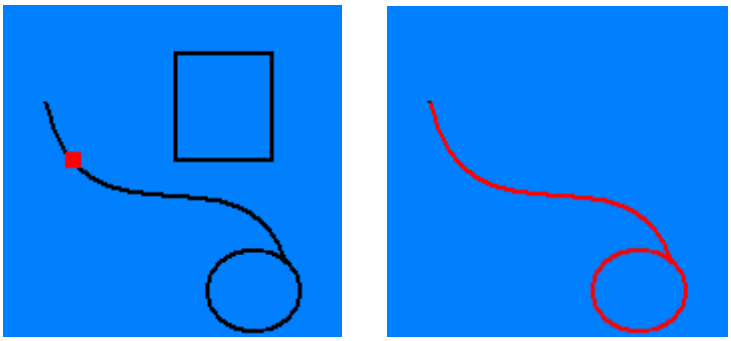
Pronalaženje ekvivalentnog piksela je rekurzivna metoda koja kao parametre uzima početnu koordinatu i radijus koji označava prostor traženja. Izlaz joj je koordinata piksela koji je prvi uspjela naći ili (-1, -1). Ovdje se ekvivalenti piksel smatra onaj koji je najbliži. Prilikom usporedbe prošloga predloška i novopronađene regije u većini slučajeva je najbliži piksel ujedno i njegov ekvivalent iako općenito ne vrijedi. Donja slika prikazuje primjer traženja ekvivalentnog piksela. Crno obojeni pravokutnik predstavlja novopronađeni ekvivalent starom predlošku, a zeleno obojeni stari predložak. Desni dio slike prikazuje rezultat rada ovog algoritma za neki prostor traženja. Preostaju samo oni pikseli koji su u radijusu pretraživanja.



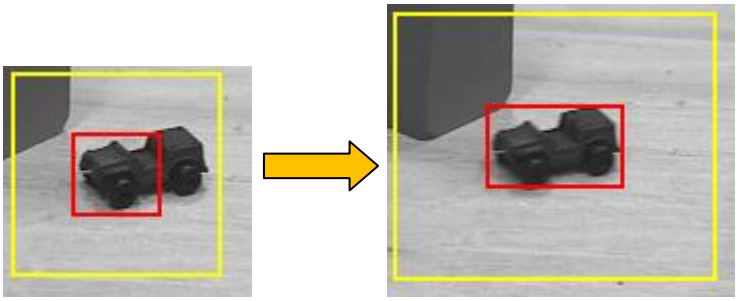
Slika 3.11 Primjer pronalaženja ekvivalentnog piksela

Proširivanje rubova pomoću piksela

Proširivanje ruba pomoću piksela je rekurzivna metoda koja kao parametar prima koordinatu piksela, izlaza nema nego upisuje dobiveni rub u sliku predanu parametrom. Algoritam nastoji za koordinatu piksela ukoliko se ona nalazi na rubu izlučiti rub koji jednu od koordinata imaš baš tu zadanu. Donja slika prikazuje primjer širenja ruba.

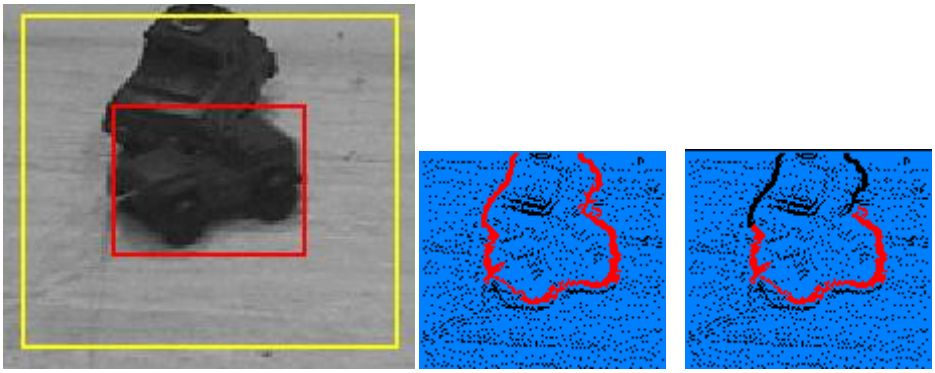


Slika 3.12 Primjer proširivanja rubova pomoću piksela (označen crveno)

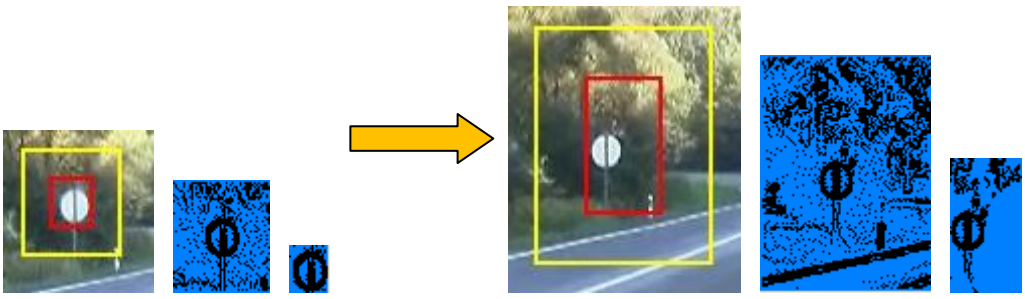


Slika 3.13 Primjer potpunog širenja ruba

Izlučivanje cijelog ruba katkada nije poželjno. Za slučaj kada su objekti spojeni nepoželjno je izlučiti cijeli rub jer bi se tada praćeni objekt proširio i na drugi objekt što nije u redu. Za smanjenje intenziteta ovog problema uzimala se u obzir duljina ekvivalentnog ruba u prošlom predlošku te duljina novog ruba je bila ograničena u odnosu na duljinu prošlog ruba. Takvo rješenje pokazalo se korisnim ako su objekti na kratko vrijeme spojeni (nekoliko uzastopnih slika videa) kako je prikazano donjom slikom. Ako bi se dogodilo da su objekti dulje vrijeme spojeni objekt bi se proširio na susjedni. Takav slučaj prikazuje slika Slika 3.14. U implementaciji priloženoj ovom radu takvo rješenje je odbačeno i uvijek se uzima maksimalna duljina ruba jer ionako predlošku ne mijenjamo veličinu.



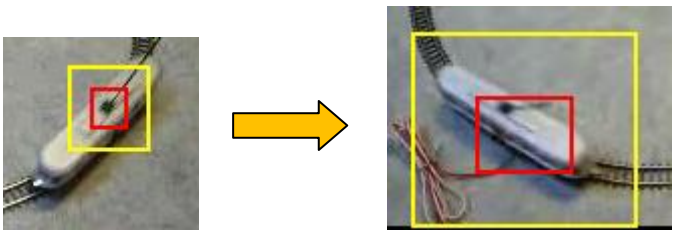
Slika 3.14 Primjer potpunog i djelomičnog širenja ruba



Slika 3.15 Akumulirana pogreška pri širenju ruba

3.3.2. Poteškoće pri praćenju objekta i konačna implementacija

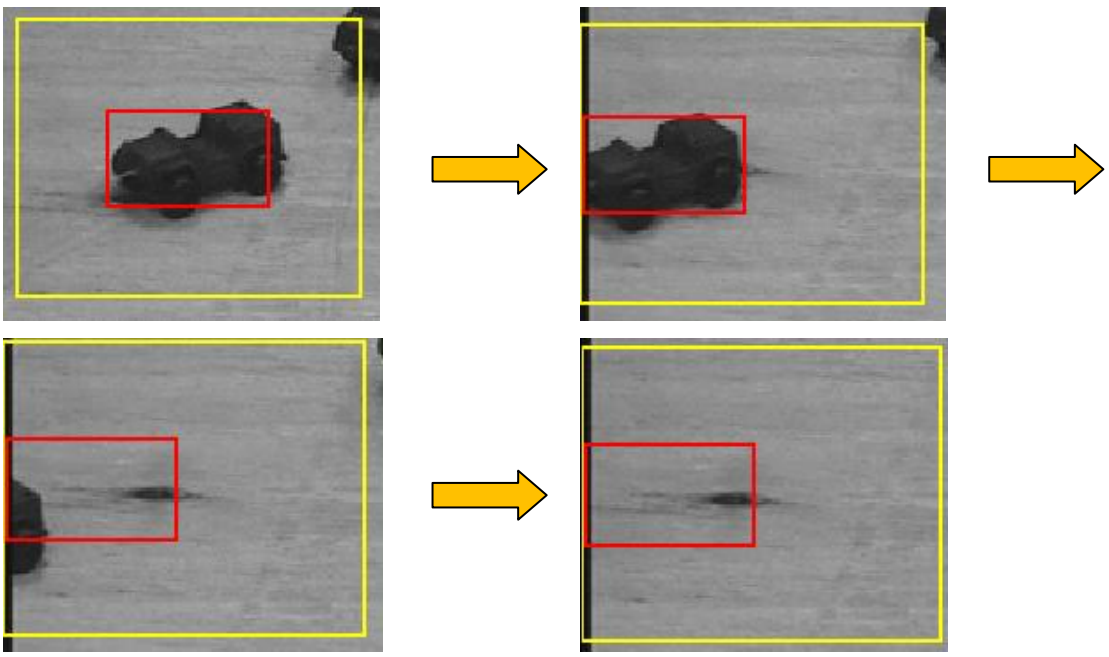
Poteškoće pri praćenju objekta očituju se u nemogućnosti ispravnog ažuriranja objekta. Kako se ekspanzijom rubova nastoji pokriti cijeli objekt velike se poteškoće javljaju ako dva objekta čine cjelinu veći dio vremena ili cijelo vrijeme. Problem je prikazan slikom Slika 3.15. Donja slika ilustrira poteškoću ako dva objekta čine cjelinu.



Slika 3.16 Akumulirana greška prilikom širenja ruba

Kako nisam našao odgovarajuće rješenje promjena veličine predloška se ne radi. Funkcije za ažuriranje veličine postoje u izvornom kodu, ali se ne rabe.

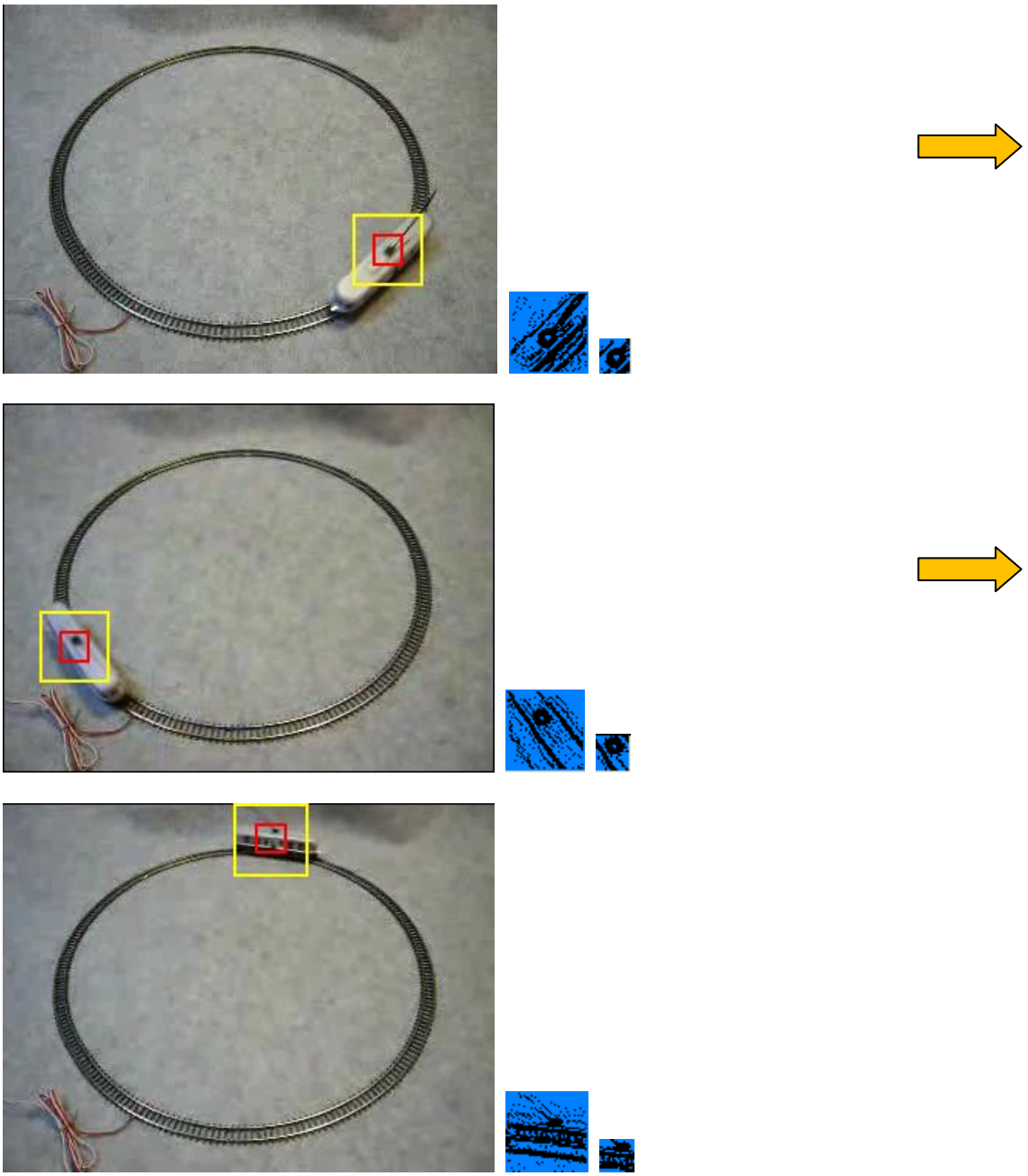
Također kako se nakon svake slike videa predložak ažurira dolazi do gubitka veze sa početnim objektom. Kako se veličina predloška ne mijenja praćenje nikada neće stati odnosno algoritam neće znati kada je praćeni objekt nestao, izašao iz kadra. Problem bi se jednostavno riješio da se ažurira veličina prozora. Naime tada bi se samo morala definirati njegova najmanja veličina. Ovaj problem demonstrira donja slika. Problem je uvijek prisutan.



Slika 3.17 Primjer problema neprestana praćenja

4. Rezultati

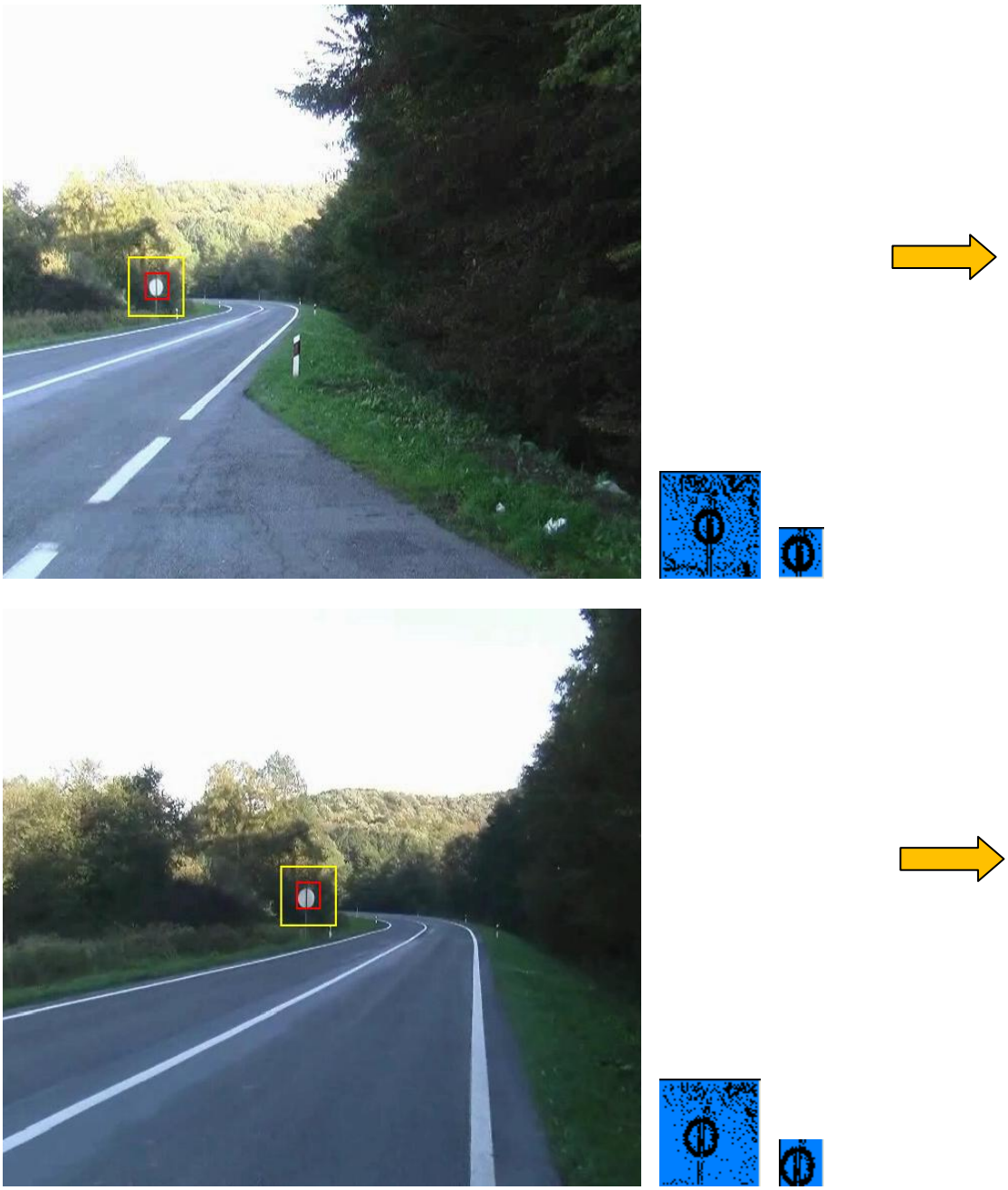
Rezultati su djelomično prikazani prošlim poglavljima. Ovdje će biti prikazati rezultati koji su dobiveni trenutnom implementacijom čije su značajke rečene prošlim poglavljem. Uz svaku sliku kakvu vidi korisnik prilikom praćenja objekta prikazana je slika područja pretrage te slika obnovljenog predloška. Također ocijenjena je uspješnost praćenja prometnih znakova.

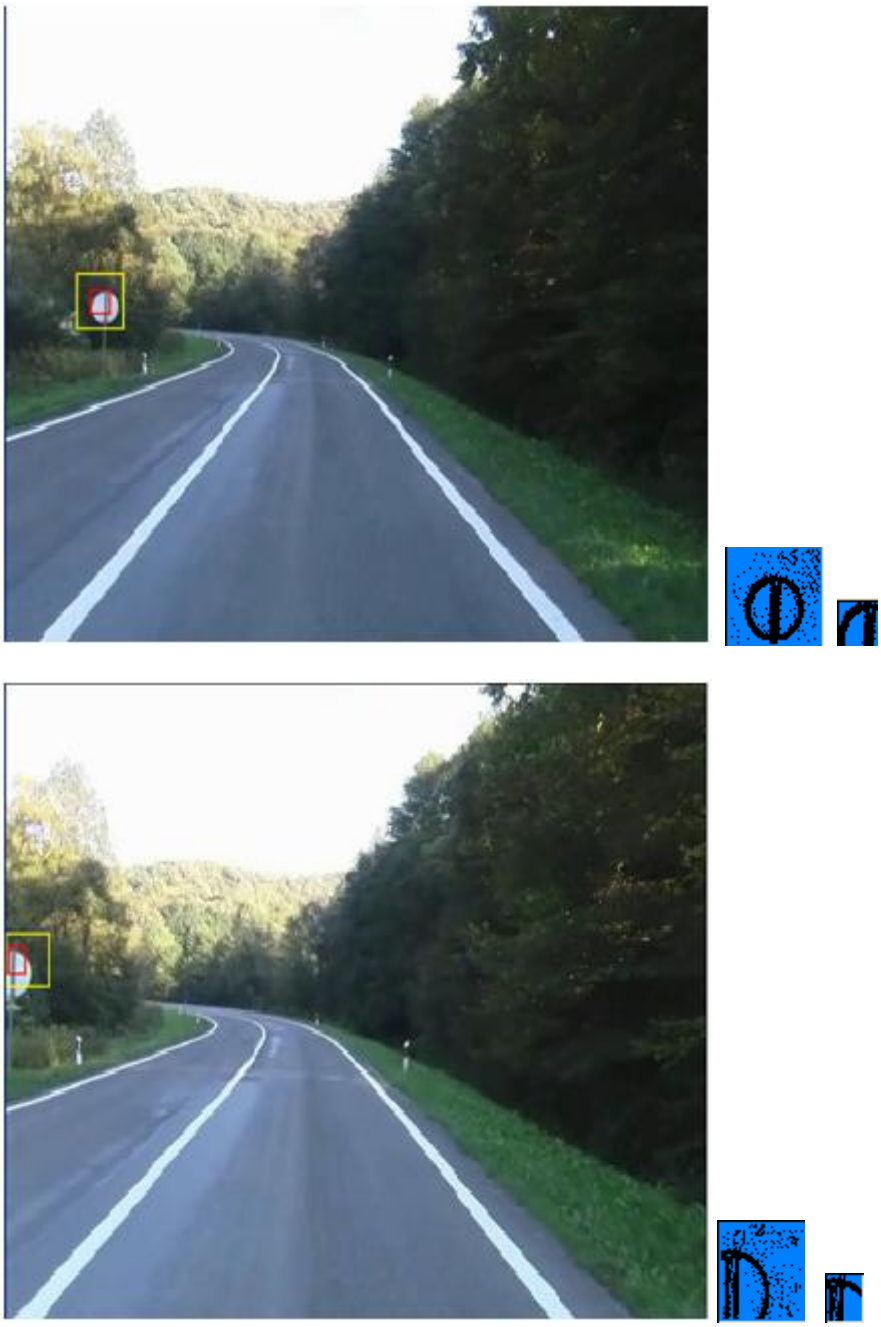


Slika 4.1 Uspješno praćenje objekta kod rotacije

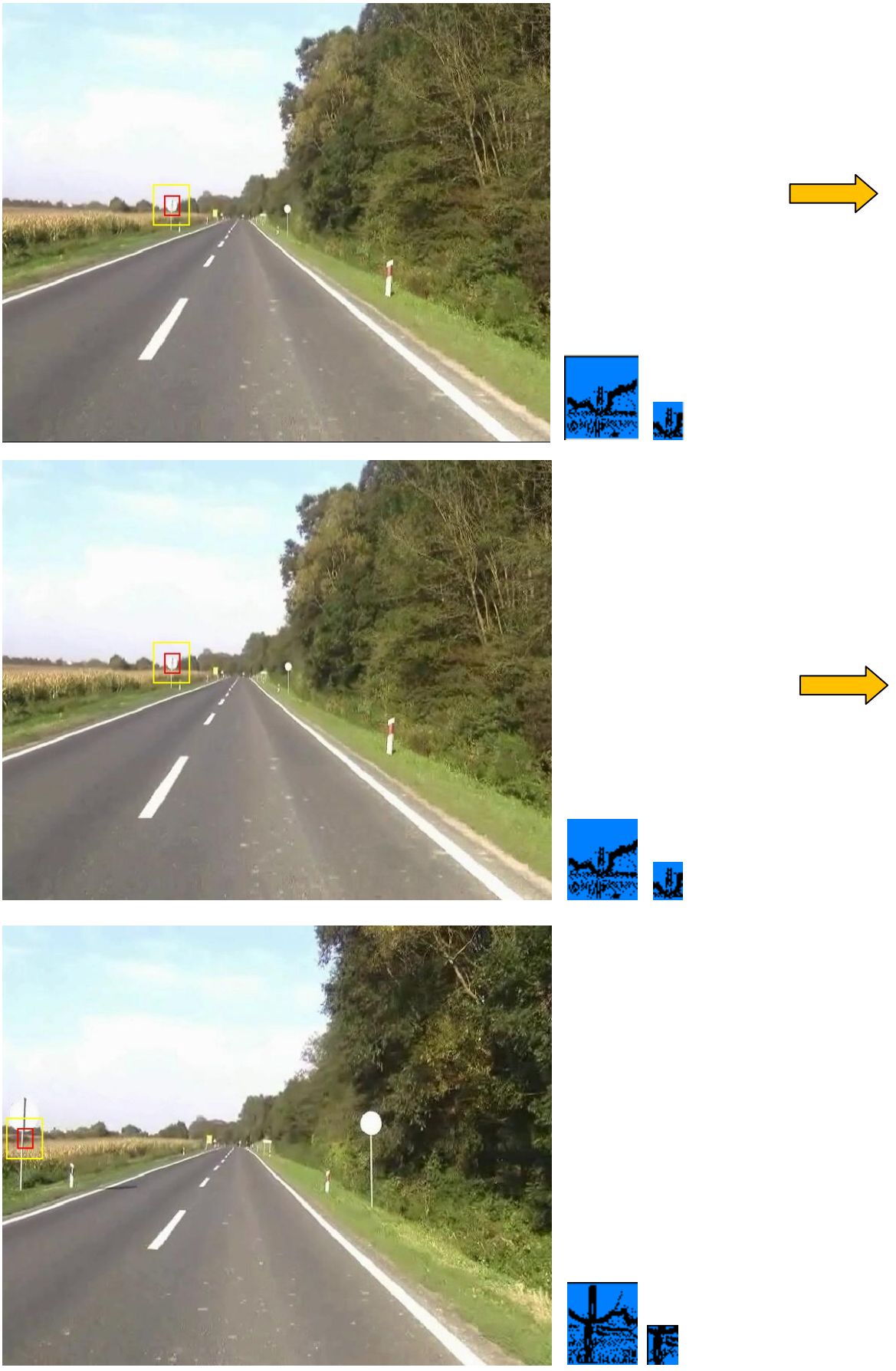
4.1. Uspješnost praćenja prometnih znakova

Ovdje su prikazani rezultati dobiveni praćenjem prometnih znakova u slikovnim sekvencama videa. Još jednom treba napomenuti da se predlošku ne ažurira veličina. Prilikom ocjene uspješnosti praćenja ova činjenica je uzeta u obzir. Za evaluaciju prometnih znakova korišten je video [12] kojeg je bilo potrebno pretvoriti u format „*avi*“.



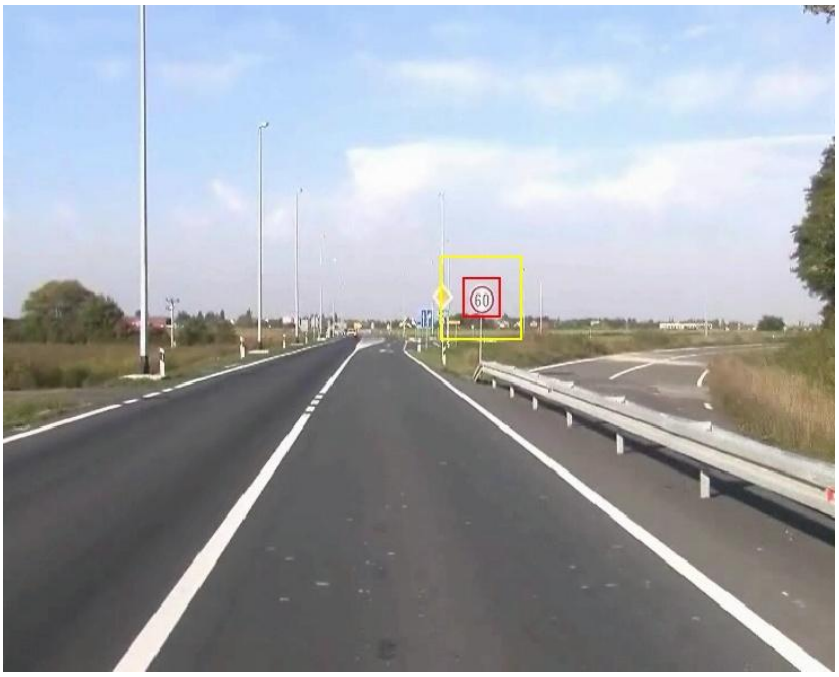


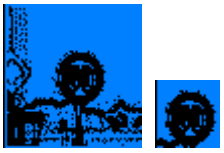
Slika 4.2 Primjer uspješnog praćenja prometnog znaka (stražnja strana).



Slika 4.3 Primjer neuspješnog praćenja prometnog znaka (stražnja strana).

Znak se nije uspio pratiti zbog nemogućnosti valjane ekstrakcije rubova koja je nastupila radi slabog kontrasta među susjednim bojama (boja neba i poleđine znaka).





Slika 4.4 Primjer uspješnog praćenog prometnog znaka (prednja strana).

Evaluacija je provedena na ukupno 17 znakova (brojeći i one okrenute stražnjom stranom) od kojih 2 nisu praćena ispravno. Svi ostali su praćeni uspješno obzirom na ograničenje konstantne veličine predloška.

Iz ovih podataka ocjenjuje se da je praćenje prometnih znakova moguće i da je provedeno uspješno.

Zaključak

Radom je objašnjen postupak za praćenje na temelju rubova. Sastoji se od nekoliko koraka:

1. Korisnički odabir objekta
2. Pretvaranje boja slike u sivu paletu
3. Uporaba ekstrakcije rubova (Sobel)
4. Uporaba binarizacije (Stucki)
5. Uporaba transformacije udaljenosti (Chamfer)
6. Obnavljanje predloška (pronalaženje ekvivalentnog piksela i širenje ruba)

Proces praćenja počinje korisničkim odabirom dijela slike koji se želi pratiti. Za samo izlučivanje rubova potrebno je bilo upotrijebiti nekoliko metoda koje su navedene u koracima 2, 3 i 4 iako se sama ekstrakcija vrši korakom 3. Većina algoritama za ekstrakciju rubova kao ulaz prima slike čiji pikseli sadrže vrijednosti sive palete, stoga je korak 2 nužan. Da bi se metoda transformacije mogla izvesti slika mora biti binarizirana. Razlozi odabira specifičnih algoritama navedeni su prilikom njihova objašnjavanja u prošlim poglavljima. Metoda transformacije udaljenosti je omogućila usporedbu i pronalazak praćenog objekta u slijedećoj slici videa. Njezino računanje izvedeno je u složenosti $O(n)$. Ipak sama pretraga je računski najzahtjevnija te zbog toga implementacija algoritma za veće objekte radi dosta sporo. Manji objekti mogu se pratiti u realnom vremenu. Kako se objekt mijenja bilo je potrebno periodički obnavljati predložak. U tu svrhu implementiran je algoritam koji prvo pronalazi piksele koji najbolje odgovaraju starim pikselima po udaljenosti, a zatim se rubovi koji sadrže te piksele prošire. Metoda je u radu detaljno objašnjena te su prikazane njezine slabosti. Najveći nedostatak ovakvog obnavljanja rubova objekta je nestabilnost koja se očituje pri promijeni veličine objekta. Zbog toga prostor traženja drži se konstantom veličinom (ona veličina koju je korisnik odabrao). Ipak metoda je u stanju vrlo dobro pratiti objekte i kod umjerenog pozadinskog šuma što je pokazano na praćenju prometnih znakova.

Darko Jurić

Literatura

[1] *Image Analysis: Image Matching Using Distance Transform*, Springer Berlin / Heidelberg, 2003, 212-229.

[2] Gavrila, D. M., *Pedestrian Detection from a Moving Vehicle*, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.122.2244&rep=rep1&type=pdf, 14.6.2010.

[3] De Smith, J. M., Distance and Path, *Centre for Advanced Spatial Analysis,* University College, London, 2004, 168-176

[4] Coeurjolly D., *Distance Transform*, http://tc18.liris.cnrs.fr/subfields/distance_skeletons/DistanceTransform.pdf, 2-30, 14.6.2010.

[5] Szeliski, R., *Computer Vision: Algorithms and Applications*, http://szeliski.org/Book/drafts/SzeliskiBook_20100517_draft.pdf, 205-246; 268-286

[6] Tsuji, H., Tokumasu, S., Takahashi, H., Nakajima, M., *Extracting Objects Using Contour Evolutions in Edge-Based Object Tracking*, Japan, 2005, http://www.fujipress.jp/finder/preview_download.php?pdf_filename=PRE_JACII001000030014.pdf&frompage=abst_page&pid=123&lang=English, 14.6.2010.

[7] Chakravarty, P., Visually *Guided Autonomous Robot Navigation*, 2003, http://users.monash.edu.au/~pcha25/Presentation_slides_AUSCC.ppt, 14.6.2010.

[8] *Sobel operator*, http://en.wikipedia.org/wiki/Sobel_operator, 14.6.2010.

[9] *Class Stucki*, http://sekal.ics.p.lodz.pl/~andrey/html/po/doc/org/plant/kzpif/filters/dithering/Stucki.html, 14.6.2010.

[10] *Floyd–Steinberg dithering*, http://en.wikipedia.org/wiki/Floyd%E2%80%93Steinberg_dithering, 14.6.2010.

[11] Chen, Z., Husz, Z., Wallace, I., Wallace, A., *Video Object Tracking Based on a Chamfer Distance Transform*, Edinburgh, UK,

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.107.2223&rep=rep1&type=pdf, 14.6.2010.

Korištene videosekvence:

[12] https://garo.zemris.fer.hr/mastif/V-24_9_2007-16_22_58-D5-A.wmv, 14.6.2010.

[13] http://www.zemris.fer.hr/projects/ObjectTracking/autici.html, 14.6.2010.

[14] http://www.zemris.fer.hr/projects/ObjectTracking/misevi.html, 14.6.2010.

[15] http://www.zemris.fer.hr/projects/ObjectTracking/setanje.html, 14.6.2010.

[16] http://www.zemris.fer.hr/projects/ObjectTracking/ruka.html, 14.6.2010.

[17] http://www.zemris.fer.hr/projects/ObjectTracking/vukovarska.html, 14.6.2010.

Sažetak

Praćenje objekata u slikovnim sekvencama na temelju informacije o rubovima

Opisan je algoritam koji se temelji na praćenju rubova objekta. Najprije se nizom metoda izlučuju rubovi koji se zatim uspoređuju i prate pomoću metode transformacije udaljenosti. Objekt se u vremenu mijenja te je informacije o njemu (rubove) potrebno ažurirati. Razvijen je algoritam za ažuriranje rubova objekta koji se temelji na pronalaženju ekvivalentnih piksela i širenju rubova. Postupak se pokazao djelomično uspješnim. Naime, moguće je praćenje objekta, ali zbog nestabilnosti pri praćenju onemogućeno je ažuriranje veličine predloška. Sve metode i postupci su detaljno opisani. Dana je i ocjena uspješnosti pri praćenju prometnih znakova.

Ključne riječi:

praćenje objekata, detekcija rubova, transformacija udaljenosti

Summary

Tracking objects in image sequences based on edge information

An algorithm for object tracking is presented which is based on tracking edges of an object. First, different methods are used to extract edges, which are then compared and tracked by method called distance transform. The objects are changing over time, and information about them (the edges) needs to be updated. A method for updating tracked objects is presented which is based on finding the equivalent pixels and spreading the edges. The procedure proved to be partially successful. It is possible to track an object, but due to instability in tracking, template size update is disabled. All used methods and procedures are described in detail. An assessment of tracking traffic signs is given.

Keywords:

object tracking, edge detection, distance transform

Skraćenice

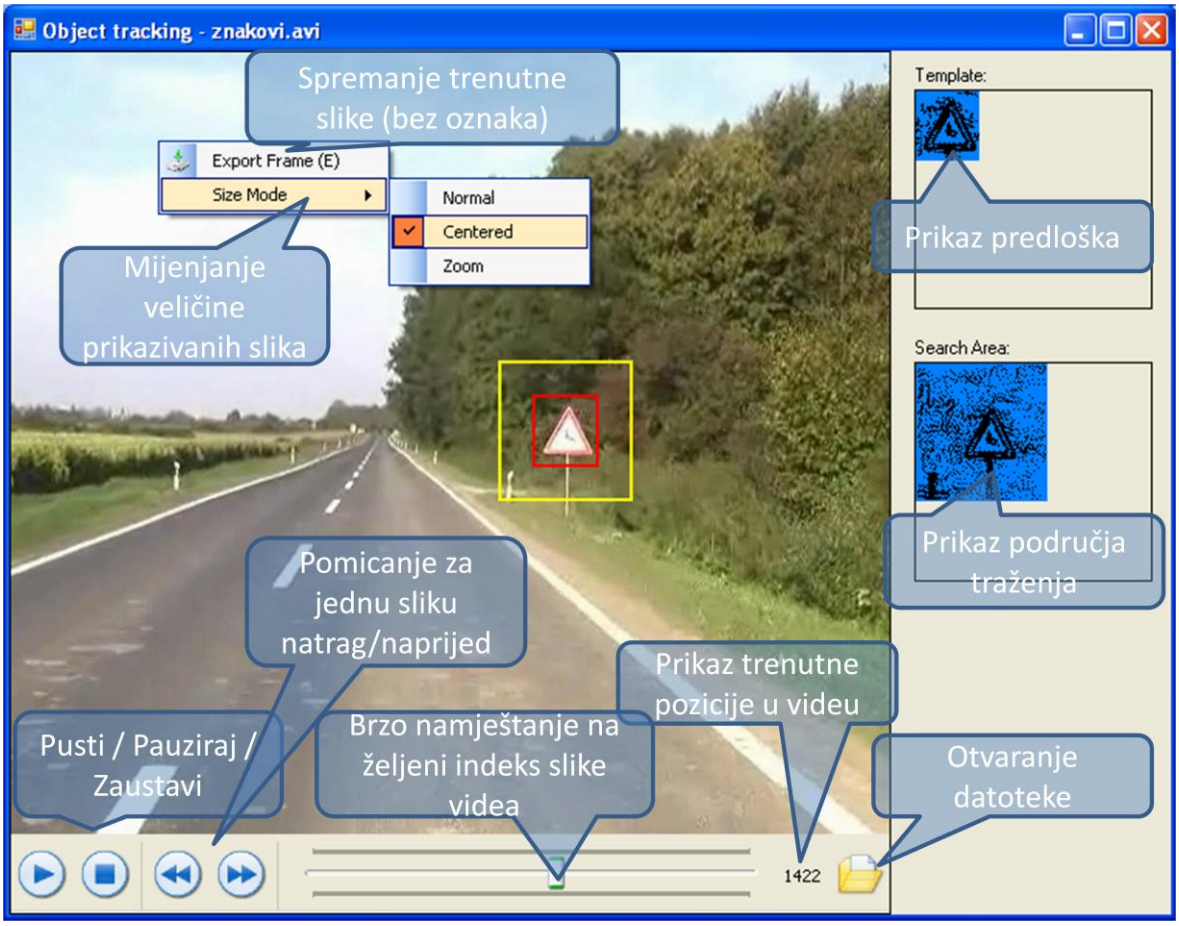
DT	*Distance Transform*	transformacija udaljenosti
AVI	*Audio Video Interleave*	format za video definiran od strane tvrtke Microsoft
EDT	*Euclidean Distance Transform*	transformacija udaljenosti provođena Euklidskom metrikom
KLT	*Kanade-Lucas-Tomasi (feature tracker)*	naziv algoritama za praćenje objekata u slikovnim sekvencama

Dodatak

Instalacija programske podrške

Program je potrebno samo pokrenuti, nije potrebna nikakva njegova instalacija. Moguća je pojava pogreške prilikom otvaranja videa formata „avi“ ako video datoteka koristi koder/dekoder (*eng. codec*) koji nije instaliran na računalu na kojem je pokrenut. U tom slučaju je potrebno instalirati dekoder čiji se naziv pojavljuje u ispisu pogreške. Kako bi se greška izbjegla obično je dovoljno instalirati neku od posljednjih verzija programa naziva „FFDShow“ (u trenutku pisanja to je 2010-05-31; datum označava i verziju) čime se automatski instaliraju neki od kodera/dekodera za „avi“ vrstu video datoteke.

Upute za korištenje programske podrške



Slika 0.1 Prikaz i objašnjenje sučelja programa

Sučelje programa je prikazano slikom Slika 0.1. Objekt se označava na način da se povlačenjem pokazivača uz pritisnutu desnu tipku prijeđe preko željenog dijela slike. Objekt se može početi označavati i u toku prikazivanja videa. U tom slučaju program će trenutno zaustaviti prikazivanje videa.

Posebne napomene

Slikovni prikaz pri pokretanju programa je namješten na “Zoom”. Pri prikazivanju nekih video datoteka mogu se pojaviti artefakti. Potrebno je samo promijeniti veličinu prozora ili promijeniti slikovni prikaz u jedan od preostala dva.