SVEUČILIŠTE U SPLITU
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE
Poslijediplomski doktorski studij Elektrotehnike i informacijske tehnologije

Kvalifikacijski doktorski ispit

Detekcija ispupčenih objekata

Željko Marušić

Split, 15.01. 2016

Tablica sadržaja

1. UVOD

Ljudski vizualni sustav je sposoban veoma brzo i praktično bez napora prosuditi i pronaći važnije dijelove scene koristeći mehanizam selektivne pažnje [1]. Teorije ljudske pažnje polaze od pretpostavke da ljudski vizualni sustav obrađuje samo dijelove slike detaljno dok druge ostavljaju praktično ne obrađenim.Nakon pronalaska vizualno karakterističnih i prepoznatljivih regija u sceni obrađuje fine detalje kako bi izdvojio informacije više razine. Iz perspektive računalom podržanog vida, detekcija upadljivih regija u sceni je veoma zahtjevna iz razloga nedovoljne istraženosti ljudskog vizualnog sustava. Uvriježen izraz na engleskom jeziku za vizualno karakteristična područja u slici jeste *saliency,* što označava vizualnu jedinstvenost, nepredvidljivost, rijetkost ili iznenađenje i često je karakterizirana varijacijama u slici kao što su boja, gradijent,rubovi i granice.

U hrvatskom jeziku engleski izraz *saliency* se može prevesti kao istaknutost, isturenost, upadljivost, ispupčenost ili prominentnost i ove izraze ću koristiti ravnopravno u daljnjem tekstu. Vizualna upadljivost (*engl. visual saliency*) je jako blisko povezana sa načinima kako shvaćamo i obrađujemo vizualne stimulanse stoga se istražuje u mnogim disciplinama koje uključuju kognitivnu psihologiju [1], neurobiologiju [2] i računalni vid [3]. Osobito je izražena veza između kognitivnih znanosti i računalnog vida u kojem se istražuju mehanizmi pronalaska objekata i regije koji efikasno predstavljaju scenu i na taj način predstavljaju probleme računalnog vida kao problem razumijevanja scene.

Najčešće se upadljivi dijelovi slike predstavljaju maskama koje izostavljaju manje važne dijelove scene odnosno mapama ispupčenosti (*engl. saliency maps*). Ove mape se potom naširoko koriste u segmentaciji objekata od interesa [4] [5], prepoznavanju objekata [6], adaptivnoj kompresiji slika [7], prikupljanju slika [8] i mijenjanju veličina slike s obzirom na sadržaj [9]. Na slici 1.1. predstavljene su četiri slike i njihove pripadajuće mape ispupčenosti.

Slika 1.1. U donjem redu su prikazane mape ispupčenosti za četiri fotografije [10]

U načelu postoji dva pristupa u detekciji ispupčenih ili istaknutih objekata: pristup odozgo prema dolje (*engl. top-down*) i pristup odozdo prema gore (*engl. bottom up*). U metodama prve kategorije poznate su nam informacije o traženim objektima prije samog procesa detekcije pa su ove metode vođene ciljem ili zadatkom. Temelje se na kognitivnim znanjem ljudskog mozga i riječ je o spontanom odnosno voluntarističkom procesu. Tradicionalne metode koje se temelje na pravilima ili metode za detekciju objekata na osnovu prethodno treniranih podataka su primjeri ovog pristupa. S druge strane, u drugu kategoriju spadaju tehnike vođene podacima ili vizualnim stimulansima. Temelje se na odzivu ljudskog vizualnog sustava na vanjske stimulanse kao što su osvijetljenost, boja, kontrast, oblik koji se upečatljivo razlikuje od okoline, neuobičajeni pokret ili drugim značajkama niske razine.

U kvalifikacijskom radu ograničiti ću se na modele koji se zasnovani na principu odozdo prema gore. U prvom dijelu rada dati ću pregled literature ovog područja, opis značajnijih modela u detekciji istaknutih objekata, te opisati načine evaluacije i mjerenja performansi takvih modela. U drugom djelu rada fokusirati ću se na programske modele paralelnog programiranja na hibridnim ili heterogenim sustavima koji se sastoje od procesora opće namjene i grafičkih procesora. U zadnjem dijelu rada primijeniti ću neke od modela detekcije ispupčenih objekata na slikama dobivenih preko bespilotnih letjelica u svrhu detekcije ljude u operacijama potrage i spašavanja. Rad ću završiti sa mogućnostima paralelizacije algoritma u detekciji objekata za potrebe pretrage i spašavanja.

2. Detekcija objekata na temelju mapa ispupčenosti

Detekcija istaknutih objekata ili segmentacija istaknutih objekata se uobičajeno interpretira kao proces koji uključuje dvije faze:

a) Detekcija najistaknutijih objekta
b) Segmentacija točnih granica tog objekta

Veoma rijetko u literaturi imamo modele koji eksplicitno razlikuju ove dvije faze. Tradicionalni radovi i predloženi modeli kao u radovima od Itti [3] i Liu-a [11] usvajaju koncept *istaknutosti* u kojima se ove faze izvode simultano.
Prva faza nije nužno ograničena ka detekciji jednog objekta. Većina postojećih modela pokušavaju segmentirati najistaknutiji objekt, iako se njihove mape predikcije/predviđanja mogu koristiti i da se pronađe nekoliko objekata u sceni.

Druga faza spada u domenu klasičnih segmentacijskih problema u računalnom vidu, ali sa određenimrazlikama. Primjerice, točnost se određuje s obzirom na najistaknutiji objekt. Općenito, istraživači se slažu da dobar model detekcije treba zadovoljiti bar tri slijedeća kriterija:

1) **Dobra detekcija**: mala vjerojatnost pogreške odnosno vjerojatnost da se istaknuta regija u slici lažno označi kao pozadina treba biti mala.
2) **Visoka rezolucija**: mapa ispupčenosti treba imati visoku ili izvornu rezoluciju kao i u originalnoj slici kako bi se točno locirali objekti i zadržale informacije o originalnoj slici.
3) **Računalna efikasnost:** budući se modeli koriste kao uvodni dio u kompleksnim procesima oni moraju biti efikasni, te biti u stanju brzo detektirati regije istaknutosti.

Za objašnjavanje modela detekcije objekata na temelju mapa uočljivosti potrebno je utvrditi sličnosti i razlike između bliskih i vezanih područja u računalnom vidu kao što su predviđanje fiksacije, segmentacija i predlošci objekata.

2.1. Predviđanje fiksacije

Modeli predviđanja fiksacije su izvorno konstruirani za razumijevanje ljudskog vizualnog sustava i predviđanja kretanja očiju [3] [12]. Detekcija istaknutih objekata i modeli predviđanja fiksacije imaju dvije fundamentalne razlike. Prvom modelu je cilj detektirati i segmentirati najistaknutiji objekt/e u cijelosti (crtajući točne siluete objekta), dok drugi model nastoji predvidjeti točke u koje ljudi gledaju (gledanjem u prirodne scene obično u trajanju od 3 do 5 sekundi).

U teoriji model koji dobro radi u području jednog problema neće dobro raditi u drugom području. Primjerice, modeli detekcije istaknutih objekata će segmentirati cijeli objekt i generirati će mnogo lažno pozitivnih primjera kada se evaluira s obzirom na ljudske fiksacije. S druge strane, modeli predviđanja fiksacije će promašiti mnogo točaka unutar istaknutog objekta što će dovesti do brojnih lažno negativnih slučajeva kada ih se evaluira sa maskama za istaknute objekte. Također, zbog prisutnosti šuma u praćenju objekata prouzročenih brzim kretnjama oka između dvije fiksacijske točke (uobičajeno je riječ o 1 stupnja što približno odgovara 30 piksela na slici) visoko precizne mape ispupčenosti na razini piksela su manje poželjne. Ponekad zbog ovog šuma zaglađivanje slike povećava performanse modela. Nasuprot tome modeli za detekciju proizvode mape u kojima jasno razlikovanje granica objekata je veoma poželjno, posebice u aplikacijama.

Međutim u praksi, modeli se mogu koristiti naizmjenično budući se generiraju slične mape ispupčenosti. Primjerice nekoliko istraživača je koristilo fiksacijske mape da detektiraju i segmentiraju uočljive proto-objekte [13] [14].

2.2. Segmentacija slike

Segmentacija slike (uključujući označavanje scene ili semantičku segmentaciju) je veoma dobro istraženo područje u računalnom vidu [15]. Cilj segmentacije je dodijeliti svakom pikselu oznaku kojom se indicira da li pripada objektu ili pozadini. Nasuprot tome, modeli za detekciju istaknutih objekata brinu samo za najuočljivije objekte i tretiraju zadatak segmentacije kao binarni problem označavanja slično kao i klasični problem „objekt-pozadina“ koji se navodi u literaturi za segmentaciju. Cilj je vidjeti da li piksel pripada najuočljivijem objektu. U praksi, moguće je prvo segmentirati cijelu scenu i potom izabrati objekt koji je najuočljiviji. Međutim ovaj pristup se nije pokazao dobrim zbog dva razloga:

1) Visoko precizni algoritam segmentacije opće namjene još uvijek ne postoji
2) Takav pristup je spor, a detekcija i segmentacija istaknutih objekata bi trebala biti brza budući je ovo često pred-obradna faza prema kompleksnijim operacijama (često detekcija nije jedini cilj!)

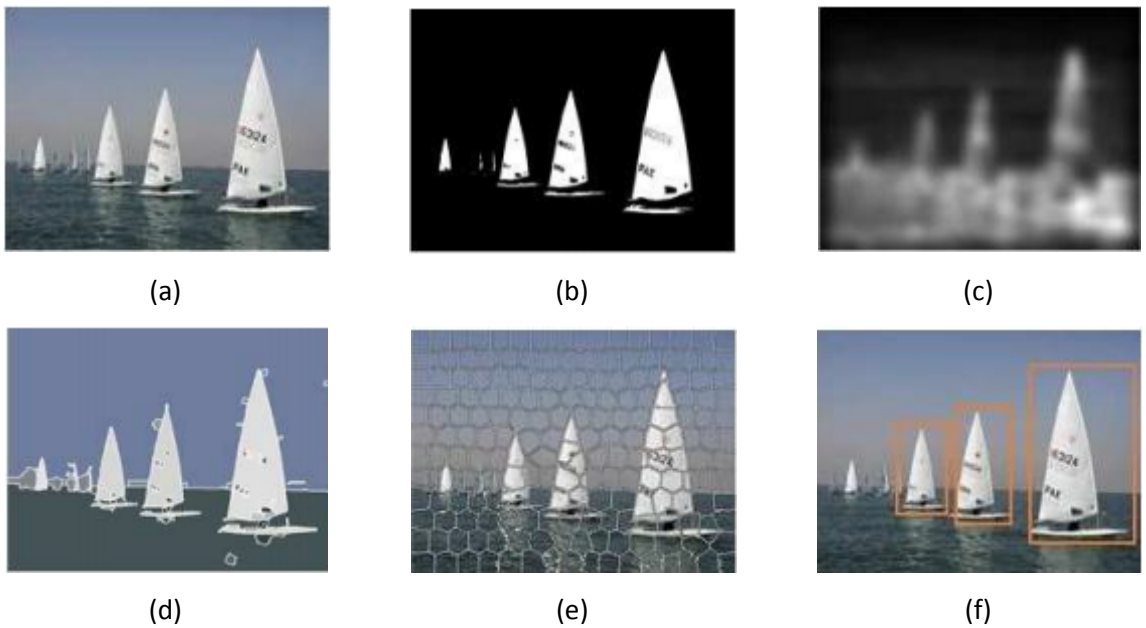
Kako bi balansirali između ova dva izazova, novi modeli za detekciju su iskoristili prednosti superpiksela koji nisu veoma precizni u segmentiranju objekata (često pre/pod-segmentiraju scenu), ali su veoma brzi u izvedbi.

2.3. Generiranje predložaka objekta

Modeli zasnovani na generiranju predloška objekta ili mjerenju objektnosti pokušavaju generirati mali skup (nekoliko stotina do tisuća) regija objekta tako da ove regije prekriju sve objekte u slici bez obzira na posebnosti i specifične kategorije tih objekata [16] [17]. Kada se uspoređuje sa tradicionalnom paradigmom klizajućih prozora [18] [19], procjene zasnovane na predlošcima objekta u fazi pred-obrade imaju tri glavne prednosti:

1) Bolje se uklapaju sa ljudskim mehanizmom prepoznavanja objekata koji brzo percipira objekte prije nego ih identificira
2) Uvelike ubrzava računanje reducirajući lokacije pretraživanja (umjesto milijuna riječ je o nekoliko tisuća lokacija) osobito kada je broj klasa objekata koji se trebaju detektirati velik
3) Unaprjeđuje preciznost detekcije dopuštajući korištenje jakih klasifikatora tijekom testiranja

Generiranje predložaka objekta i pristup detekcije istaknutih objekata su veoma čvrsto povezani. S jedne strane prvi pristup gleda na istaknutost kao korisni trag za mjerenje objektnosti regije. Drugim riječima veća je vjerojatnost da objekt bude istaknut u odnosu na regiju u pozadini. Ova saznanja se temelje na činjenici da je pozadina više strukturirana i homogenija (manje istaknuta) nego objekti. S druge strane kasniji pristup koristi mjere objektnosti da dodijeli veće vrijednosti istaknutosti objektima nego pozadini.Na slici 2.1. ilustrirana je razlika između opisanih modela bliskih području detekcije objekata na osnovu mapa ispupčenosti.



(a) (b) (c)

(d) (e) (f)

Slika 2.1. Rezultati različitih modela s obzirom na sliku a): (b) detekcija istaknutih objekata, (c) predikcija fiksacije, (d) segmentacija slike, (e) segmentacija slike putem superpiksela, (f) predlošci objekata

3. Modeli za detekciju istaknutih objekata

U posljednjoj dekadi mnogo je predloženih primjera za detekciju istaknutih tj. prominentnih ili interesantnih objekata u slici. Cilj većine pristupa je prvo identificirati vizualne pod-skupove (izračunati mapu ispupčenosti), a potom ih integrirati kako bi dobili cijele istaknute objekte. Vizualni pod-skupovi mogu biti pikseli, blokovi, superpikseli ili regije. Blokovi predstavljaju manje pravokutne segmente (*engl. patch*) koji su uniformno uzorkovani iz slike. Pikseli su blokovi dimenzije 1 x 1. Superpikseli i regije predstavljaju percepcijski homogene segmente koji su usklađeni sa intenzitetom rubova. Za istu sliku superpikseli su često usporedivih dimenzija, dok oblik i veličina pojedinih regija mogu značajno varirati.

Većina modela se može podijeliti okvirno u dvije glavne grupe, odnosno modeli koji su temeljeni na blokovima i modeli koji su temeljeni na regijama. U slijedećem dijelu detaljno su opisana značajni radovi iz svake grupe, a dat je pregled i ostali značajnih radova.

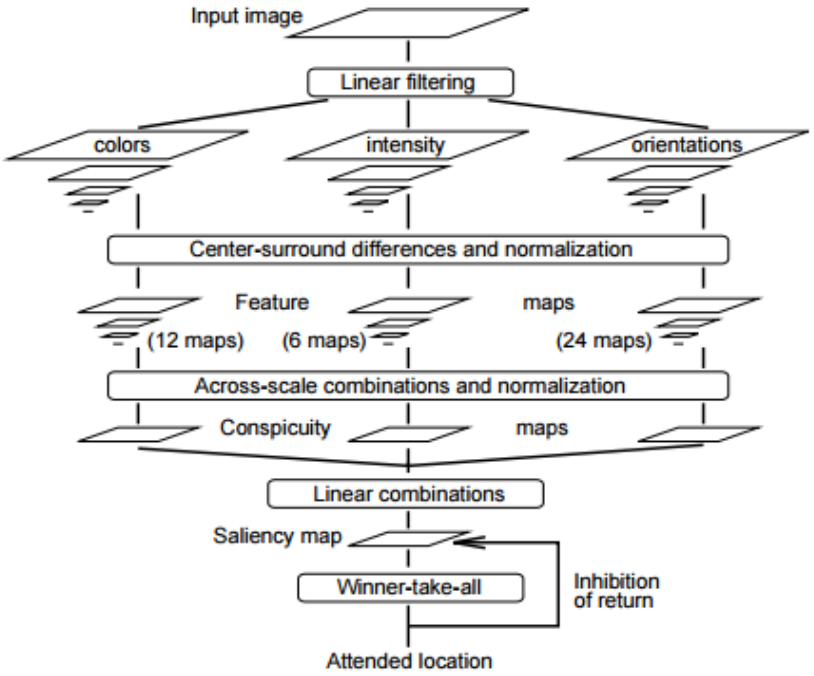
3.1. Modeli temeljeni na pikselima/blokovima

3.1.1. Model ispupčenosti temeljen na pažnji za brze analizu scene

Na temelju publikacije u [20] istražena je mogućnost dizajniranja algoritma koji reflektira ponašanje ljudskog oka i živčanog sustava u trenutku dok gleda u sliku. Pretpostavili su da selektivna vizualna pažnja funkcionira na principu kojeg zovemo *rana reprezentacija, a* koji predstavlja skup topografskih kortikalnih mapa koje kodiraju vizualno okruženje. Rana reprezentacija uključuje različite vizualne mape koje sadržavaju elementarne značajke kao što su orijentacija rubova, boja, disparitet i smjer pokreta. Arhitektura predloženog modela oponaša svojstva ranih procesa u vizualnom sustava u primata, a temelji se na tzv. teoriji integracije značajki (*engl. feature integration theory*).

Vizualni ulaz se prvo dekomponira u skup topografskih mapa značajki. Svaka mapa sadrži mnogo uočljivih objekte na različitim lokacijama, ali opstaju samo oni koje se izdvajaju u odnosu na relativno bliže okruženje odnosno oni koji su ispupčeni/izraženi. Sve mape značajki se spajaju u glavnu mapu ispupčenosti koja topografski opisuje lokalne uočljivije(*eng. conspicuity*) objekte unutar cijele vizualne scene po principu odozdo prema gore (*engl. bottom-up*). U primata se vjeruje da je takva mapa locirana u stražnjem parijetalnom korteksu.

Na slici 3prikazana je arhitektura modela. Radni okvir modela možemo gledati i kroz prizmu masivno paralelne metode brze selekcije malog broja zanimljivih lokacija u slici koje se dalje obrađuju u kompleksnijim i vremenski konzumirajućim procesima prepoznavanja objekata.



Slika 3.1. Generalna arhitektura Itti/Koch modela

Faze modela

Model se može podijeliti na više faza, a kao ulaz se koristi slika u boji.

U prvoj fazi iz jedne slike se stvara 9 slika različitih veličina upotrebom Gausove piramide koja služi kao progresivni nisko-propusni filtar i koja pod-uzorkuje ulaznu slike na način da se dimenzije slike reduciraju u horizontalnom i vertikalnom smjeru. Reducirajući faktori su u rasponu od omjera 1:1 (razina 0) do omjera 1:256 (razina 8) odnosno za 8 oktava.

Na dobivenim slikama se primjenjuju operacije srodne vizualnim receptivnim poljima. Naime, tipični vizualni neuroni su najosjetljiviji na signal u malom području tj. u središtu, dok se s udaljavanjem od središta (okruženja) odziv neurona inhibira. Takva arhitektura je osjetljiva na lokalne prostorne diskontinuitete te je osobito dobro prilagođena za detektiranje lokacija koji se ističu u svom bližem okruženju. Ovaj mehanizam predstavlja princip *centar-okruženje*. Ovaj princip se može implementirati tako da se pronađe razlika između dvaju istih slika različitih rezolucija. Slika veće,finije rezolucije se koristi kao ona koja predstavlja središte, dok slika manje, grublje rezolucije predstavlja okruženje. Kako bi simulirali princip *centar-okruženje* koriste se slike iz spomenute Gausove piramide.

Središnje slike se skaliraju sa faktorima $c \in \{2,3,4\}$, a one iz okruženja odgovaraju faktorima skaliranja $s = c + \delta$ gdje je $\delta \in \{3,4\}$. Razlika odgovarajućih piksela između dvaju mapa/slika različitih veličina se označava s operatorom $\ominus$ kojim se predstavlja operacija *centar-okruženje* (*engl. center-surround*). Budući su slike različitih veličina, manja slika se interpolira i skalira na veličinu veće te se potom izvodi piksel nasuprot piksel operacija razlike. Na ovaj način se mogu otkriti lokacije koje se ističu u svom okruženju te se princip centar-okruženje koristi za dobivanje mapa značajki intenziteta, boje i orijentacije.

Izdvajanje ranih vizualnih značajki

Nakon linearnog filtriranja pristupa se izdvajanju značajki niske razine: *intenzitet, boja i orijentacija*.Iz RGB slike se izdvajaju pojedini kanali, crveni (*r*) kanal, zeleni (*g*) i plavi (*b*) kanal, te se potom računa siva slika sa intenzitetima piksela I koristeći se formulom $I = (r + g + b)/3$. Mapa intenziteta *I* se koristi da se stvori Gausova piramida $I(\sigma)$, gdje su $\sigma \in [0 \dots 8]$ faktori skaliranja.
Kanali r,g,b se normaliziraju sa vrijednostima intenziteta I kako bi odvojili nijanse boje od intenziteta, a nakon toga se stvaraju četiri podešena kanala (crveni, zeleni, plavi i žuti kanal) preko slijedećih formula:

$$R = r - \frac{(g + b)}{2} \qquad \text{Crveni kanal}$$

$$G = g - \frac{(r + b)}{2} \qquad \text{Zeleni kanal}$$

$$B = b - \frac{(r + g)}{2} \qquad \text{Plavi kanal}$$

$$Y = \frac{(r + g)}{2} - \frac{(r - g)}{2} - b \qquad \text{Žuti kanal}$$

Negativne vrijednosti se postavljaju na nulu. Iz ovih kanala se stvaraju četiri Gausove piramide $R(\sigma), G(\sigma), B(\sigma), Y(\sigma)$ gdje su $\sigma \in [0 \dots 8]$ faktori skaliranja.

Razlike *centar-okruženje* između središta koji je predstavljen finijom veličinom slike *c* i okruženja koji je predstavljen grubljom veličinom slike *s* daje mape značajki. Prvi skup mapa značajki ističe je vezan za kontrast intenziteta koji se u sisavaca detektira neuronima osjetljivim na tamna središta na svijetloj pozadini/okruženju ili na svijetla središta na tamnoj pozadini/okruženju. Ovdje se oba tipa osjetljivosti simultano računaju (korištenjem rektifikacije) u skupu od 6 mapa gdje su $c \in \{2,3,4\}$ i $s = c + \delta,$ $\delta \in \{3,4\}$ $I(c,s) = |I(c) \ominus I(s)|$

Drugi skup mapa je vezan za boje i konstruira se za kanale boja koji su u korteksu prikazani korištenjem tzv. „dvostruko –oponentnog sustava boja“.Stvoreno je dvanaest dvostrukih oponent mapa krome između parova boja crvene-zelene $(RG(c,s))$, te oponentnih kanala parova boja plave-

žute $\big(BY(c,s)\big)$. Oni su upareni na ovakav način zbog toga što su u centru receptivnih polja neuroni pobuđeni sa jednom bojom (primjerice crvenom), a inhibirani sa drugom (primjerice zelenom), dok za okruženje vrijedi suprotno. Takva prostorna i kromatska antinomija postoji za parove crvene/zelene, zelene/crvene, plave/žute i žute/plave u ljudskom primarnom vizualnom režnju [21].

$$RG(c,s) = |(R(c) - G(c)) \ominus (G(s) - R(s))|$$

$$BY(c,s) = |(B(c) - Y(c)) \ominus (Y(s) - B(s))|$$

Lokalne informacije o orijentacijama se dobivaju iz sive slike sa vrijednostima intenziteta i korištenjem Gaborovih filtara različitih kutova. Gaborovi filtri predstavljaju produkt kosinus rešetke i 2D Gausove ovojnice te se može reći da relativno dobro aproksimiraju receptivno polje tj. odziv na orijentacije neurona u primarnom vizualnom korteksu. Slično kao i u primjerima za intenzitet i boje korištene je Gausova piramida $O(\sigma,\theta)$ gdje $\sigma \in [0 \dots 8]$, $\theta \in \{0°, 45°, 90°, 135°\}$ su preferirane orijentacije. Orijentacijske mape značajki $O(c,s,\theta)$ se kodiraju kao grupa lokalni kontrast orijentacija između središta i okruženja u slikama različitih veličina.

U konačnici dobijemo 42 mape značajki: šest za intenzitet, 12 za boju i 24 za orijentaciju.

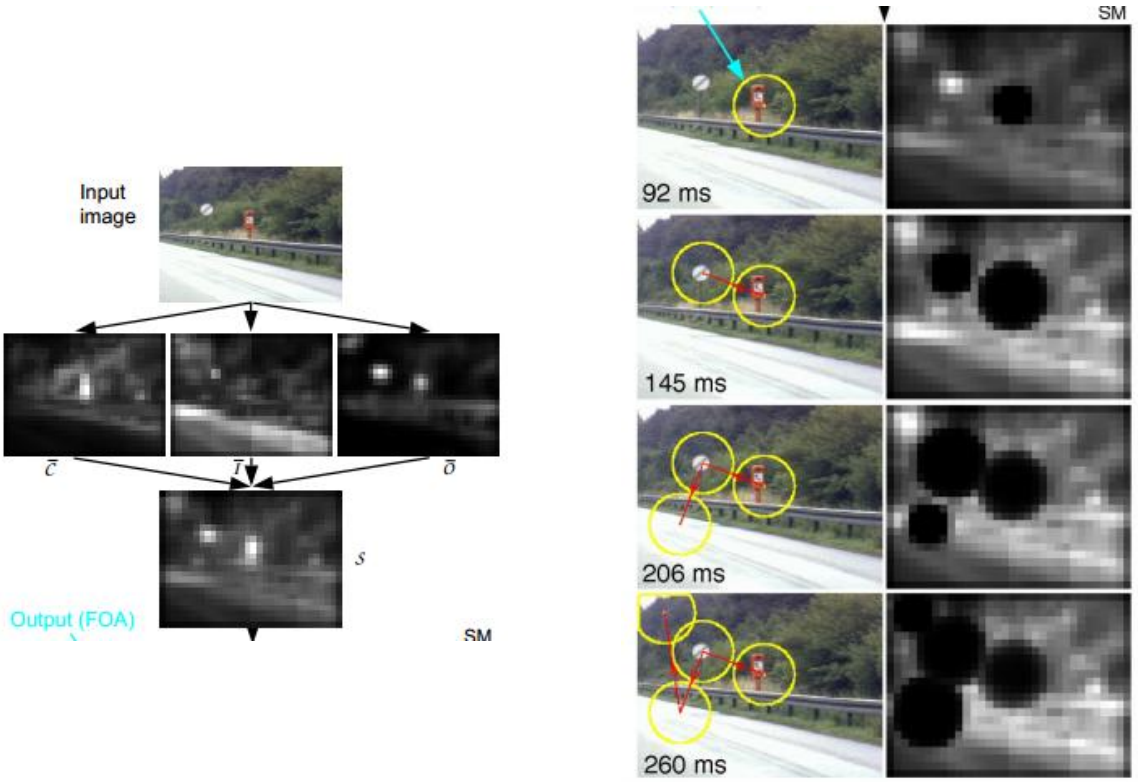
Kombiniranje mapa i normalizacija

Svrha mape ispupčenosti je predstaviti uočljivost ili istaknutostsvake lokacije vizualnog polja preko skalarne vrijednosti te vođenje procesa selekcije pristupanim lokacijama na temelju prostorne distribucije. Kombiniranjem mapa značajki po principu odozdo-prema-gore je modelirano kao dinamička neuronska mreža. Jedna od teškoća u kombiniranju različitih mapa značajki je da predstavljaju ne usporedive modalitete sa različitim dinamičkim rasponima i mehanizmima izdvajanja. Budući se sve 42 mape kombiniraju, istaknuti objekti koji su jako izraženi u samo nekoliko mapa mogu biti maskirani uslijed šuma ili zbog velikog broja manje izraženih objekata prisutnih u drugim mapama. Zbog toga se uvodi normalizacijski operator N koji promovira mape u kojima se nalazi manji broj jako izraženih vrhova aktivnosti (uočljive lokacije). Prije same normalizacije, istovrsne mape značajki se kombiniraju u tri mape uočljivosti odnosno mapa intenziteta$\bar{I}$, boje$\bar{C}$ i orijentacije$\bar{O}$na način da se mape reduciraju za faktor četiri, a potom izvrši točkasto zbrajanje vrijednosti pripadajućih piksela.

Konačna mapa istaknutosti iznosi:

$$S = \frac{N(\bar{I}) + N(\bar{C}) + N(\bar{O})}{3}$$

Algoritam identificira maksimalnu vrijednost mape istaknutosti (SM) kao centar pažnje i potom koristi neuronsku mrežu strategije „pobjednik uzima sve“ da identificira sekundarne maksimume kako bi generirali aproksimaciju prebacivanja fokusa pažnje ljudskog oka.



Slika 3.2. Primjer operacije na slikama iz prirode.

Paralelnim izdvajanjem značajki dobivamo tri mape uočljivosti za kontrast boja (slika 3.2.), kontrast intenziteta i kontrast orijentacije. Potom se kombiniraju u mapu istaknutosti (SM). Najistaknutiji objekt je telefonska govornica i to je zapravo prva posjećena lokacija. Nakon toga se odabiru drugi istaknuti objekti.

3.1.2. Detekcija ispupčenih područja podešavanjem u frekvencijskom području (IG)

Mape ispupčenosti koje se generiraju kod većine metoda imaju malu rezoluciju. Itti–eva metoda [3] proizvodi mape veličine 1/256 piksela u odnosu na izvornu slike dok metoda od Hou i Zhanga [22] daje izlazne mape fiksne veličine od 64 x 64 piksela. Neke metode ističu samo granice objekata, ali ne ističu dobro cijele ispupčene regije.

U radu*Frequency-tuned Salient Region Detection* [23] autori su postavili nekoliko zahtjeva i kriterija:

- Naglasiti najveće istaknute objekte
- Uniformno istaknuti cijele ispupčene regije

- Ustvrditi dobro-označene granice istaknutih objekata
- Odbaciti visoke frekvencije koje pripadaju teksturama, šumovima
- Rezultat treba biti mapa ispupčenosti pune rezolucije

Pristup se temelji na analizi u frekvencijskoj domeni. Neka w_{lc} bude *cut-off* vrijednost niske frekvencije, a w_{hc}*cut-off* vrijednost visoke frekvencije. Kako bi istakli velike istaknute objekte moramo uzeti u obzir vrlo niske frekvencije, pa w_{lc} mora biti nizak (prvi kriterij) što također pomaže da se uniformno, jednoliko istaknu objekti (drugi kriterij). Da bi dobro definirali granice, moraju se zadržati visoke frekvencije iz izvorne slike pa w_{hc} mora biti relativno velik (treći kriterij), ali najviše frekvencije treba zanemariti jer one predstavljaju šumove, artefakte i uzorke tekstura (četvrti kriterij). Sama mapa sadrži veći skup frekvencija pa se kombiniraju izlazi iz nekoliko propusnih filtara sa kontinuiranim širinom $[w_{lc}, w_{hc}]$ pojasa.

Za propusne filtre koristili su razliku Gausiana (*eng. difference of Gaussians*) odnosno popularni DoG filtar koji se naširoko koristi i kao detektor interesnih točaka. DoG filtar je dan sa formulom:

$$DoG(x,y) = \frac{1}{2\pi}\left[\frac{1}{\sigma_1^2}e^{\frac{-(x^2+y^2)}{2\sigma_1^2}} - \frac{1}{\sigma_2^2}e^{\frac{-(x^2+y^2)}{2\sigma_2^2}}\right] = G(x,y,\sigma_1) - G(x,y,\sigma_2) \quad (1)$$

gdje su σ_1 i σ_2 su standardne devijacije Gausiana ($\sigma_1 > \sigma_2$).

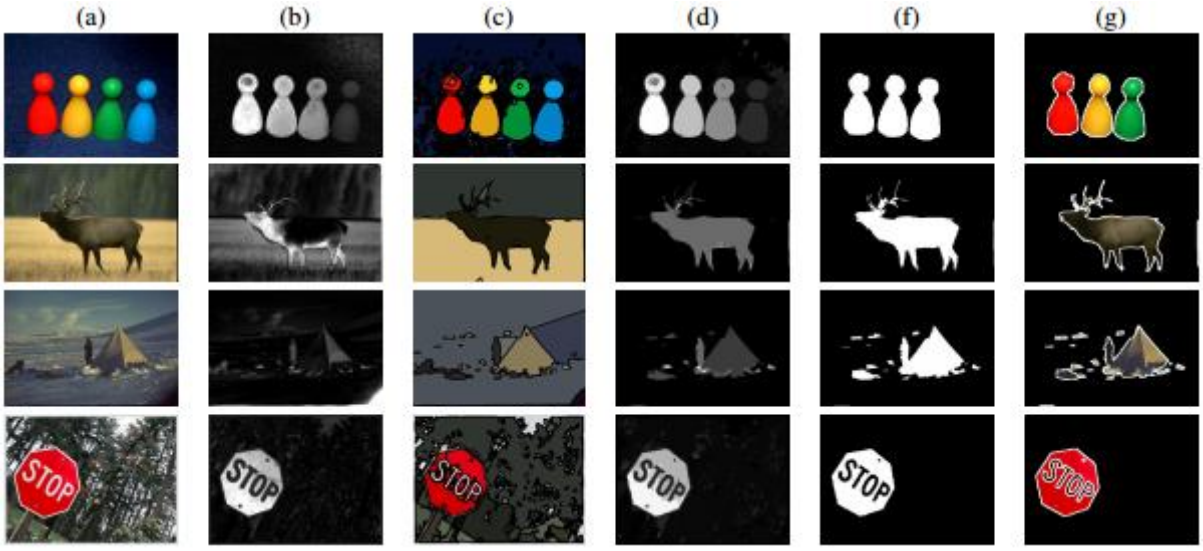
DoG filter je jednostavni propusni filtar čija se širina pojasa kontrolira omjerom $\sigma_1 : \sigma_2$. Odabir vrijednosti standardnih devijacija su ključne stavke koji će ponuditi odgovarajući propusni pojas i zadržali one frekvencije koje su nam potrebne kako bi zadovoljili navedene zahtjeve i kriterije. Za $\sigma_1 > \sigma_2$, ω_{lc} je određen sa σ_1, a ω_{hc} je određen sa σ_2. Za standardnu devijaciju σ_1 odabire se ona vrijednost koja ide u beskončanost čime efektivno zadržavamo sve frekvencije osim DC komponente, a za odstranjivanje visokih frekvencija šuma i tekstura koristi se Gausov filtar male vrijednosti.

Pronalaženje mape ispupčenost*S* za sliku *I* širine *W* i visine *H* se može formulirati kao:

$$S(x,y) = \left|\left|I_\mu - I_{\omega_{hc}}(x,y)\right|\right| \quad (2)$$

gdje I_μ je srednja aritmetička vrijednost piksela slike, a $I_{\omega_{hc}}$ je slika zaglađena Gausovim filterom. Koristi se norma razlika jer nas zanimaju samo magnitude razlika.

Mapu ispupčenosti dobivenu opisanom metodom autori su koristili u segmentaciji objekata.



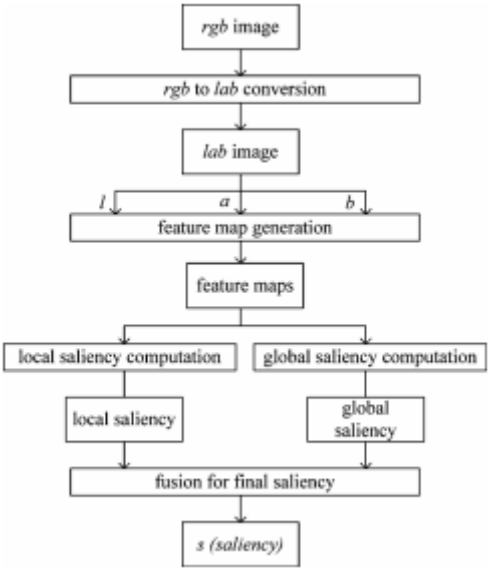
Slika 3.3. (a) su izvorne slike. (b) mapa ispupčenosti. (c) mean-shift segmentacija slike. (d)Prosječna vrijednost ispupčenosti po segmentu (f) stvarni istaknuti objektu i (f) segmentirani objekti se računa korištenjem mape ispupčenosti

Na slici 3.3. je naveden tok korištenja detektora (b) u kombinaciji sa*mean-shift (c)*segmentacijskim algoritmom. Nakon kombiniranja (d) uslijedilo je izdvajanje segmenata objekata na slikama (a).

3.1.3. Model detekcije istaknutosti korištenjem značajki niske razine koje se zasnivaju na transformaciji valićima.

U ovom radu model detekcije istaknutih objekata se dobiva korištenjem značajki niske razinetransformacijom putem valića (*engl*. *A Saliency Detection Model Using Low-Level Features Based on Wavelet Transform*) [24]. Transformacija valićima (WT) je zaokupila istraživače u modeliranju vizualne pažnje [25] [26] zbog mogućnosti pružanja višerazinske prostorne i frekvencijske analize u isto vrijeme [27]. Ovakvi modeli ovise o lokalnim i globalnim varijacijama koeficijenata valića na višestrukim razinama.Najčešće se biraju one točke/ koeficijenti koji imaju značajne globalne varijacije na većoj/grubljoj razini, a potom se ove odabrane točke prate duž finijih razina kako bi se detektirale ispupčene točke. Ispupčene točke dobivamo sumom koeficijenata valića duž lokacija točaka koje se prate kroz višerazinsku transformaciju ili dekompoziciju koristeći valiće.

Uzimajući u obzir i lokalne i globalne koeficijente valića generira se mapa ispupčenosti koja predstavlja kontrast odnosno tzv. pristup*centar-okruženje*. Dekompozicija valićimaizdvaja orijentirane detalja (horizontalne, vertikalne i dijagonalne) u više-razina što daje visoku prostornu rezoluciju sa visokim komponentama frekvencije i niskom prostornom rezolucijom sa niskim komponentamafrekvencije bez gubitka informacija o detaljima tijekom procesa dekompozicije.



Slika 3.4. Okvir predloženog modela za detekciju ispupčenosti

Na *slici 3.4.* se radni okvir predloženog modela koji integrira dvije različite mape sastavljene od lokalne i globalne mape ispupčenosti. Obe mape su sastavljene od istih mapa značajki dobivenih na temelju koeficijenata valića. Model se odvija u nekoliko faza.

Generiranje mapi značajki

Prvi korak modela počinje računanjem mapi značajki. Umjesto *RGB* prostora boja, slika se pretvara u *CIELab* prostor boja budući je isti relativno uniforman i sličan ljudskoj percepciji, a sastoji se od jednog kanala *luminantnosti* i dva kanala *krome*. Nakon toga se primjenjuje *3x3* 2D Gausov nisko propusnifiltar kako bi se smanjio šum visokih frekvencija, a potom se svaki kanal normalizira u rasponu od 0 do 255.

Korištenjem transformacije valićima za određeni broj razina preko jednadžbe (1) formiraju se pod-kanali (*engl. sub-band*). Za valiće su uzeti Daubchievi valići zbog svoje veličine filtriranja, vremena računanja i sveukupnog rezultata.

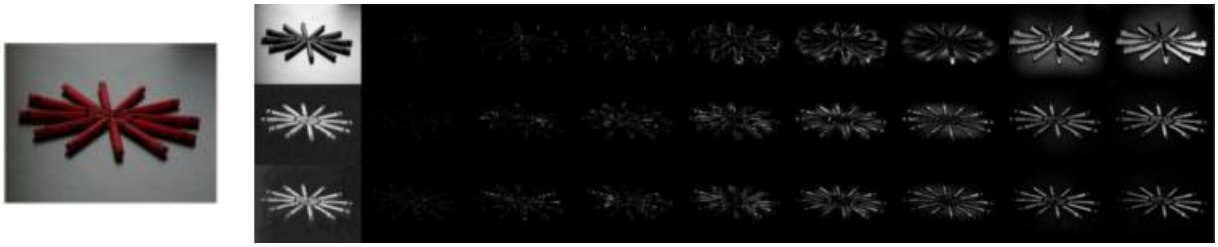
$$[A_N^c, H_s^c, V_s^c, D_s^c] = WT_N \ (g'^c) \tag{1}$$

Indeks N predstavlja broj razina u WT dekompozicijskom procesu. U rezolucijskom indeksu $s \in \{1, \dots, N\}$, N-ta razina odgovara najgrubljoj rezoluciji; *c* su kanali od g'^c (filtrirani ulazni signal) kao $c \in \{L, a, b\}$; A_N^c je aproksimacijski izlaz u najgrubljoj rezoluciji za svaki kanal; H_s^c, V_s^c, D_s^c su koeficijenti valića horizontalnih, vertikalnih i dijagonalnih valića za zadani *c* i *s*.

Koeficijenti valića *(H,V,D*) predstavljaju detalje slike u različitim veličinama, a koriste se da stvore nekoliko mapa značajki koji predstavljaju kontrast između rubova i tekstura. Mape značajki se mogu izračunati sa inverznom transformacijom valića IWT koristeći jednadžbu 2.

$$f_s^c(x,y) = \frac{(IWT_s(H_s^c, V_s^c, D_s^c))^2}{\eta} \quad (2)$$

$f_s^c(x,y)$predstavlja mapu značajki koja se generira za *s-tu* razinu dekompozicije za svaki pod-kanal *c*, a η je faktor skaliranja. Budući je raspon vrijednosti ulazne slike u *Lab* prostoru boja za svaki kanal relativno velik [0,255], koristi se faktor skaliranjaη kako bi se ograničilevrijednosti mape značajke i velike varijacije. $IWT_s(\cdot)$ predstavlja rekonstrukcijsku funkciju koja se odnosi na inverznu transformaciju valićima od $H_s^c, V_s^c \; i \; D_s^c$ zanemarujući A_N^c. Na taj način za ulaznu sliku u boji gdje je$s \in \{1, \dots, N\} i \; c \in \{L, a, b\}$korištenjem jednadžbe(2) stvara se 3xN mapa značajki, a svaka rezolucija mapa značajki je jednaka veličini ulazne slike. Na slici 3.5ilustriran je naznačeni proces formiranja značajki za svaki od *L,a, b* kanala.



Slika 3.5. Izdvajanje mapa značajki za 3 kanala slike

Za svaki kanal boje ulazne slikepostoji osam mapa značajki koje predstavljaju rezultate rekonstrukcije koeficijenata valića počevši od prve razine dekompozicije do osme razine dekompozicije (gledano sa lijeva na desno na slici 7). Rekonstrukcija osme razine se sastoji od detalja iz nagrublje razine koji se propagiraju sve do najfinije prve razine. Rekonstrukcijska razina sedam se sastoji od detalja koeficijenata valića počevši od sedme razine pasve do prve najfinije razine odnosno analogno za ostale razine.

Globalna distribucija značajki

Nakon stvaranja mapa značajki slijedeći korak je računanje globalne distribucije lokalnih značajki kako bi dobili globalnu mapu značajki. Iz funkcije$f_s^c(x,y)$ dobivenoj iz (3), svaka (x,y) lokacija se može predstaviti kao vektor značajki f(x,y) dužine 3xN (tri kanala L, a i b, svakis N razina značajki) iz svih mapa značajki.

Vjerojatnost značajke na zadanoj lokaciji se može definirati sa funkcijom gustoće vjerojatnosti (PDF) normalne distribucije [28] [29]višedimenzionalnom prostoru. Funkcija se u više-dimenzionalnom prostoru može pisati kao:

$$p\big(f(x,y)\big) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} * e^{\left(-\frac{1}{2}(f(x,y)-\mu)\right)^T \Sigma^{-1}(\mathrm{f(x,y)}-\mu)} \quad (3)$$

Sa

$$\Sigma = \mathrm{E}[(f(x,y) - \mu)(f(x,y) - \mu)^T] \quad (4)$$

gdje je μ vektor predstavlja srednju vrijednost svake mape značajki odnosno $\mu = E[f]$; T je operacija transponiranja; Σ u (4) je $n \times n$ matrica kovarijance; n=3xN predstavlja broj vektora značajki,a uključuje 3 kanala boje i N mapa značajki za svaki kanal boja;$|\Sigma|$je determinanta matrice kovarijance.

Upotrebom PDF funkcije iz (3) globalna mapa ispupčenosti se može računati jednadžbom.

$$s_G(x,y) = (\log(p(f(x,y))^{-1}))^{1/2} * I_{k\,x\,k} \quad (5)$$

Ova jednadžba sadrži informacije i lokalnog i globalnog karaktera, ali se u radu smatra globalnom mapom ispupčenosti budući je efekt globalne distribucije na mapu ispupčenosti puno izraženiji i može postati dominantan. Osim toga sadrži i statističke informacije koje se ne mogu detektirati preko lokalnih značajki.

Linearna kombinacija lokalnih značajki

Lokalna mapa ispupčenosti se stvara spajanjem ili fuzijom mapa značajki linearno na svakoj razini korištenjem formule (6). Iz ove formule može se primijetiti da se uzima u obzir maksimalna vrijednosti između pojedinih kanala ulaza slike te njihovih razina dekompozicije. Lokalna mapa ispupčenosti $s_L(x,y)$ dobivena iz mapa značajki (2) se računa kao:

$$s_L(x,y) = \left(\sum_{s=1}^{N} \arg\max(f_s^L(x,y), f_s^a(x,y), f_s^b(x,y) \right) * I_{k\,x\,k} \quad (6)$$

gdje su $f_s^L(x,y), f_s^a(x,y)\ i\ f_s^b(x,y)$ mape značajki razinesza kanale L, a i b;

Kombinacija globalnih i lokalnih mapa istaknutosti

Na temelju (5) i (6) stvara se globalna i lokalna mapa ispupčenosti. Konačna mapa je rezultat kombiniranja ovih dvaju mapa odnosno modulacije putem jednadžbe 7:

$$s'(x,y) = M\left(s_L'(x,y)\ x\ e^{s_G'(x,y)}\right) *\ I_{k\,x\,k} \quad (7)$$

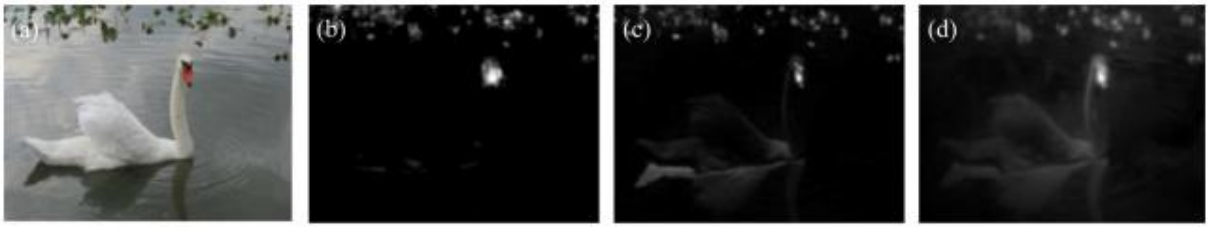
Gdje $s'(x,y)$ predstavlja konačnu mapa ispupčenosti, a$s_L'(x,y)$ i $s_G'(x,y)$ su lokalna i globalne mapa ispučenostičije vrijednosti su linearno skalirane u rasponu [0,1]. Funkcija M predstavlja nelinearnu normalizacijsku funkciju $ln\sqrt{2}/\sqrt{2}$.

Osim normalizacije, mapa ispupčenosti se poboljšava po Gestaltovom principu koji tvrdi dase lokacijama oko fokusa pažnje (FoA) mora dati više pozornosti nego onim regijama koje su dalje od fokusa pažnje [30]. Zbog toga se vrijednosti istaknutosti oko najprominentniji točaka povećavaju kako bi poboljšali performanse mapa ispupčenosti [30]:

$$s(x,y) = s'(x',y')\big(1 - d_{cFoA}(x,y)\big) \quad (8)$$

Gdje su*s(x,y)*vrijednosti ispupčenosti u točki (x,y),$s'(x',y')$ je vrijednost najistaknutijih točaka na lokaciji x',y' izdvojenih iz mape ispupčenosti u (7) koristeći prag odsijecanja od 0.8 kao;$d_{cFoA}(x,y)$ je udaljenost između lokacije (x,y) i najbližeg centra pažnje FoA na lokaciji (x',y'). Vrijednost istaknutosti oko regija ispupčenosti će se povećavati u konačnoj mapi istaknutosti; s druge strane vrijednosti istaknutosti točaka koje su udaljene regijama pod pažnjom će se umanjiti ili ostati uglavnom nepromijenjenim.

Proces kreiranja mapa ispupčenosti sa glavnim fazama prikazan je na slici 3.6.



Slika 3.6. a) RGB slika, b) globalna mapa ispupčenosti, c) lokalna mapa ispupčenosti , d) konačna mapa ispupčenosti

3.2. Modeli koje se temelje na regijama

Modeli koji se temelje na regijama često prvo segmentiraju ulaznu sliku u regije koje se podudaraju sa intenzitetima rubova, a potom računaju mapu ispupčenosti po regijama.

3.2.1. Detekcija ispupčenosti poboljšana informacijama iz regija

Članak(*engl. Region Enhanced Scale-Invariant Saliency Detection*) [31]spada u najranije radove koji su poboljšali pronalaženje mapa ispupčenosti koristeći informacije na razini većih područja ili regija. Riječ je o hibridnoj metodi koja kombinira informacije niske razine kao što je kontrast, a potom ih poboljšava informacijama dobivenih putem segmentacije slike odnosno iz regija. Naime, informacije niske razine dobivene na razini piksela ili manjih blokova piksela daju dobre rezultate u stvaranju mapa ispupčenosti, ali daju nezadovoljavajuće rezultate kada je potrebno lokalizirati

objekt. Korištenjeminformacijaiz većih područja olakšava selokalizacija objekata čime ova hibridna shema iskorištava prednosti oba pristupa.

Također autori su riješili problem ovisnosti o fiksnim veličinama faktora skaliranja u radu [3], na način da se konstruiraju mape ispupčenosti koje ne ovise o veličini slike. S ovim je osigurana invarijantnost na skaliranje.

Model se sastoji od nekoliko faza i koraka:

Korak 1*:* Transformacija slike u percepcijski uniforman prostor boja (*CIELuv*)

Korak 2: Gradi se Gausovu piramidu iz slike. Broj razina n_1 se računa iz izvorne slike (w, h) koristeći formulu $log_2(\frac{\min(w,h)}{10})$ gdje je *w* širina, a *h* visina slike

Korak 3 *:* U slijedećem koraku konstruira se piramida kontrasta za svaku skaliranu sliku, a vrijednosti kontrasta$C_{i,j,l}$ se definiraju kao težinska suma razlika između piksela(i, j) i drugih piksela u susjedstvu po formuli:

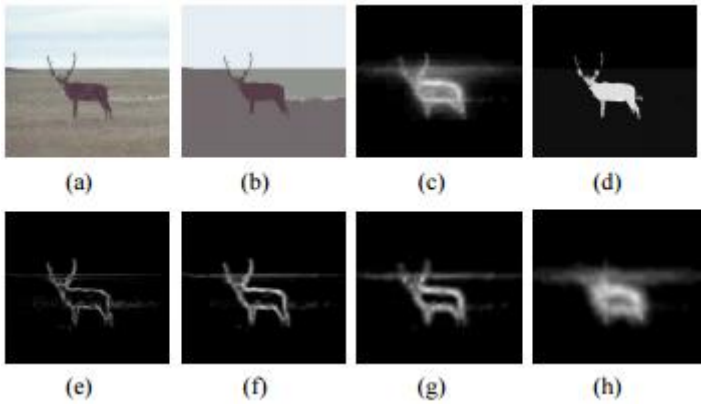
$$C_{i,j,l} = \sum_{q \in \Theta} w_{i,j,l} d(p_{i,j,l}, p_q) \tag{1}$$

Težinski faktori se definiraju kao $w_{i,j,l} = 1 - \frac{r_{i,j,l}}{r_{l,max}}$;$\Theta$ je susjedstvo piksela *(i,j)* veličine *l*;*p* je boja piksela;*q* označava da piksel pripada Θ;*d* je udaljenost između boja putemL^2 norme;$r_{i,j,l}$ je prostorna udaljenost između piksela *(i,j)* veličine *l* i središta slike;$r_{l,max}$ označava maksimalnu udaljenost od središta slike. Težinski faktori se uvode kako bi favorizirali istaknute objekte blizu središta slike budući se istaknuti objekti češće nalaze bliže središtu nego u blizini rubova.

Korak 4*:* Rekonstrukcija mape ispupčenosti iz Gausove piramide kontrasta vrši se sumiranjem svih mapa međusobno različitih veličina.

Korak 5: Segmentacija izvorne slike *mean-shift* algoritmom za potrebe izdvajanja objekata i regija odnosno lokalizacije. Računa se prosječna vrijednost istaknutosti piksela za određeni segment. Oni segmenti koji imaju najveću prosječnu vrijednost predstavljaju istaknute objekte.

Postupak poboljšanja modela ilustriran je i na slici 3.7. koja prikazuje jelena u prirodi.

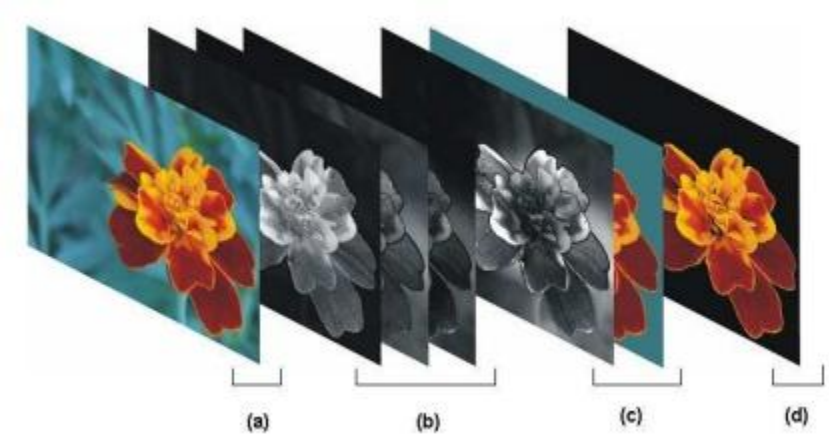


Slika 3.7. Detekcija ispupčenosti na temelju regija. a) izvorna slika. b) rezultat segmentacije. c) mapa ispupčenosti invarijantna na skaliranje. d) ispupčenost na temelju informacija iz regija. e),f),g),h) ispupčenost s obzirom na mjeru /veličinu slike 0,1,2 i 3.

Iz *slike 3.7. (e), (f), (g) i (h)* može se vidjeti da se mapa kontrasta razlikuje s obzirom na veličine slike. Naprimjer, unutarnja regija tijela jelena nije istaknuta na skali 0 i skali 1, ali je istaknuta na skali 3. S druge strane rogovi jelena se ističu na skali 0 i 1, ali se ne primjećuju na skali 3. Mapa ispupčenosti koja kombinira različite veličine/mjere se nalazi na slici 9. c). Konačna mapa je ilustrirana na slici 9d).

3.2.2. Detekcija i segmentacija istaknutih regija(AC)

Model prikazan u radu (*engl. Salient region detection and segmentation*) [32] je sličan prethodno opisanom modelu. Pregled kompletnog algoritma se može predstaviti naslici 3.8.

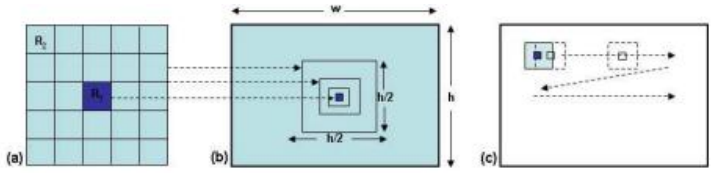


Slika 3.8. Pregled procesa pronalaženja ispupčenih regija a) Ulazna slika. b) Mape ispupčenosti u različitim veličinama. c) Konačna mapa i segmentirana slika. d) Izlazna slika

Prvo se kreiraju mape ispupčenosti različitih veličinaza ulaznu sliku a). Nakon računanja mapa ispupčenosti b) kombiniranjem na razini piksela te normalizacijomdobivamo konačnu mapu

ispupčenosti. Ulazna slika se pre-segmentira te evaluira s obzirom na mapu ispupčenosti c), na način da se biraju oni segmenti čija prosječna vrijednost ispupčenosti prelazi određeni prag što rezultira slikom d).

Ispupčenost se računa kao lokalni kontrast nekog područja u slici u odnosu na njegovo susjedno okruženje koristeći različita mjerila. Izračunava se udaljenost između prosječnog vektora značajki piksela u pod-regiji i prosječnog vektora značajki u susjedstvu (slika 3.9.).



Slika 3.9. Filter za detekciju kontrasta

Na slici 11. je prikazana unutarnja regija R1 i vanjska regija R2. Vrijednost ispupčenosti na temelju kontrasta $c_{i,j}$ za piksel na poziciji *(i,j)*se određuje kao udaljenost D između prosječnih značajki piksela u unutrašnjoj regiji R1 i u vanjskoj regiji R2 kao:

$$c_{i,j} = D[(\frac{1}{N_1}\sum_{p=1}^{N_1} v_p), (\frac{1}{N_2}\sum_{q=1}^{N_2} v_q),] \quad (1)$$

N1 i N2 predstavljaju broj piksela u R1 i R2;*v* je vektor značajki piksela. Udaljenost D je Euklidova akoznačajke u vektoru*v*nisu korelirane, odnosno Mahalanobljeva udaljenost ako su elementi vektora korelirani. U radu su koristili CIELab prostor boja kako bi generirali vektore značajki za boju i svjetlinu. Budući su percepcijske razlike u CIElab prostoru boja približno Euklidske za jednadžbu (1) *D* iznosi:

$$c_{i,j} = \|v_1 - v_2\| \quad (2)$$

Gdje su$v_1 = [L_1, a_1, b_1]^T$ i $v_2 = [L_2, a_2, b_2]^T$prosječni vektori za regije R1 i r2. Umjesto skaliranja slike, autori su se odlučili za skaliranje filtara te na taj način su dobili mapu ispupčenosti iste veličine i rezolucije kao ulazna slika. Veličina regije R1 varira od jednog piksela do prozora veličine N x N. Za sliku širine *w* piksela i visine *h* piksela, širina regije R2 varira po formuli:

$$\frac{w}{2} \geq (w_{R_2}) \geq \frac{w}{8} \quad (3)$$

Uz pretpostavku da je *w* manje nego *h*. Formula se temelji na opservaciji da veliki prozori mogu istaknuti ne-ispupčene regije, a mali prozori zapravo predstavljaju detektore rubova. Za svaku sliku filtriranje se provodi na različitim veličinama (prema jednadžbi 3), a konačna mapa ispupčenosti se određuje kao suma vrijednosti ispupčenosti po različitim veličinama S:

$$m_{i,j} = \sum_{S} c_{i,j} \quad (4)$$

$\forall i \in [1, w], j \in [1, h]$ gdje je $m_{i,j}$ element mape ispupčenosti M dobivene točkastom sumiranjem ispupčenih vrijednosti duž različitih mapa.

Ulazna slika se segmentira korištenjem *K-means* algoritma za grupiranje. Vrijednost K se traži *uz hill-climbing* algoritam.

3.2.3. SuperRare: objektno- orijentiran algoritam predviđanja pažnje temeljen na rijetkosti superpiksela

SuperRare [33] je objektno-orijentirani algoritam predviđanja pažnje koji se temelji na *rijetkosti* odnosno na rijetkim područjima na slici koji zaslužuju pažnju i pozornost. Model umjesto piksela koristi superpiksele nekoliko različitih veličina. Ovaj pristup omogućuje da *SuperRare* algoritam reagira brzo i učinkovito na istaknute objekte svih veličina. Predloženi model za značajke koristi samo boju odnosno fokus je na onim bojama koji se rjeđe pojavljuju u određenoj slici

SuperRare algoritam se može podijeliti u četiri glavne faze:

- Odabir prostora boja
- Pred-procesiranje
- Superpiksel segmentacija
- Mehanizam za istaknute objekte na temelju rijetkosti
- Fuzija ili spajanje
- Mapa ispupčenosti

Odabir prostora boja

SuperRare procjenjuje ispupčenost koristeći rijetkost boje kao osnovnu značajku. Točnost ove značajke je veoma važna stoga je ova metoda pod velikim utjecajem na izbor reprezentacije boje. Rezultati mogu varirati od loših do dobrih za istu sliku za različite prostore boja. Međutim uvijek postoji bar jedan prostor boja koji daje dobre rezultate za svaku sliku.

Pred-procesiranje

Nakon odabira prostora boja algoritam započinje sa *median* filtriranjem slike koji odstranjuje dijelove šuma, a potom se koristi *bilateralnim filterom* koji čuva glavne rubove, a istovremeno reducira šumova. Rezultat ovog pred-procesiranja je slika sa pojačanim *e*fektom crtanog filma (*engl. cartoon effect*) koji proizvodi glatkije superpiksel segmente.

Superpiksel segmentacija

Drugi korak SuperRare algoritma je reduciranje slike u manje vektore informacija korištenjem superpiksela. Superpikseli mogu zamijeniti strukturu izvorne slike koja je bazirana na pikselima sa skupom atomičnih regija koji hvataju redundantnost lokalnih informacija. Oni pružaju zgodan način za sumiranje slike te tako reduciraju kompleksnost u daljnjoj obradi i ubrzavaju algoritme. U ovom radu korištena je SLIC metoda zbog svoje jednostavnosti i performansi [34]. Superpiksel algoritmi omogućuju održavanje performansi detekcije ispupčenih objekata bez obzira na njihovu veličinu u slici. Ovo se ostvaruje dekompozicijom izvorne slike u više-dimenzija sa nekoliko skupova superpiksela različitih veličina. Ovaj pristup je inspiriran sa [35] gdje se koristi piramidalna dekompozicija za izdvajanje istaknutih elemenata različitih veličina.

Superpikseli uglađuju i eliminiraju redundantne informacije iz inicijalne slike. Ako postoji velika razlika između veličine superpiksela i veličine istaknutih objekata može se dogoditi da zbog prevelike veličine superpiksela objekti i informacije o njemu mogu iščeznuti (slika 3.9.). Kako bi se ovo izbjeglo, primjenjuje se *mehanizam rijetkosti* za različite razine superpiksel dekompozicije tako da se u nekoj od razina detalja uhvate istaknuti objekti.

a) b) c)

Slika 3.9. a) izvorna slika. b) segmentacija sa 50 superpiksela. c) segmentacija sa 100 superpiksela

Kao što je pokazano na slici 3.9. izvorna slika se segmentira korištenjem prvo 50, a potom 100 superpiksela. Može se primijetiti da u prvoj segmentaciji osoba na slici nestaje i apsorbira se u odgovarajući superpiksel. Izvodeći drugu, detaljniju dekompoziciju superpikseli ispravno segmentiraju ispupčenu osobu. Konačna mapa ispupčenosti će biti kombinacija dekompozicije svake razine kako bi se zadržale informacije o velikim kao i o malim objektima.

Mehanizam za istaknutost na temelju rijetkosti superpiksela

Mehanizam rijetkosti nastoji za svaki superpiksel *N* izračunati vjerojatnost pojavljivanja. Za svaku komponentu boje *i*, vrijednost rijetkosti se dobije normalizacijom sume vjerojatnosti pojavljivanja superpiksela što je pokazano u jednadžbi 1. P_i predstavlja vjerojatnost pojavljivanja svakog superpiksela *Spi* u odnosu na empirijsku distribuciju vjerojatnosti koja je predstavljena histogramom unutar *i-tog* kanala boja.

$$Rarity\ (Sp_i)_i = -\log(\frac{P_i}{N}) \quad (1)$$

Potom se koristi *auto-informacija* (*engl. self-information*) da se predstavi vrijednost pažnje za pojedini piksel. Ovaj mehanizam daje veće vrijednosti regijama s više kontrasta te onima koji serjeđe pojavljuju. Vrijednost parametra rijetkosti pada između 0 (svi superpikseli su isti) i 1 (jedan superpiksel različit od drugih).

Fuzija ili spajanje

Mape rijetkosti dobivene mehanizmom rijetkosti na svakom kanalu boja se prvo kombiniraju unutar razina. Fuzija između mapa rijetkosti boja se ostvaruje na svakoj razini koristeći metodu fuzije iz rada Itti [36]. Ideja je pružiti veće težinske vrijednosti onim mapama koje imaju važnije vrhove u usporedbi sa srednjom vrijednosti.

$$S = \sum_{i=1}^{N} EC_i * map_i \quad (2)$$

gdje EC_i predstavlja koeficijent efikasnosti za svaki kanal i računa se u jednadžbi 3.

$$EC_i = (max_i - mean_i)^2 \quad (3)$$

Ovi koeficijenti nam dopuštaju da se različite mape (map_i) sortiraju na temelju koeficijenata efikasnosti EC_i. Svaka mapa se potom multiplicira sa fiksnom težinom definiranom kao $i = 1, \dots, K$ gdje*K*predstavlja broj mapa koji se mogu miješati (ovdje je *K = 3*), a indeks*i* predstavlja rang sortiranih mapa kao što je prikazano u prvoj liniji jednadžbe 4. T je empirijski prag odsijecanja (*engl. threshold*) definiran u [35].

$$\forall i \in [1, K] = \begin{cases} saliency_i = 0 & if \quad \frac{EC_i}{EC_k} < T \\ saliency_i = \frac{i}{K} * map_i & if \quad \frac{EC_i}{EC_k} \geq T \end{cases} \quad (4)$$

Na kraju ovog prvog procesa fuzije model pruža jednu mapu ispupčenosti za jednu razinu superpiksel segmentacije. Kako algoritam radi bar za dvije razine segmentacije (primjerice za mapu od 100 i mapu od 200 superpiksela) , te koristi nekoliko prostora boja ovaj navedeni proces fuzije se ponavlja za sve ove razine.

Konačna mapa ispupčenosti

Za svaki prostor boja se opetovano radi fuzija kao u jednadžbi 2. Konačan rezultat uglavnom korelira sa prostorom boja koji pruža najbolji rezultat.

3.2.4. Detekcija regija na temelju globalnog kontrasta

U radu (*engl. Global Contrast based Salient Region Detection*) [37]predložena je metoda izračuna globalnog kontrasta zasnovana na histogramu (HC) kako bi mjerili ispupčenost na temelju slijedećih opservacija:

- Metode zasnovane na globalnom kontrastu se preferiraju u odnosu na lokalni kontrast iz razloga što lokalni kontrast proizvodi veće vrijednosti istaknutosti u blizini ili na samom rubu.
- Razmatranja na globalnoj razini omogućuju uspoređivanje vrijednosti istaknutosti sličnih regija u slici koje mogu uniformno istaknuti cijele objekte.
- Ispupčenost regije ovisi poglavito o kontrastu bližih regija dok kontrast prema udaljenim regijama je manje značajan.
- Generiranje mape ispupčenosti treba biti brzo i lakokako bi se omogućila obrada velike kolekcije slika i omogućile efikasnu klasifikaciju i prikupljanje slika.

Za stvaranje mapa ispupčenosti pune rezolucije HC model koristi boju kao glavni kriterij. Korištenjem metoda zasnovanih na histogramu osigurava se efikasna obrada. Osim HC mapaugrađenesu i prostorne relacije kako bi se proizvele mape zasnovane na kontrastu regija (RC) za koje se prvo segmentira ulazna slika u regije, a potom im se dodjeljuju vrijednosti ispupčenosti. Vrijednost ispupčenosti regije se računa korištenjem globalnog kontrasta koji se dobiva mjerenjem ispupčenosti regija i prostornih udaljenosti u drugim regijama.

Kontrast temeljen na histogramu

Na temelju opservacija dobivenih iz bioloških vizualnih sustava otkriveno je da je vizualni sustav osjetljiv na kontrast u vizualnom signalu. Korištenjem statistike boje ulazne slike autori predlažu kontrast koji se temelji na histogramu. Ispupčenost piksela se definira kontrastom boje u odnosu na za sve druge piksele u slici. Vrijednost ispupčenosti piksela I_k u slici *I* se definira kao:

$$S(I_k) = \sum_{\forall I_i \in I} D(I_k, I_i) \tag{1}$$

gdje je $D(I_k, I_i)$ udaljenost boje između piksela I_k i pikselaI_i u CIE*Lab* prostoru boja. Jednadžba 1 se može proširiti na razini svakog piksela na slijedeći način:

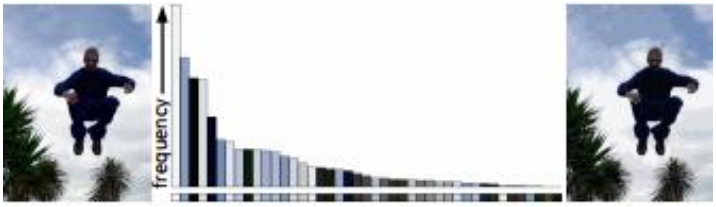
$$S(I_k) = D(I_k, I_1) + D(I_k, I_2) + \cdots + D(I_k, I_N) \tag{2}$$

gdje*N*predstavlja broj piksela u slici *I*. Pikseli iste bojeimaju istu vrijednost ispupčenosti te se jednadžba 2 može preurediti tako da su pikseli boje c_j grupirani zajedno, te time dobivamo vrijednost ispupčenosti za svaku boju kao:

$$S(I_k) = S(c_l) = \sum_{j=1}^{n} f_j D(c_l, c_j) \tag{3}$$

gdje jec_l vrijednost boje piksela I_k; n je broj različitih boja piksela, a f_j je vjerojatnost boje piksela c_j u slici *I*.

Razlog transformacije iz (1) u (3) leži u zahtjevnosti izračuna jednadžbe 1 čija je složenost $O(N^2)$ što je računalno preskupo čak i za slike srednje veličine. Ekvivalentna reprezentacija u (3) pak zahtjeva $O(N) + O(n^2)$ što implicira da se računalna efikasnost može naprijediti na $O(N)$ ako je $O(n^2) \leq O(N)$. Ključna stavka u ubrzanju je reduciranje broja mogućih boja pojedinog piksela slika. Međutim, jedan prostor boja sadrži 256^3 mogućih vrijednosti što je tipično veće od broja piksela u slici.Kako bi reducirali broj mogućih boja radi se kvantizacija svakog kanala na 12 različitih vrijednosti čime se broj mogućih boja reducira na $12^3 = 1728$. Karakteristika prirodnih slika je da su predstavljene sa relativno malim brojem boja od ukupnog prostora boja, stoga se vrši daljnja redukcija boja ignoriranjem manje frekventnijih. Odabirom učestalijih boja i zahtjevom da one prekrivaju više od 95% piksela u slikama broj vrijednosti boja se svede i do n=85 boja. Vrijednosti boje preostalih 5% piksela se zamjenjuju najbližim bojama u histogramu. Na slici 13 se nalazi primjer takve kvantizacije.



Slika 3.10. Za zadanu sliku (lijevo), računamo histogram boja (sredina). Kvantizirana slika (desno)

Na slici 3.10.rezultanti histogram boja je pokazan u donjem dijelu. Kvantizirana slika (desno) koristi samo 43 bina boja u histogramu i još uvijek zadržava dovoljnu vizualnu kvalitetu za detekciju ispupčenosti.

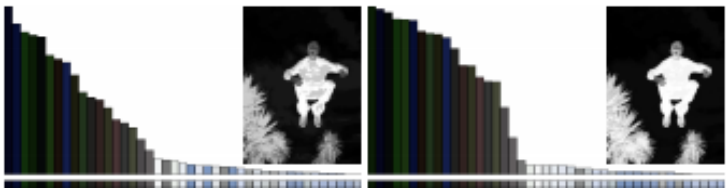
Uglađivanje prostora boja

Problemkvantizacije je što se neke slične boje mogu kvantizirati na različite vrijednosti. Kako bi reducirali šumove u rezultatima prouzrokovanih takvim slučajnostima koristimo proceduru koja rafinira vrijednosti ispupčenosti za svaku boju. Vrijednosti ispupčenosti svake boje se zamjenjuje sa

ponderiranom prosječnom vrijednosti ispupčenosti sličnih boja. Ovim postupkom vrši se uglađivanje u prostoru boja. Biramo *m=n/4* najbližih boja za rafiniranje vrijednosti ispupčenosti boje *c* koristeći

$$S'(c) = \frac{1}{(m-1)T}\sum_{i=1}^{m}\big(T - D(c, c_i)\big)S(c_i) \tag{4}$$

gdje je $T = \sum_{i=1}^{m} D(c, c_i)$suma razlika između boje *c* i njihovih *m* najbližih susjeda c_i te normalizacijskog faktora koji dolazi iz $\sum_{i=1}^{m}\big(T - D(c, c_i)\big) = (m-1)T$. Izraz *(T – D(c, ci))* predstavlja linearno-varirajući faktor kojidodjeljuje veće težinske vrijednosti onim bojama koje su bliže boji *c* u prostoru boja.



Slika 3.11.. Vrijednosti ispupčenosti svake boje prije(lijevo) i poslije (desno) zaglađivanja prostora boja, te pripadajuće mape ispupčenosti.

Kontrast po regijama uspoređivanjem histograma (RC)

Ljudi daju više pažnje područjima sa visokim kontrastom u odnosu na okruženje. Osim kontrasta važne su i prostorne relacije. Visoki kontrast u odnosu na okruženje obično daje jači dokaz za ispupčenost u regiji nego usporedivi kontrast u udaljenim regijama. Budući je računanje prostornih relacija kontrasta na razini piksela računalno skupo, uvodi se metoda analize kontrasta po regijama (RC) kako bi se integrirale prostorne relacije u izračunavanje kontrasta.

Ulazna slika se segmentira algoritmom na temelju grafova [38]. Tada se za svaku regiju gradi histogram boja koristeći HC algoritam. Ispupčenost regijer_k računamo mjerenjem kontrasta boje u odnosu na sve druge regije u slici prema

$$S(r_k) = \sum_{rk \neq ri} w(r_i) D_r(r_k, r_i) \tag{5}$$

gdje $w(r_i)$ predstavlja težinu regije r_i;$D_r(.,.)$ je metrička udaljenost boje između dvije regije. Udaljenost se ponderiras brojem piksela u r_i preko$w(r_i)$ da se naglasi kontrast boja prema velikim regijama. Razlika boja između dvije regije r_1 i r_2 je

$$D_r(r_1, r_2) = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} f\big(c_{1,i}\big) f\big(c_{2,j}\big) D(c_{1,i}, c_{2,j}) \tag{6}$$

gdje $f(c_{k,i})$predstavlja vjerojatnost *i-te* boje $c_{k,i}$ duž svih n_k boja u k-toj regiji $r_k, k = \{1,2\}$.

Uvođenjem prostornog težinskog izraza u (5) povećavaju se efekti bližih regija i smanjuju efekti daljnjih regija. Specifično, za bilo koju regiju r_k, prostorna težinska regija zasnovana na ispupčenosti kontrasta iznosi:

$$S(r_k) = w_s(r_k) \sum_{rk \neq ri} e^{\frac{D_S(rk,ri)}{-\sigma_S^2}} w(r_i) D_r(r_k, r_i) \quad (7)$$

gdje je $D_s(r_k, r_i)$ je prostorna udaljenost između regije r_k i r_i, σ_s kontrolira jačinu prostorne udaljenosti, $w(r_i)$ je težina regije r_i definirane brojem piksela r_i, i $w_s(r_k)$ je težinski prior koji je sličan središnjoj pristranosti iz [39]. Za težinski *prior* vrijedi $w_s(r_k) = \exp(-9d_k^2)$, pri čemu je d_k prosječna udaljenost između piksela u regiji r_k i središta slike sa koordinatama piksela normaliziranih na [0,1]. Stoga $w_s(r_k)$ daje veće vrijednosti ako je regija r_k bliža središtu slike, a daje manju vrijednost ako je riječ o regiji bližoj granicama slike. Veće vrijednosti σ_s reduciraju efekte ponderiranja tako da kontrast prema udaljenijim regijama doprinosi više u ispupčenosti trenutne regije.

3.3. Ostali odabrani radovi

U ovoj kategoriji kraće ću opisati ostale značajne radove koje doprinijeli razvoju ovog područja, a koji se mogu svrstati ili u jednu od prethodno opisanih kategorija ili u kategoriju radova koje koriste različite druge indikatore.

U radu [40] (engl. *Saliency, Attention and Visual Search: Saliency, attention, and visual search: An information theoretic approach) (AIM)* autori predstavljaju teoretsku formulaciju modela pažnje ljudi te na njemu grade algoritam za detekciju istaknutosti AIM (*engl. Attention based on Information Maximization*). Okosnicu algoritma čini faza izdvajanja značajki,a izvodi se upotrebom velikog broja empirijskih matrica prozvanih „naučeni filtri“. Iz pred-procesiranih prirodnih slika nasumično se proizvode blokovi piksela (*engl. patch*). Ti blokovi se koriste u ICA fazi algoritma tj. fazi neovisnog izdvajanja značajki (*engl. independent feature extraction*) radi dobivanju baznih koeficijenata. Bazni koeficijenti potom služe za opisivanje svakog bloka u slici. Algoritam je računalno kompleksan i vremenski zahtjevan, ali funkcionira u paralelnom okružju te tako simulira način na koji funkcionira mozak.

Autori rada [41] (*engl. SUN: A Bayesian Framework for Saliency Using Natural Statistics*) predlažu probabilističku formulu u računanju ispupčenosti. Algoritam se može temeljiti na značajkama

dobivenih iz razlika Gaussiana (DoG) ili značajkama ICA algoritma poznatog iz prethodno opisane AIM metode. DoG inačica ove metode počinje s odvajanjem triju komponenti boja slike (crvena, zelena i plava) iz kojih se računaju kanal intenziteta , te oponenti RG i BY kanal.

DoG filtar za (x,y) piksel je predstavljen razlikom dviju Gausovih funkcija kao:

$$DoG(x,y) = \frac{1}{\sigma^2} * \exp\left(-\frac{x^2+y^2}{\sigma^2}\right) - \frac{1}{(1.6\sigma)^2} * \exp\left(-\frac{x^2+y^2}{1.6\sigma^2}\right) \quad (1)$$

DoG filtar se konvoluira za tri kanala boja (I,RG i BY) kroz četiri σ skale (4,8,16 i 32 piksela) kako bi stvorio 12 odziva odnosno mapa značajki.

Za ovaj model nužno je mjeriti razdiobu vjerojatnosti na značajkama. Autori su odlučili iskoristiti statističke vrijednosti dobivene na kolekciji 138 prirodnih slika. Ove vrijednosti se združuju sa 12 mapa značajki kako bi se generirala procjena razdiobe vjerojatnosti svake od 12 značajki.

Procijenjena razdioba se zatim parametrizira upotrebom eksponencijalnu razdiobe:

$$p(f;\sigma,\theta) = \frac{\theta}{2\sigma\Gamma\left(\frac{1}{\theta}\right)} \exp\left(-\left|\frac{f}{\sigma}\right|^{\theta}\right) \quad (2)$$

Gdje Γ je gama funkcija, θ je parameta r oblika, σ je parametar skaliranja i f je izlaz odziva filtera.

U radu [42] (*engl. Graph-Based Visual Saliency*) pronalaženje mape ispupčenosti se zasniva na teoriji grafova. Postupak je podijeljen u tri faze: ekstrakcija i formiranje vektora značajki, generiranje aktivacijske mape na osnovu vektora značajki i normalizacije, te potom kombinacija aktivacijskih mapa u jednu mapu ispupčenosti. Dio koji se bavi izdvajanjem značajki je izostavljen, a algoritam pretpostavlja postojanje mapa značajki. I aktivacijska i normalizacijska faza koriste interpretaciju slike preko Markovljevih lanaca.

Prvi korak algoritma je dekompozicija slike u seriju mapa značajki te izvođenje preliminarne analize putem drugih metoda kako bi se reducirala rezolucija slike prebacujući prostor računanja sa piksela na regije. Potom se uvodi se funkcija različitosti kako bi se mjerile razlike između regija (piksela) mape značajki koja se definira kao:

$$d((i,j)||(p,q)) = \left|log\frac{M(i,j)}{M(p,q)}\right| \quad (3)$$

Mapa se sada pretvara u potpuno spojene usmjerene grafove G_A tako što se svaka regija tretira kao čvor nad kojim se crtaju potpuno spojeni grafovi. Težina usmjerenih bridova je definirana kao proporcionalna različitosti dva njihova kraja i njihove relativne pozicije na mapi:

$$w_1((i,j),(p,q)) = d((i,j)||(p,q)) * F(i-p, j-q) \quad (4)$$

Gdje $F(a,b) = \exp\left(-\frac{a^2+b^2}{2\sigma^2}\right)$, a σ je parametar slobode.

Sljedeći korak je definiranje Markovljevog lanca iz G_A grafa uzimajući čvorove za stanja, a težine bridova kao tranziciju vjerojatnosti. Težine izlaznih bridova svakog čvora se prethodno normaliziraju na 1. U raspodjeli ovog lanca masa se akumulira u stanjima s visokom različitosti u odnosu na okolne čvorove. Sami lanac je ergodički jer graf na kojim se temelji je čvrsto povezan, tako da distribucija ravnoteže postoji i jedinstvena je. Ova distribucija ravnoteže je osnova za izradu mape aktivacije *A*. Aktivacijska mapa A predstavlja osnovu za stvaranje mape ispupčenosti.

Autori ovo rješenje gledaju kao "organsko" budući neuroni rade samostalno, ali i utječu na jedni druge. Slično tome svako stanje/čvor se izvodi neovisno , a rezultat je stanje ravnoteže ostvareno uz pomoć ulaza iz svih stanja/čvorova.

U radu [13] (*engl. Spectral residual approach*) predložena je metoda čiji se principi temelji na spektralnoj domeni. Autori su započeli tvrdnjom iz teorije informacija da efikasno kodiranje dekomponira informacije iz slike H u dva dijela tzv. inovacijski dio i redudantni dio koji nam je otprije poznat, a koji je potrebno suspregnuti od strane sustava kodiranja. Budući se smatra da prirodne slike nisu nasumične već da se ponašaju u skladu sa distribucijama koje se mogu predvidjeti autori su otklanjajući statistički redundantne informacije u spektralnoj domeni tražili inovacijski dio. Traženu inovaciju slike su nazvali *spektralni rezidual*, a izračunali su je kao razliku između logaritmirane srednje vrijednosti Fourierove transformacije nad signalom te općeg oblika Fouriera.

Preko inverzne Fourierove transformacije spektralni rezidual se pretvara u prostornu domenu, gdje se koristi za stvaranje mape ispupčenosti. Konačna mapa ispupčenosti se dobije kvadriranjem i izglađivanjem vrijednosti.

Veoma značajan model predstavljen je u radu [43] (engl. *Visual attention detection in video sequences using spatiotemporal cues*) u kojem je razvijen model detekcije istaknutih objekata za video sekvence. U usporedbi sa detekcijom ispupčenosti u nepomičnim slikama, detekcija ispupčenosti u videu sekvencama je dosta izazovniji model zbog komplikacija u detekciji i korištenju vremenskih i prostornih informacija. Autori su izdvojeno promatrali temporalni i prostorni mode. U prostornom modelu, na osnovu statistike boje slike dobivaju se informacije o kontrastu te se konstruira hijerarhijska reprezentacija ispupčenosti na razini piksela u linearnom vremenu. Za određenu mapu ispupčenosti posjećene točke se detektiraju pronalaskom lokalnih vrijednosti maksimuma. Od tih točaka se stvara se pažnja na razini regije. Regije se iterativno šire računanjem potencijala ekspanzije s obzirom na krajnje točke tvoreći ispupčena područja.

4. Mjere evaluacije i performanse modela

U ovom dijelu ćemo objasniti standardne mjere koje su opće prihvaćene u evaluaciji modela detekcije istaknutosti te navesti performanse nekih modela na testnim bazama podataka za ovo područje.

4.1. Mjere evaluacije

Prve dvije evaluacijske mjere se temelje na preklapanju područja između subjektivne anotacije objekata od strane ljudi i predikcije objekta te uključuje *preciznost-odziv* (*engl. precision-recall PR*) i *ROC krivulju* (*engl. Receiver Operating Characteristics*). Iz ove dvije mjere se izvodi *F-mjera* koja združeno razmatra odziv i preciznost te AUS mjera koja predstavlja područje ispod ROC krivulje. Često se koristi i srednja apsolutna greška (MAE) između procijenjene mape ispupčenosti i stvarnog stanja. Radi jednostavnosti neka*S*predstavlja mapu ispupčenosti normaliziranu između [0,255], a neka G predstavlja binarnu masku ispravno detektiranih istaknutih objekata (*matrica istinitosti*). Za binarnu masku koristi se $|\cdot|$ da se predstavi broj ne-nultih zapisa u maski.

Preciznost-odziv (PR) mjera

Mapaispučenosti/istaknutosti S se prvo pretvara u binarnu masku M pomoću koje računamo preciznost i odziv (ili osjetljivost)uspoređujući M sa mapom istinitosti G.

$$Preciznost = \frac{|M \cap G|}{|M|}, \qquad Odziv = \frac{|M \cap G|}{|G|}, \tag{1}$$

Iz ove definicije možemo vidjeti da binarizirana maska Mpredstavlja ključni korak u evaluaciji.

Postoje tri popularna načina za izvođenje binarizacije [44].Jedan od načina je,o slici ovisno adaptivno odsijecanjes obzirom na prag T_α (*eng. threshold*) kojise može izračunati kao dvostruka srednja vrijednost mape istaknutosti od S:

$$T_\alpha = \frac{2}{W\ x\ H} \sum_{x=1}^{W} \sum_{y=1}^{H} S(x, y) \tag{2}$$

gdje su *W*i*H* širina i visina mape istaknutosti S.

Drugi način binarizacije mape ispupčenosti S je korištenje iterativne metode fiksnog odsijecanja praga koji se mijenja od 0 do 255. Pri svakom odsijecanju računa se par (preciznost, odziv), te se ovi parovi kombinirajuu (preciznost-odziv) krivulju kako bi opisali performanse modela u različitim situacijama.

Treći način da izvedemo binarizaciju je korištenje GrabCut [45] algoritma. U ovakvoj situaciji prvo se izračuna PR krivulja,a potom se bira ona vrijednost algoritma za odsijecanje praga koji daje 95 % odziva. S ovom vrijednosti se dobije inicijalna binarna maska koja se može koristiti za iterativnu GrabCut segmentaciju.

F-mjera

Često niti preciznost niti odziv nemogu sveobuhvatno evaluirati kvalitetu mape istaknutosti. F-mjera se računa na način da se traži ponderirana harmonijska srednja vrijednost preciznosti i odziva i ne-negativne težine β^2.

$$F_\beta = \frac{(1+\beta^2)preciznost\ x\ recall}{\beta^2 preciznost + recall} \tag{3}$$

Najčešće se β^2 težina postavlja na 0.3 kako bi se istaknula važnost parametra preciznosti budući je važniji od odziva. Primjerice odziv 100% se može dobiti ako postavimo cijelu regiju kao pozadinu.
S obzirom na nekoliko načine za binariziranje mape istaknutosti postoji dva načina za računanje F mjere. Kad koristimo adaptivno odsijecanje praga ili GrabCut algoritam može se generirati jedan F_β za svaku sliku, a konačna F-mjera se računa kao prosjek odF_β. Ako se generira jedinstvena PR krivulja na svim testnim slikama, računa seF_β za svaki par preciznost-odziv, a potom prosjek. F mjera je ograničena na vrijednosti od 0 do 1.

ROC krivulja

Kada binariziramo mapu istaknutosti koristeći se drugom metodom odnosno iterativnom metodom fiksnog odsijecanja praga može se prikupiti stopa lažno pozitivnih (FPR) i stopa točno pozitivnih (TPR) primjera koristeći se formulom:

$$TPR = \frac{|M \cap G|}{|G|}, \quad FPR = \frac{|M \cap G|}{|M \cap G| + |\bar{M} \cap \bar{G}|} \tag{4}$$

gdje $\bar{M}\ i\ \bar{G}$ opisuju inverz binarne maske M i mape istinitosti G. ROC krivulja predstavlja crtež relacijeTPR-a nasuprot FPR-a dobivenog testiranjem svih mogućih operacija odsijecanja praga.

Područje ispod ROC krivulje (AUC)

Za razliku od ROC krivulje koja predstavlja dvodimenzionalnu reprezentaciju performansi modela, AUC destilira ovu informaciju u jedan skalar. Kao što ime implicira, računa se kao površina ispod ROC krivulje. Savršen model daje AUC vrijednost 1, dok nasumično pogađanje daje AUV vrijednost 0.5.

Srednja apsolutna vrijednost greške (MAE)

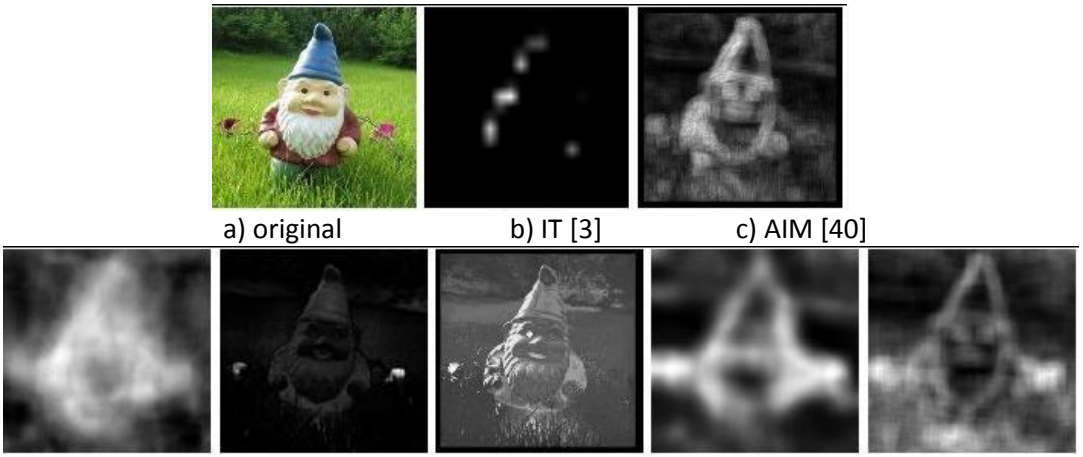
Evaluacijske mjere zasnovane na preklapanju ne razmatraju točne negativne dodjele tj. piksele točno označene kao ne-istaknute. Na taj način favoriziraju metode koje uspješno daju veliku vjerojatnost za istaknute piksele, ali ne uspijevaju detektirati regije koje nisu istaknute. Zbog obuhvatnijeg uspoređivanja preporuka je da se evaluira srednja apsolutna greška (MAE) između kontinuirane mape istaknutosti S i mape istinitosti G, obje normalizirane u rasponu [0,1]. MAE vrijednost se definira kao:

$$MAE = \frac{1}{W\ x\ H}\sum_{x=1}^{W}\sum_{y=1}^{H}\left|\left|S(x,y) - G(x,y)\right|\right| \tag{5}$$

4.2. Performanse pojedinih modela

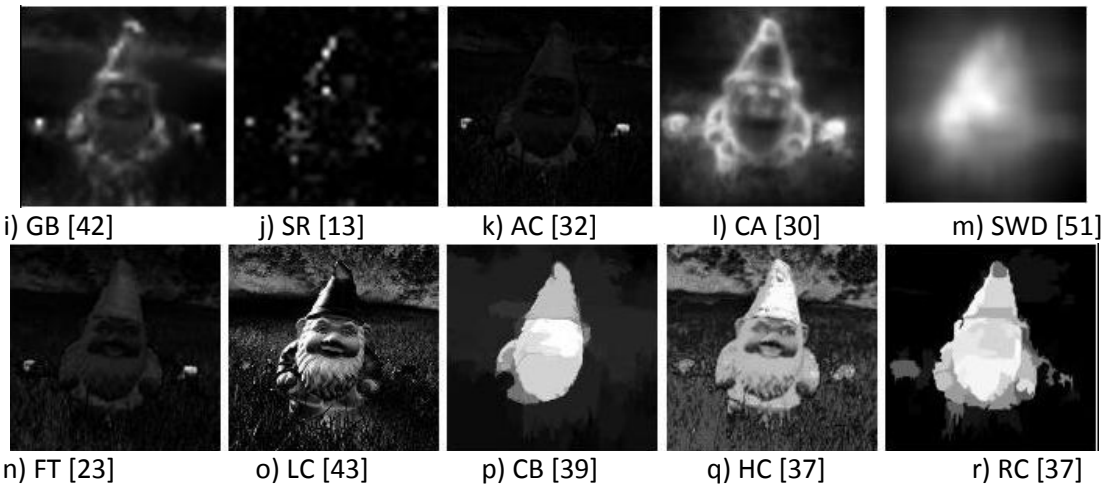
U radu [37] iz 2014. godine nalazi se dosta dobra evaluacija različitih modela pa ću u ovom dijelu dati prikaz performansi nekih značajnih modela. Testiranje je izvršeno na dvije javno dostupne baze slika THURSK [46] i MSRA10k. Baza THURSK sadrži 3000 slike koje su preuzete sa Flickr servisa. Istaknute regije su markirane na razini točnosti piksela. Baze ne sadrže sve slike u kojima su istaknute oznake regija budući svaka slika ne sadrži istaknuti objekt. Baza MSRA10k sadrži 10000 fotografija u kojoj je izvršena anotacija svih piksela. Baza je nastala je iz ranije MSRA baze [11]. Tipična veličina slika iznosi 400x300 piksela.

Na slici 4.1. a) je prikazan primjer fotografije za koju se pronalaze mape ispupčenosti različitih modela detekcije istaknutih objekata. Nazivi skraćenica modela ispupčenosti se nalaze u prilogu.



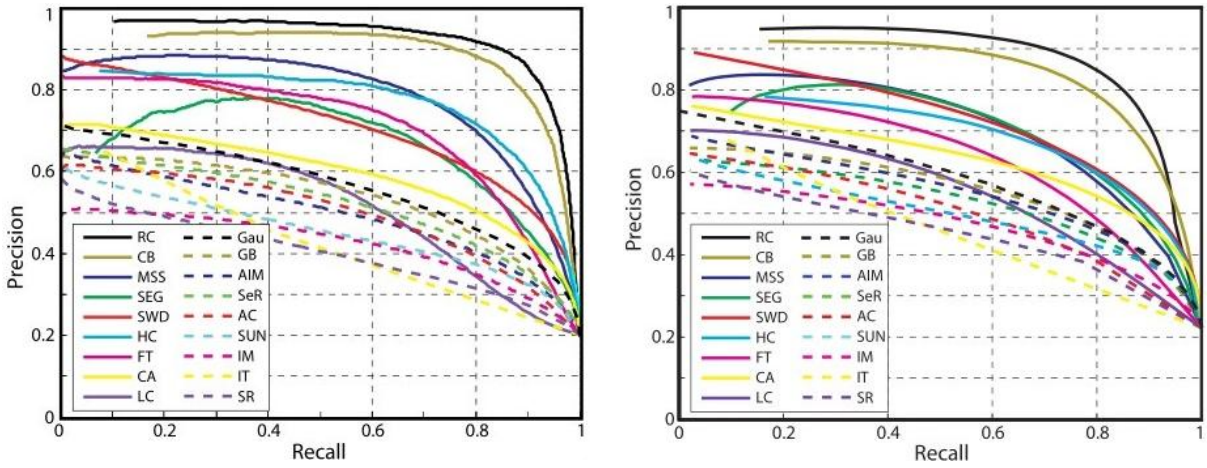
a) original b) IT [3] c) AIM [40]

d) IM [47] e) MSS [48] f) SEG [49] g) SeR [50] h) SUN [41]



Slika 4.1. Mape ispupčenosti za 17 različitih modela detekcije istaknutosti

U slijedećoj slici 4.2. nalaze se krivulja *preciznost-odziv* za dvije navedene baze slika, a u tablici 1 se nalazi vrijeme izvođenja testiranih modela za bazu MSRA10k.



Slika 4.2. Statistički usporedni rezultati metoda za detekciju objekata MSRA10k (desno) i THURSK (lijevo)

Kada se promatra krivulja preciznost-odziva na slici 4.2. jasno se pokazuju da RC metoda nadmašuje sve druge metode u eksperimentu, a jedini konkurentan model jeste CB međutim on je veoma računalno zahtjevan što se vidi iz tablice 1.

Tablica 1. Prosječno vrijeme izvođenja modela

Metoda	IT Matlab	AIM Matlab	IM Matlab	MSS Matlab	SEG Matlab	SeR Matlab	SUN Matlab	SWD Matlab	CB M & C
Vrijeme(s)	0.611	4.288	0.991	0.106	4.921	1.019	1.116	0.100	5.568
Metoda	GB Matlab	SR Matlab	FT C++	AC Matlab	CA Matlab	LC C++	HC C++	RC C++	
Vrijeme(s)	1.614	0.064	0.102	0.109	53.1	0.018	0.019	0.254	

Vrijeme izvođenja 17 modela prikazano je u tablici 1 za slike MSRA10K . Autori su koristili paralelno okruženje kako bi Matlab implementacije učinili efikasnijim na Dual Core 2.6 GHz stroju sa 2GB RAM. Model LC je se pokazao najbržim (oko 0.009 sekundi), a prati ga HC i GC model. Model HC osim što je veoma brz pokazao je i odlične rezultate kada je riječ o preciznosti.

5. Paralelno programiranje za hibridne sustave

Za neke računalne probleme, algoritmi koji su temeljeni za CPU nisu dovoljno brzi kako bi dali rješenje u razumnom vremenu. Nadalje, ovi problemi mogu postati još veći do te mjere da ni algoritmi prilagođeni za više-jezgrene CPU nisu dovoljno brzi.

Primjeri ovakvih problema se mogu naći u prirodnim znanostima [52] [53] [54] (fizika, biologija, kemija), informacijskoj tehnologiji [55] (IT), geoprostorniminformacijskim sustavima [56] (GIS), strukturnim problemima mehanike , pa čak i apstraktnim matematičkim/informatičkim (CS) problemima [57] [58] [59] [60]. Danas se mnogi od tih problema mogu riješiti brže i učinkovitije pomoću masivno paralelnih procesora.

Priča o nastanku masivno paralelnih procesora je unikatna jer kombinira dva područja koja nisu jako međusobno povezana, a riječ je računalnoj znanosti i industriji videoigara. U znanosti postoji stalna potreba za rješavanje najvećih i najtežih problema u razumnoj količini vremena. Ova potreba je dovela do izgradnje masivnih paralelnih super-računala za razumijevanje fenomena poput formiranja galaksija, molekularne dinamike kao i klimatskih promjena. S druge strane, industrija video-igara je u stalnoj potrebi za postizanjem foto-realistične grafike u realnom vremenu gdje je glavno ograničenje izvršavanje algoritama na prosječnom računalnom sklopovlju. Potreba za realističnim video-igrama je dovela do izuma grafičkog akceleratora, malog paralelnog procesora koji rukuje mnogim grafičkim izračunima upotrebom hardverski implementiranih funkcija. Iz ove dvije potrebe izrodio se jedan od najvažnijih hardvera za paralelno računanje: GPU.

GPU (grafička procesorska jedinica) predstavlja vodeći eksponent paralelnog računarstva. Fizički je relativno mala i može stati u osobno računalo, a s druge strane masivno je paralelna kao minijaturno super-računalo sposobno za upravljanje na tisuće izvršnih tokova paralelno. Prilagodbom nekih algoritama za izvođenje na GPU postignuta su značajna ubrzanja u odnosu na CPU algoritme čak i za dva reda veličine.

Neki problemi nemaju paralelno rješenje [61]. Na primjer, aproksimacija$\sqrt{x}$ pomoću Newton-Rhapson načina [62] ne može se paralelizirati jer svaka iteracija ovisi o vrijednosti prethodne iteracije, odnosno postoji pitanje vremenske ovisnosti. Takvi problemi nemaju koristi od paralelizacije i najbolje se mogu riješiti korištenjem CPU-a. S druge strane, postoje problemi koji se mogu prirodno podijeliti u mnogo nezavisnih pod-problema. Eklatantni primjer je množenje matrica koji se može podijeliti u mnogo neovisnih pod-problema. Tako se množenje matrica može podijeliti u nekoliko neovisnih računanja množenje i zbrajanja. Ti problemi su masivno paralelni i oni su vrlo česti u

računalnoj fizici, a najbolje mogu riješiti pomoću GPU. U nekim slučajevima, ovi problemi se mogu tako dobro paralelizirati da su dobili naziv *sramotno paralelni* problemi.

Najvažniji aspekt paralelnog računarstva je povezanost sa postojećim hardverom i programskim modelima. Tipična pitanja u ovom području su: Koji je tip problema s kojim se susrećem? Da li bi trebao koristiti CPU ili GPU? Da li je riječ o SIMD ili MIMD arhitekturi? Radi li se o distribuiranom ili dijeljenom memorijskom sustavu? Što koristiti: OpenMP, MPI, CUDA ili OpenCL?. Zašto performanse nisu onakve kakve se mogu očekivati? Kako se može dizajnirati paralelni algoritam?. Ova pitanja su uistinu važna kada se traži rješenje visokih performansi, a odgovor leži u području algoritama, računalne arhitekture, računalnih modela i programskih modela. GPU računarstvo također donosi dodatne izazove kao što je ručno korištenje keš memorije, obrasci za paralelni pristup memoriji, komunikacija, mapiranje niti i sinkronizacija. Ovi izazovi su kritični kod implementacije efikasnih GPU algoritama.

5.1. Osnovni koncepti

Izraz konkurentnost i paralelizam se često debatiraju u znanstvenoj zajednici i ponekad može biti nejasno koja je razlika između ova dva koncepta što može dovesti do nerazumijevanja osnovnih koncepata. Slijedeća definicija konkurentnosti i paralelizma je konzistentna i smatra se točnom [63].

*Definicija 1:***Konkurentnost** je svojstvo programa (na razini dizajna) gdje dva ili više zadataka mogu biti u simultanom progresu.

Definicija 2: **Paralelizam** je svojstvo u- izvršavanju*(run-time*) gdje se dva ili više zadataka izvršavaju simultano.

Postoji razliku između stanja kada kažemo da je proces u progresu/napredovanju i u izvršavanju budući prvi ne uključuje nužno izvršavanje. Neka *C* i *P* predstavljaju konkurentnost i paralelizam, tada je *P* podskup od *C*. Drugim riječima paralelizam zahtjeva konkurentnost, ali konkurentnost ne zahtjeva paralelizam. Zgodan primjer predstavlja operacijski sustav (OS); konkurentan je po dizajnu (izvodi višezadaćni rad, mnogi zadaci /procesi se nalaze napredovanju), a broj zadataka koje može izvršiti paralelno ovisi o broju fizičkih jedinica za obradu.

Definicija 3. **Paralelno računarstvo** predstavlja način rješavanja problema veličine *n* dijeljenjem domene u *k* dijelova ($k \geq 2, \ k \in \mathbb{N}$) te njihovim simultanim rješavanjem sa *p* fizičkih procesora.

Biti u mogućnosti identificirati tip problema je esencijalno u formuliranju paralelnog algoritma. Neka P_D bude problem u domeni D. Ako je P_D moguće paralelizirati tada se D može dekompozirati u *k* pod-problema:

$$D = d_1 + d_2 + \cdots + d_k = \sum_{i=1}^{k} d_i \quad (1)$$

P_D predstavlja podatkovni problem ako je D sastavljena od podatkovnih elemenata i rješavanje problema zahtjeva primjenu jezgrene funkcije *f(...)* na cijelu domenu:

$$f(D) = f(d_1) + f(d_2) + \cdots + f(d_k) = \sum_{i=1}^{k} f(d_i) \quad (2)$$

P_D predstavlja paralelizam po zadacima ako je D sastavljena od funkcija i njihovo rješavanje zahtjeva primjenu svake od tih funkcije na zajedničkom toku podataka S:

$$D(S) = d_1(S) + d_2(S) + \cdots + d_k(S) = \sum_{i=1}^{k} d_i(s) \quad (3)$$

Podatkovni paralelni problemi su idealni kandidati za GPU. Razlog tome je što GPU arhitektura najbolje funkcionira kada sve niti izvršavaju iste instrukcije, ali na drugačijim podacima. S druge strane paralelni problemi koji se temelje na zadacima su najprikladniji za CPU. Razlog tome je što CPU arhitektura omogućuje izvođenje različitih zadataka na svakoj niti.

Ova klasifikacijska shema je kritična za ostvarivanje najboljeg particioniranja problema u domeni,što zapravo predstavlja prvi korak pri izradi paralelnih algoritama. Također osigurava korisne informacije pri odabiru najboljeg hardvera za implementaciju (CPU ili GPU). Računalni problemi vezani za obradu slike često se klasificiraju kao podatkovno paralelni, pa su dobri kandidati za masivnu paralelizaciju na GPU.

5.2. Mjere za performanse

Mjerenje performansi se sastoji od skupa metrika koje se mogu koristiti za kvantificiranje kvalitete algoritma. Za sekvencijalne algoritme dovoljna metrika su vrijeme i memorijski prostor. Za ispitivanje kvalitete paralelnih algoritama potrebne su i druge metrike kao što su ubrzanje i efikasnost. Nadalje kada se algoritam ne može u potpunosti paralelizirati korisno je napraviti teoretsku procjenu maksimalnog mogućeg ubrzanja. U ovim slučajevima zakoni kao Amdahloov i Gustafsonov postaju korisni u analizi. Na eksperimentalnoj strani, metrike kao što su memorijska propusnost i broj operacija sa pomičnim zarezom u sekundi definiraju performanse paralelne arhitekture kada se algoritmi izvode paralelno.

Za zadani problem veličine *n*, vrijeme izvođenja paralelnog algoritma korištenjem *p* procesora se označava kao $T(n,p)$. Sa teoretskog stajališta *rad i raspon* predstavljaju osnovu za druge metrike kao što su ubrzanje i efikasnost.

Rad izvršen od strane p procesora predstavlja ukupni broj primitivnih operacija koje procesor može izvesti ignorirajući dodatno vrijeme (*engl. overhead*) komunikacije zbog sinkronizacije procesora. Zapravo je jednako vremenu izvođenja računanja na jednom procesoru; označava se $T(n,1)$
Raspon ili kritična putanja definira se kao duljina najduljeg niza operacija koji se moraju izvesti sekvencijalno zbog podatkovne ovisnosti [64]. Odnosno najdulje vrijeme koje potrebno za izvršavanje paralelne putanje jedne izvršne niti; označava se kao $T(n,\infty)$.
Raspon trajanja je ekvivalentan mjerenju vremena kada se koristi beskonačan broj procesora.

Ove dvije metrike pružaju donje granice za $T(n,p)$. Jednadžba *zakona rada* kaže da je prva donja granica:

$$T(n,p) \geq \frac{T(n,1)}{p} \tag{4}$$

Vrijeme izvođenja paralelnog algoritma iznosi minimalno *1/p* trajanja rada. S ovim zakonom može se doći do zaključka da paralelni algoritmi rade brže kada je rad po procesoru uravnoteženiji.

Zakon za raspon trajanja definira drugu donju granicu za $T(n,p)$:

$$T(n,p) \geq T(n,\infty) \tag{5}$$

Ovo znači da vrijeme izvođenja paralelnog algoritma ne može biti manje od raspona trajanja ili minimalne količine vremena koje je potrebno računalnom stroju sa beskonačnim brojem procesora.

5.2.1. Ubrzanje

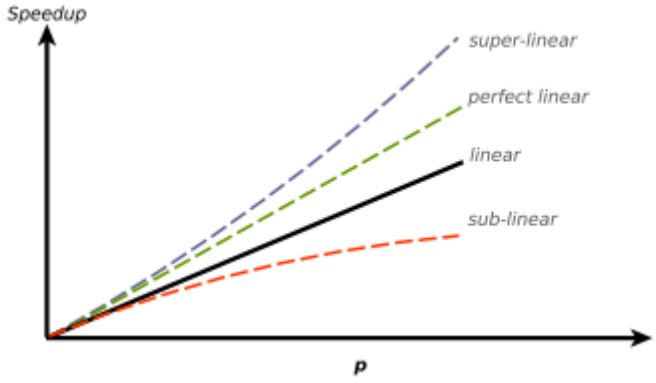
Jedan od najvažnijih postupaka u paralelnom računarstvu je mjerenje kako dobro se paralelni algoritam izvršava u odnosu na najbolju implementaciju sekvencijalnog algoritma. Ova mjera se zove ubrzanje. Za problem veličine, izraz za ubrzanje je:

$$S_p = \frac{T_s(n,1)}{T(n,p)} \tag{6}$$

gdje$T_s(n,1)$predstavlja najbolji sekvencijalni algoritam, a $T(n,p)$ predstavlja vrijeme izvođenja paralelnog algoritma sa *p* procesora koji izvršavaju isti problem.

Ubrzanje je ograničeno na gornju granicu kada je *n* fiksiran zbog zakona o radu iz jednadžbe 4. pa je $S_p \leq p$

Ako se ubrzanje povećava linearno kao funkcija od *p*, tada govorimo o linearnom ubrzanju. Linearno ubrzanje znači da je *overhead* algoritma uvijek u istom odnosu s obzirom na vrijeme izvođenja za svaki p. U posebnom slučaju kada je $T(n,p) = {}^{T_s(n,p)}/_{p}$ tada govorimo o idealnom ubrzanju ili *savršenom linearnom ubrzanju*. To je maksimalna teoretska vrijednost ubrzanja koji paralelni algoritam može ostvariti. U praksi je veoma teško ostvariti linearno ubrzanje, a kamoli savršeno linearno ubrzanje zbog uskog grla memorije i dodatnog troška vremena zbog komunikacije koji se povećava kao funkcija *p*. Ono što nalazimo u praksi jest da većina programa ostvaruje sub-linearno ubrzanje koje iznosi $T(n,p) \geq T_s(n,p)/p$. Slika 5.1 pokazuje četiri moguće krivulje ubrzanja.



Slika 5.1. Četiri moguće krivulje za ubrzanje

Već tri desetljeća debatira se da li je superlinearno ubrzanje moguće ili je riječ o fantaziji. Super-linearno ubrzanje je važna materija u paralelnom računarstvu jer bi dokazivanje samog postojanja značilo da su paralelni strojevi više od sume njihovih dijelova [65]. Smith i Faber u svojim radovima kažu da nije moguće postići super-linearno ubrzanje jer da takav algoritam postoji onda bi njegovo izvršavanje na jednom procesoru ili jezgri bilo *p* puta sporije (što opet dovodi do linearnog ubrzanja). S druge strane u Parkisonovu radu veznom za efikasnost paralelizma kaže se da je superlinearno ubrzanje nekad moguće zbog overheada u petlji jednog procesora. Gustafson podržava ideju superlinearnog ubrzanja.

5.2.2. Amdahlov zakon

Neka *c* predstavlja dio programa koji je paralelan, $(1-c)$ je dio koji se izvršava sekvencijalno, a p broj procesora. Amdahlov zakon [66]kaže da za problem fiksne veličine očekivano ukupno ubrzanje je dano jednadžbom:

$$S(p) = \frac{1}{(1-c) + \frac{c}{p}} \tag{6}$$

Ako je $p \approx \infty$ jednadžba postaje:

$$S(p) = \frac{1}{1-c} \tag{7}$$

tj. ako računalo ima veći broj procesora (super –računalo ili moderni GPU) tada je maksimalno ubrzanje limitirano sa sekvencijalnim dijelom programa (tj. ako je c=4/5 tada je maksimalno ubrzanje 5x).

5.2.3. Gustafsonov zakon

Gustafsonov zakon je druga korisna mjera za teoretsku analizu performansi. Ova metrika ne podrazumijeva fiksnu veličinu problema kao Amdahlov zakon. Umjesto toga koristi fiksni vremenski model gdje se rad po procesoru drži konstantnim kada se povećava *p* i *n*. U Gustafsonovom zakonu vrijeme paralelnog programa se sastoji od sekvencijalnog dijela *s* i paralelnog dijela *c* kojeg izvršava *p* procesora.

$$T(p) = s + c \tag{8}$$

Ako je sekvencijalno vrijeme za računanje *s+cp* tada je ubrzanje:

$$S(p) = \frac{s+cp}{s+c} = \frac{s}{s+c} + \frac{cp}{s+c} \tag{9}$$

Definiranjem α kao dio serijskog računanja $\alpha = s/(s+c)$, tada je paralelni dio $1-\alpha = c/(s+c)$. Najzad jednadžba postaje ubrzanje sa fiksnim vremenom S(p):

$$S(p) = \alpha + p(1-\alpha) = p - \alpha(p-1) \tag{10}$$

Gustafsonov zakon predstavlja važno proširenje znanja o paralelnom računarstvu i definiranju ubrzanja. Sa Gustafsonovim zakonom predstavljeno je povećanje linearnog rada kao funkcija od *p* i *n*. Sada problem više nije fiksan, već umjesto toga fiksan je rad po procesoru. Ovo skaliranje je poznato i kao *meko skaliranje*. Postoji mnogo aplikacija gdje se veličina problema uvećava ako je dostupna veća računalna moć. Predviđanja vremena, računalna grafika, Monte Carlo algoritmi, simulacija čestica su samo neki od primjera. Fiksiranje veličine problema i mjerenjem problema kao vrijeme nasuprot broju procesora *p* se najčešće radi za akademske svrhe. Što problem postaje veći, paralelni dio *p* može rasti brže nego α.

Iako je istina da je ubrzanje jedno od najvažnijih mjera za paralelno računanje, postoje također i druge metrike koje daju dodatne informacije o kvaliteti paralelnih algoritama kao što je efikasnost.

5.2.4. Efikasnost

Ako podijelimo jednadžbu 6 sa *p* dobivamo:

$$E_p = \frac{S_p}{p} = \frac{T_s(n,1)}{pT(n,p)} \leq 1 \tag{11}$$

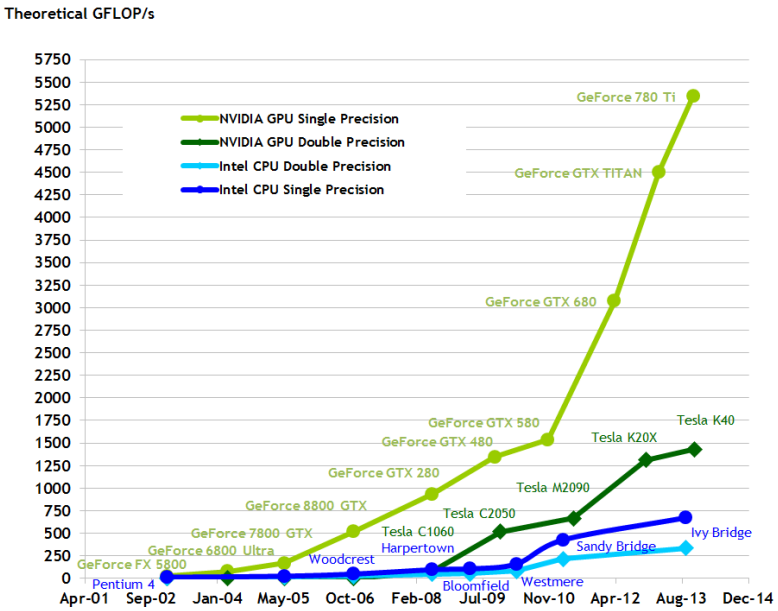
Parametar E_p predstavlja efikasnost algoritma koji radi na *p* procesora i koji nam govori kako dobro se iskorištava procesor. $E_p = 1$ je maksimalna efikasnost i znači optimalno korištenje računalnih resursa. Maksimalna efikasnost je teško ostvariva u implementacijama. Danas je efikasnost postala važna kao i ubrzanje, ako ne i više jer nam govori kako dobro se koristi hardver i govori nam koje implementacije imaju prioritet kada se natječu za limitirane resurse(klaster, superračunalo, radna stanica).

5.2.5. FLOPS

FLOPS metrika predstavlja sirove aritmetičke performanse i mjeri se kao broj operacija sa pomičnim zarezom u sekundi. Neka F_h budu vršne performansi u operacijama sa pomičnim zarezom za poznati hardver, a F_e performanse implementacija algoritma u vidu operacija sa pomičnim zarezom tada se F_e definira kao:

$$F_c = \frac{F_e}{F_h} \tag{12}$$

Fc predstavlja efikasnost numeričkog proračuna koje je relativno za zadani hardver. Vrijednost Fc=1 znači maksimum iskorištenja hardvera u svrhu numeričkih proračuna.



Slika 5.2 Teoretski broj GFLOP/s između GPU-a i CPU-a

Današnji *high-end* CPU pruža 240 GFLOP/s jednostruke preciznosti (Intel Xeon E5 2690), dok high-end GPU pružaju približno 4 TFLOP/s (Nvidia Tesla K20x) jednostruke preciznosti što je ekvivalentno superračunalima otprije 10 godina.

5.2.6. Performanse po Watt

U nedavno vrijeme potrošnja energije je postala važnija od samog sirovog ubrzanja. Performanse po *watt-u* je jedna od najvažnijih mjera za biranje hardvera i predmet je istraživanja. Inicijativa za razvoj energetski efikasnog hardvera je započela razvijanjem HPC-a na odgovoran način. Titan superračunalo troši 8.2 MW i pruža 2.1 GFlops/W dok Nvidia Tesla K20X pruža 16.8 GFlops/W korištenjem 300W. Integracija GPU u superračunala je pomogla da ovakvi sustavi budu više energetski efikasni.

5.2.7. Memorijska propusnost

Memorijska propusnost je stopa pri kojoj se podaci prenose između procesora i glavne memorije. Uobičajeno se mjeri kao GB/s. Memorijska efikasnost B_C implementacije se mjeri dijeljenjem eksperimentalne propusnosti B_e sa maksimalnom propusnosti hardvera:

$$B_c = \frac{B_e}{B_h} \tag{13}$$

Vrijednost B_C=1 znači da aplikacija koristi maksimalnu memorijsku propusnost koja je dostupna na hardveru. Aktualni CPU *high-end* procesori imaju memorijsku propusnost u rangu od $40GB/s \leq B_h \leq 80GB/s$ dok high-end GPU kartice imaju raspon memorijske propusnosti od $200GB/s \leq B_h \leq 300GB/s$.

Ostvarenje maksimalne propusnosti u GPU je mnogo teže nego na CPU. Glavni razlog je što su memorijske performanse ovisne o problemu. Strukture podataka moraju biti poravnate u jednostavne uzorke tako da se mnogi skupovi podataka čitaju ili pišu simultano sa minimalnim troškovima hardvera. Neregularni pristupi podacima, poravnanja i različite veličine podataka rezultiraju manjom memorijskom propusnosti.

5.3. Arhitektura

Računalna arhitektura definira način na koji procesori rade, komuniciraju i kako je memorija organizirana. Cilj računalne arhitekture je omogućiti računalima da izvode programe efikasno i što je brže moguće. U prošlosti, implementacije su automatski ostvarivale veće performanse zbog toga što je hardverska industrija povećavala frekvenciju procesora. U to vrijeme nije bilo mnogo promjena kada je riječ o računalnoj arhitekturi. Danas su računala evoluirala u paralelne strojeve budući su

procesori sa jednom jezgrom dosegli svoj limit kada je riječ o frekvenciji [67]. Današnje najvažnije arhitekture su više-jezgreni i mnogo-jezgreni procesori koji su predstavljeni sa CPU i GPU. Nažalost sekvencijalne implementacije programa neće raditi brže sa samom kupnjom boljeg hardvera. Moraju biti redizajnirane kao paralelni algoritmi koji mogu skalirati svoje performanse sa dodavanjem više procesora. Aspekti kao što su tip instrukcija/podatkovnih tokova, organizacije memorije i komunikacije procesora uistinu pomažu u ostvarivanju boljih implementacija.

5.3.1. Flynn-ova taksonomija

Arhitektura računala se može klasificirati po Flynn-ovoj taksonomiji u četiri kategorije. Klasifikacija ovisi o dva aspekta: broj instrukcija i broj podatkovnih tokova koji se mogu izvršavati paralelno.

SISD ili jedna instrukcija - jedan tok podataka može izvoditi samo jednu instrukciju u vremenu za jedan tok podataka. Ne postoji paralelizam uopće. Stari jedno-jezgreni procesori iz 1950-tih koji se temelje na izvornoj Von Neumanovoj arhitekturi su bili svi SISD tipa. Intel procesori od 8086 do 80486 su također SISD.

SIMD ili jedna instrukcija – više podatkovnih tokova može izvršavati jednu instrukciju, ali se simultano može primijeniti na više podatkovnih tokova. Ovakve arhitekture dozvoljavaju podatkovni paralelizam koji je koristan u znanosti kod primjene matematičkih modela za različite dijelove domene problema. Vektorska računala 70-tih i 80-tih su prva implementirala ovakvu arhitekturu. Danas se za GPU smatra da je riječ o evaluiranoj SIMD arhitekturi jer rade sa mnogo SIMD dijelova simultano. SIMD je također podržan i na CPU kao instrukcijski skup kao MMX, 3Dnow, SSE i AVX. Ovi instrukcijski skupovi dopuštaju cjelobrojne paralelne ili operacije sa pomičnim zarezom oko malih nizova podataka.

MISD, ili višestruke instrukcije na jednom toku podataka mogu rukovati sa različitim zadacima oko istog toka podataka. Ove arhitekture nisu tako uobičajene i često su implementirane za specifične namjene kao digitalni sustavi za napad (tj. otkrivanje enkripcije na podacima) ili sustavi koji toleriraju ispade i kontrolore svemirskih letova (NASA).

MIMD ili višestruke instrukcije nad višestrukim tokovima podataka je najfleksibilnija arhitektura. Može podržati jednu različitu instrukciju za svaki tok podataka i može ostvariti bilo koji tip paralelizma. Ipak, kompleksnost fizičke implementacije je velika i često overhead u upravljanju takvim različitim zadacima i podatkovnim tokovima postaje problem kada se pokušava skalirati na veći broj jezgri. Moderni CPU spadaju u ovu kategoriju (Intel, AMD i ARM više-jezgreni procesori) i novije GPU arhitekture parcijalno implementiraju ovu arhitekturu. MIMD koncept se dijeli na SPMD

(jedan program višestruki podaci) i MPMD (višestruki programi višestruki podaci). SPMD se javlja kada se jedan program izvršava na različitim procesorima. Ključna razlika u usporedbi sa SIMD je da kod SPMD svaki procesor može biti u različitim fazama izvršavanja ili na različitim putanjama koda koji uzrokuju uvjetno grananje. MPMD se javlja kada različiti neovisni programi se izvršavaju na različitim procesorima.

5.3.2. Arhitektura i organizacija memorije

Postoje dva oblika organizacije memorije: dijeljena i distribuirana. U distribuiranoj memoriji svaki čvor ima svoju memorijsku arhitekturu i od drugih čvorova je kompletno neovisan. Komunikacija se temelji na porukama između čvorova putem mreže. U scenariju distribuirane memorije mreža igra važnu ulogu i njihova topologija je različita što ovisi o kontekstu. Neke česte topologije su sabirnica, zvijezda, prsten, mreža (*engl. mesh*), hiperkub i stablo. Postoje i hibridne topologije koje se grade na temelju ovih spomenutih.
U dijeljenoj organizaciji memorije, procesori komuniciraju kroz zajedničku banku globalne memorije bez potrebe za eksplicitnim porukama kao u shemi distribuirane memorije. Danas se koriste najčešće dvije arhitekture: UMA i NUMA.

Uniformni memorijski pristup ili UMA se sastoji od dijeljene memorije u kojoj je pristupno vrijeme za bilo koji procesor uzima istu količinu vremena bez obzira na lokaciju podataka. UMA je također poznata kao Symetric Multi-Processors ili SMP. Glavni nedostatak UMA je mala skalabilnost kada se povećava broj procesora. Ovo se javlja zbog toga što se jedan memorijski kontroler dijeli na sve procesore.

Ne-uniformni memorijski pristup ili NUMA je arhitektura gdje pristupno vrijeme dijeljenoj memoriji ovisi o udaljenosti lokacije podataka u odnosu na procesor. Ovo znači da se memoriji koja je bliža procesoru pristupa mnogo brže od memorije koja je bliža drugom procesoru (trošak je funkcija udaljenosti). Kako bi se iskoristile prednosti NUMA-e, problem se može podijeliti u neovisne dijelove podataka gdje je svaki dodijeljen jedinstvenom CPU. Također, globalni podaci koji se mogu samo čitati se bolje repliciraju od dijeljenih. U praksi sve NUMA arhitekture implementiraju hardversku keš koherentnu logiku i postaju koherentni NUMA ili ccNUMA.

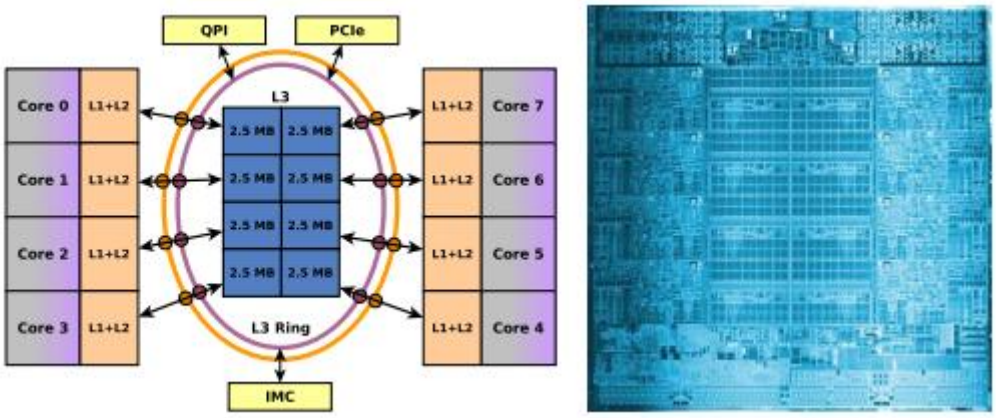
SMP arhitekturu možemo pronaći u mnogim osobnim računalima sa dual-core hardverom,a NUMA arhitekturu u modernim višejezgrenim strojevima sa dvije ili više CPU priključaka (*engl.socketa*). Slika 5.3. pokazuje UMU i NUMA arhitekturu.

Slika 5.3. UMA (odnosno SMP) i NUMA memorijska arhitektura

Valja naglasiti da se dijeljena i distribuirana arhitektura memorije može kombinirati što dovodi do hibridne konfiguracije kao što je MPI + OpenMP ili MPI + GPGPU rješenja.

5.4. Tehnički detalji modernih CPU i GPU arhitektura

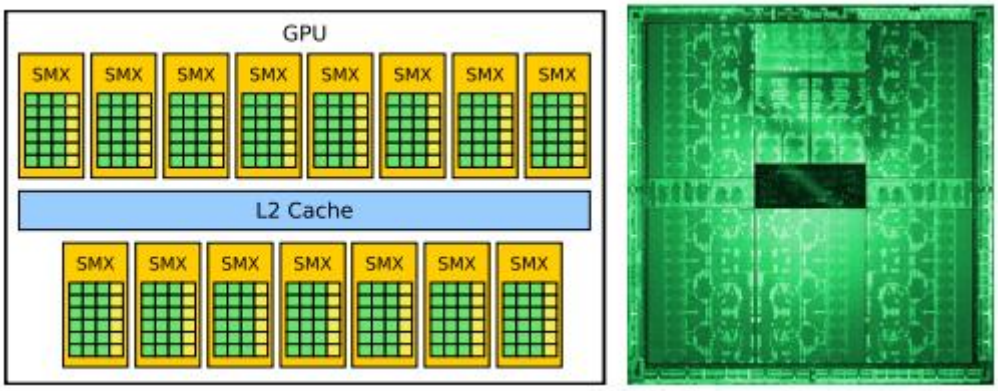
Razlike između više-jezgrenih i mnogo jezgrenih arhitektura se mogu vizualizirati shemom modernog CPU-a i GPU-a. High-end CPU-i kao što su Xeon E5 2600 su građeni sa mnogim inter-konekcijama između jezgri što pruža fleksibilnost u komunikaciji (Slika 5.4)



Slika 5.4. Intel Xeon

Svaka jezgra sadrži lokalnu L1 i L2 keš memoriju od 64KB i 256KB, a u središtu čipa se nalazi veća L3 keš memorija veličine 20MB koju dijele sve jezgre putem prstenaste sheme. Quick-Path Interconnect ili QPI sekcija čipa implementira dio NUMA arhitekture. PCI modul upravlja komunikacijom sa PCI portovima i sa Internal Memory Controller-om (unutarnjim memorijskim kontrolerom) ili IMC-om.

S druge strane moderni GPU kao Tesla K20X imaju potpuno različitu shemu čipa koja je orijentirana na masivni paralelizam. Slika 5.5. pokazuje shemu Nvidia Tesla K20X GPU kao i aktualni čip. Jezgre GPU-a su grupirane u SMX jedinice ili *streaming multiprocesore*. Najvažniji dio koji karakterizira GPU se nalazi u SMX jedinicama.



Slika 5.4. Shema Tesla K20

SMX je najmanja jedinica sposobna za izvođenje paralelnog računanja. Glavna razlika između low-end i high-end GPU–a iste arhitekture je broj SMX jedinica u čipu. U slučaju Tesla K20 GPU-a svaka SMX jedinica je sastavljena od 192 jezgre. Arhitektura je napravljena za maksimum od 15 SMX-ova što daje maksimalno 2880 jezgri.

Jezgre SMX-a su 32 bitne jedinice koje izvode osnovne cjelobrojnu i aritmetiku sa pomičnim zarezom jednostruke preciznosti (FP32). Kao dodatak postoje 32 jedinice specijalne namjere ili SFU koje izvode posebne matematičke operacije kao što su log,sqrt, sin, cost itd. Svaki SMX ima 64 jedinice za operacije sa pomičnim zarezom dvostruke preciznosti (predstavljane sa DPC) koje su poznate kao FP64 i 32 LD/ST jedinice (load /store) za pisanje i čitanje iz memorije.

Numeričke performanse GPU-a su klasificirane u dvije kategorije: performanse za FP32 i FP64. Performanse za FP32 su uvijek veće od FP64 performansi. Ovo predstavlja problem za masivno paralelne arhitekture jer se mora potrošiti površina čipa na posebne jedinice za računanje koje povećavaju FP64 performanse. Tesla K20X GPU može ostvariti do 4 TFlops-a FP32 performansi, a samo 1.1 Tflops-a u FP64 modu.

Stvarni GPU-i kao što je Tesla K20 implementiraju memorijsku hijerarhiju na četiri razine

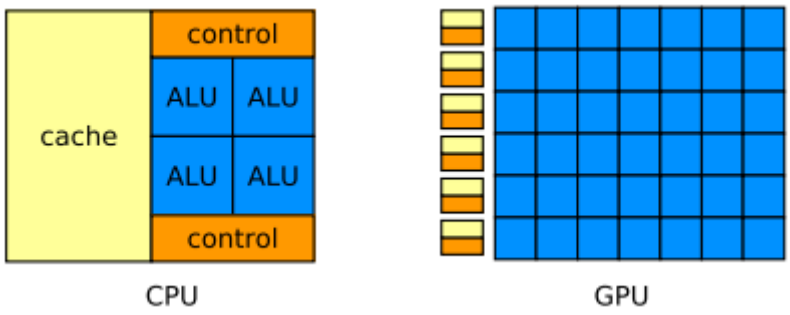
- Registri
- L1 keš memorija
- L2 keš memorija
- Globalna memorija

Svi tipovi memorije osim globalne se nalaze na samom čipu. L2 keš automatski poboljšava pristup globalnoj memoriji. L1 keš je manualan i postoji jedan po SMX-u, te može biti brz kao i registri. Kepler i Fermi temeljeni GPU-i imaju L1 keš veličine 64 KB koji se dijeli na 16 KB programibilne dijeljene memorije i 48 KB automatskog keša ili obrnuto.

5.5. Osnovne razlike između CPU i GPU arhitekture

Moderni CPU-i su evoluirali prema paralelnoj obradi implementiranjem MIMD arhitekture. Većina površine čipa je rezervirana za upravljačke jedinice i keš i tako ostavljaju malo područje za numeričke izračune. Razlog je što CPU izvodi raznovrsne zadatke te su stoga dodatni keš i dodatni kontrolni mehanizmi jedini način da ostvarimo dobre ukupne performanse.

S druge strane GPU sadrži SIMD temeljenu arhitekturu koja može veoma dobro predstaviti PRAM i UPMH modele. Glavni cilj GPU arhitekture je ostvarenje visokih performansi kroz masivni paralelizam. Nasuprot CPU, površina čipa na GPU je uglavnom okupirana od aritmetičko logičkih jedinica (ALU), a za upravljačke mehanizme i keš je ostavljeno minimalno područje. Efikasni algoritmi koji su dizajnirani za GPU su ostvarili 100x ubrzanje nasuprot CPU implementacija.



Slika 5.5. Razlika u arhitekturi GPU-a i CPU-a

Razlika u arhitekturi shodno tome ima direktne posljedicu u vidu da su GPU mnogo restriktivniji od CPU-a, ali s druge strane su mnogo moćniji ako se rješenje pažljivo dizajnira. Najnovije GPU arhitekture kao Fermi i Kepler su dodale značajni nivo fleksibilnosti uključivanjem L2 keša za pristup neregularnim pristupima memoriji i unaprjeđivanjem performansi atomičnih operacija. Ipak ova fleksibilnost uvelike zaostaje za onom u CPU.

Dakle postoji kompromis između fleksibilnosti i računalne moći. Stvarni CPU se muče da održe ravnotežu između računalne moći i opće namjenske funkcionalnosti dok GPU-i ciljaju na masivna paralelne aritmetička računanja uvodeći mnoge restrikcije. Neke od ovih restrikcija nastoje nadići fazu implementacije dok druge se moraju tretirati kada se problem paralelizira.

5.6. GPU računarstvo

GPU računarstvo koristi GPU-a kao opće-namjensku jedinicu za rješavanje zadanog problema koji nije vezan za kontekst grafike. Poznat je kao akronim GPGPU. Cilj GPU računarstva je ostvarenje velikih performansi za probleme koji su podatkovno paralelni kroz masivno paralelne algoritme koji se izvode na GPU.

GPU računarstvo je započelo kao istraživačko područje u računalnoj grafici u ranim 2000-tim godinama. U godini 2006 izdan je API za općenamjensko računanje na GPU od strane NVIDIA korporacije i nazvan je CUDA (*engl. Compute Unified Device Architecture*). Tehnički CUDA API je ekstenzija C jezika koji prevodi kod koji se izvršava na GPU. Izdavanje CUDA je postao važna prekretnica u povijesti GPU računanja jer je riječ o prvom API koji je nudio efektivnu dokumentaciju za rad u ovom području.

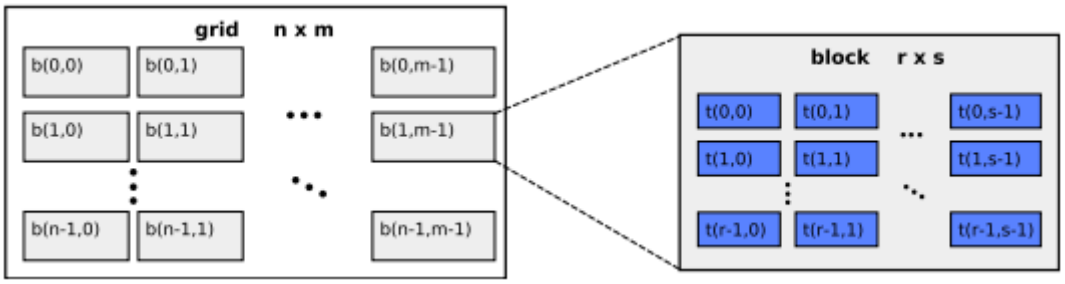
Godine 2008 izdat je otvoreni standard pod nazivom OpenCL koji je omogućio više-platformsku paradigmu za pisanje masivnog paralelnog koda. Ovaj programski model je sličan CUDA-i ali koristi različita imena za iste strukture. Programski model iza CUDA-e i OpenCL predstavlja ključni aspekt za GPU računanje jer definira nekoliko komponenti koje su esencijalne u implementaciji masivnih paralelnih algoritama.

5.6.1. Programski modeli za masivni paralelizam

GPU programiranje je karakterizirano sa visokom razinom paralelizma što je popraćeno imenom programski model za masivni paralelizam. Ovaj model je apstraktni sloj koji leži na vrhu GPU arhitekture. Dopušta dizajniranje masivnih paralelnih algoritama neovisno o broju dostupnih fizičkih procesnih jedinica.

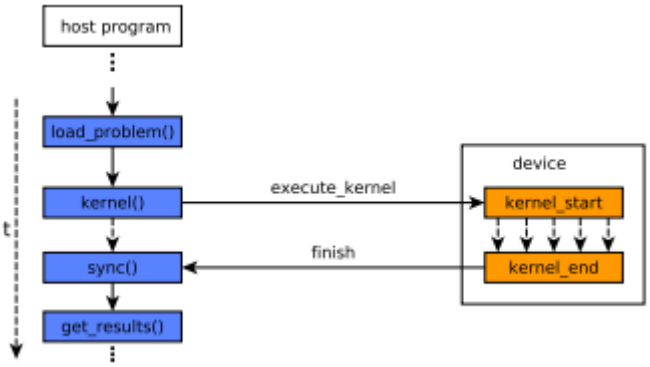
Apstrakcija je ostvarena prostorom računanja koji se definira kao diskretni prostor u kojem su organizirane veliki broj niti odnosno izvršnih tokova. U CUDA, prostor računanja je sastavljen od mreže, blokova i niti. Kod OpenCl termini su radni prostor, radna grupa i radna jedinka. Mreža ili *grid* je predstavljena kao diskretna k-dimenzionalna (sa k=1,2,3) struktura pravokutnog tipa koja definira veličinu i volumen prostora računanja. Svaki element grid-a je blok. Blokovi su manje k-dimenzionalne (k=1,2,3) jedinice identificirane preko koordinata koje su relativne mreži. Svaki blok sadrži mnogo prostorno organiziranih niti. Konačno, svaka nit sadrži koordinate relativne s obzirom na blok kojom pripadaju. Koordinatni sustav karakterizira prostor računanja i služi kao mapa za niti u različitim lokacijama problema. Slika 5.6. Ilustrira primjer dvodimenzionalnog prostora računanja. Svaki blok ima pristup maloj lokalnoj memoriji koja je u CUDI poznata kao dijeljena memorija (u

OpenCl je poznata kao lokalna memorija). U praktičnim terminima dijeljena memorija radi kao ručni keš. Važno je načiniti dobro korištenje brze memorije kako bi ostvarili vršne performanse u GPU.



Slika 5.6. Podjela izvršnih niti po blokovima i mreži

Programski radni tok GPU računarstva se gleda kao *host-device* odnos između CPU-a i GPU-a. Host program (tj. C program) radi *upload* problema na uređaj (GPU memorija), potom poziva kernel (funkcija pisana radi izvršavanja na GPU) kojoj se prosljeđuje kao parametri za grid i veličinu bloka. Host program može raditi u sinkronom i asinkronom načinu što ovisi o rezultatima dobivenih iz GPU, a rezultati se kopiraju nazad sa device na host memoriju. Slika 5.7. sumarizira radni-tok.



Slika 5.7. Tok GPU programa

5.6.2. Upravljanje nitima , GPU konkurentnost i tehnička razmatranja

GPU opravlja malim grupama koje rade u SIMD modu. Na AMD grafičkim procesorima ove grupe su poznate kao *wavefronts* i njihova veličina iznosi 64 niti za aktualne grafičke procesorske arhitekture. Kod Nvidia GPU ove grupe su poznate kao *warp-ovi* i aktualne arhitekture kao Kepler i Maxwell rade sa veličinom od 32 niti. OpenCL standard koristi deskriptivni naziv SIMD širina. Radi jednostavnosti ove grupe ćemo nazvati *warp*-ovi. Ovi grafički procesori podržavaju određeni stupanj konkurentnosti kod upravljanja cijelog prostora računanja. Većinu vremena postoji više niti nego što se paralelno može procesirati. Dok se sve niti nalaze u progresu (konkurentnosti) samo mali broj

uistinu se izvršava paralelno. Maksimalni broj paralelni niti koje se izvršavaju na GPU normalno odgovaraju broju procesnih jedinki. Ipak, maksimalni broj konkurentnih niti je mnogo veći. Npr. Geforce GTX 580 GPU može procesirati do 512 niti paralelno, ali može upravljati do 24.576 konkurentnih niti. Za najveći broj problema, preporuka je premašiti kapacitet paralelnog računanja. Razlog tome je što GPU raspoređivač niti ili dretvi je dovoljno pametan kod prebacivanja warp-ova koji su u stanju čekanja (warp-ovi koji čekaju na memorijski pristup ili rezultate specijalnih funkcija kao što se *sqrt*) sa onim koji su spremni na računanje. Drugim riječima, postoji mali cjevovod numeričkih računanja i memorijskog pristupa koji raspoređivač nastoji održavati zaposlenim cijelo vrijeme.

GPU zajednica često koristi termine kao združena memorija (*engl. coalesced memory*), padding i grananje. Riječ je o kritičnim tehničkim razmatranjima koja se moraju uzeti u obzir kako bi ostvarili najbolje performanse na GPU. Ove detalje sam više obradio u idućem poglavlju

6. Detekcija istaknutih objekata u UAV snimanju

Suvremene operacije pretrage i nadzora mogu biti jako kompleksne i skupe te mogu uključivati veliki broj ljudi uz mnogobrojne zračne platforme za pretragu. Često se pretražuju široka prostranstva i otvoreni prostori na planinama, oceanima ili pustinjskim područjima. Primjeri takvih pretraga i operacija su nestanak Steve Fosset-a [68] čiji se mali zrakoplov srušio u planinskom masivu Stjenjak u SAD-u (2007), te potraga sa ostacima leta Air France 447 (2009) [69]. Lokacija pada Fossetovog zrakoplova je slučajno otkrivena tek nakon godinu dana od pada. Pretraga je između ostalog koristila 5000 volontera koji su analizirali na tisuće digitalnih satelitskih fotografija koje su pokrivale nekoliko stotina kvadratnih kilometara površine terena. Što se tiče leta 447, potraga je trajala oko 5 dana prije nego što su u Tihom oceanu pronađeni prvi ostatci, a uključivala je na desetine brodova i zrakoplova. Ova dva slučaja predstavljaju klasični scenarij za automatsku analizu slika prikupljenih sa satelita kao i putem zračnog snimanja na različitim letjelicama. U oba scenarija tražilo se nešto jedinstveno, unikatno u jednostavnom i uniformnom okruženju (primjerice plutajuća olupina zrakoplova u slučaju leta 447 – Slika 6.1).

Slika 6.1. Ostatci zrakoplova A320 leta 447 [69].

Iako slijed događaja za pojedinu misiju pretrage i spašavanja može uvelike varirati s obzirom na scenarij, okoliš i mnoge dodatne faktore, neke zajedničke osobine i karakteristike se mogu pronaći.

Najistaknutija karakteristika je činjenica da se objekt pretrage često značajno razlikuje od okruženja u kojem se odvija pretraga. Ovakav scenarij je veoma zahvalan za aplikacije koje se temelje na detekciji objekata korištenjem računalnog vida. Jedan od mnogih ciljeva računalnog vida je automatsko izdvajanje *vizualno važnih* dijelova scena. Detekcija ispupčenih objekata korištenjem računalnog vida se razvila kao entitet unutar sebe. Postupak detekcije istaknutih objekata u svojoj osnovi predstavlja detekciju vizualno jedinstvenih objekata unutar zadane slike. Upravo ta značajka omogućuje da

detekcija dobro odgovori na probleme automatske analize slike za slučajeve spašavanja koji su ranije navedeni. Iako se mape ispupčenosti koriste unutar kompleksnijih sustava računalnog vida, iznenađujuće je da se veoma malo primjenjivala na sustave koji se bavi pretragama na slikama dobivenih snimanjem iz zraka korištenjem bespilotnih letjelica (UAV).

Jedan od razloga je i to što većina slika na kojima se radi detekcija ispupčenosti su veoma visoke kvalitete, objekti su relativno velike razlučivosti sa minimalno šuma, a najčešće se traži jedan objekt koji je predstavljen kao glavni objekt u slici (Slika 6.2.).

Slika 6.2. Tipična slika za detekciju istaknutih objekata

S druge strane, ako radimo sa slikama dobivenih preko bespilotnih letjelica sasvim je razumno očekivati višestruke objekte od interesa i važnosti te su najčešće relativno malih dimenzija. Oveslike su često lošije kvalitete jer pate od šumova uslijed turbulencije same letjelice, šumova uslijed transmisije i kompresije kao i atmosferskih šumova uslijed vremenskih uvjeta kao što je prikazano na na slikama 6.3 a), b).

a) b)

Slika 6.3 a), 6.3 b). Slike uhvaćene bespilotnom letjelicom istraživačke grupe (Blue Bear System Research) [70]

Fotografije predstavljene slikama 6.3. nam pokazuju moguće aspekte snimanja skupljeni iz UAV platforme. Varirajući uvjeti osvjetljenja, saturacija boje, zamućenje i distorzija šuma uslijed

vremenskih varijacija, brzina platforme, visina, stabilnost kao i dodatni šumovi proistekli iz prijenosa samih podataka su samo neki od mogućih poteškoća kojima se možemo susresti pri UAV snimanju. Konačno, važno je obratiti pažnju na činjenicu na razliku u tipu istaknutih objekata koje želimo detektirati. Naime, istaknuti objekti po svojoj prirodi su mnogo manje upadljivi i uočljivi u UAV slikama nego u takozvanim „subjekt slikama“ koje se tradicionalno koriste u istraživanju u ovoj domeni.

Cilj ovog poglavlja je ispitati koju su mogućnosti korištenja nekih od opisanih algoritama za detekciju ispupčenosti na UAV slikama za potrebe operacija pretrage i spašavanja (ili nadzora). Bitno je naglasiti da je fokus istraživanja na izvodivosti jednog takvog sustava u realnim uvjetima i scenarijima. Stoga ćemo za detekciju ispupčenosti koristiti algoritme koju su se pokazali jako dobrim kada je riječ o brzini izvođenja, a za vrednovanje algoritma kao najvažniji kriterij će biti preciznost.

6.1. Scenarij za detekciju istaknutih objekata

Područje na kojem se izvode misije pretrage i spašavanja se često mogu suziti na skup približno uniformnih okruženja (tj. u slučaju pada zrakoplova u ocean pretraga ostataka se vrši na uniformnim vodenim površinama). Točniji opis okruženja i okoliša nam omogućava da kod UAV misija pretraživanja i spašavanja odredimo preciznije tip slika na kojem se izvodi detekcija. Saznanje da je okruženje generalno uniformno definira pozadinu slika kao uvelike invarijantnu s obzirom na misiju pretrage. Međutim mnogo realnije su situacije kada slika nije uniformna već se traženi objekti nalaze na terenu koji sadrže detalje pa će se istražiti i ponašanje algoritama za slike u prirodnim uvjetima. Kada je riječ o objektima od interesa, zainteresirani smo za pronalaženje preživjelih, krhotina od letjelica, automobili, ostavljene stvari koji mogu poslužiti kao tragovi ili nas zanima samo mjesto nesreće.

Predložena arhitektura treba biti u stanju izdvojiti što više istaknutih predmeta različitih dimenzija unutar scene. Osim toga, zbog različitih i varirajući visina na kojima se nalazi UAV platforma arhitektura za detekciju trebala bi biti u stanju inkorporirati neke prilagodbe mjerila na temelju različitih visinu na kojima se nalazi uređaj.

6.2. Kvalitativna analiza pojedinih modela

U radu [37] detaljno su evaluirani najznačajniji algoritmi za detekciju istaknutosti s obzirom na kvalitetu produciranja mapa ispupčenosti i brzine izvođenja algoritama. Evaluacija je izvršena na standardnoj bazi slika koja se koristi u ovom području, a slike su dimenzija 400 x 300 piksela. U bespilotnim letjelica kao i u sve prisutnijim dronovima koji su namijenjeni fotografiranju ova rezolucija je neadekvatna, pa se najčešće ugrađuje visokokvalitetne, otporne kamere visoke rezolucije kao što je GoPro kamera od 12 Mpx. Dimenzije fotografije su 4000x3000 piksela, što znači 100 puta veća količina podataka i informacija koje pristižu na obradu. U ovoj analizi odabrali smo slijedeće algoritme FT [71], SR [13], SW [24] i LC [43], a objašnjenje kratice se može pronaći u prilogu.

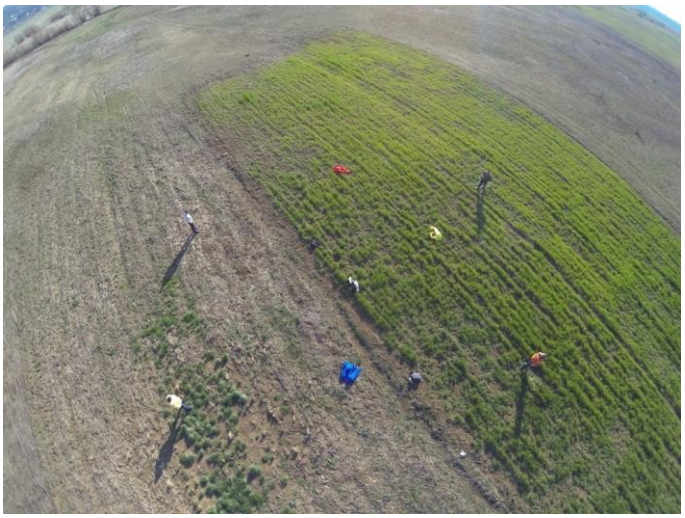
Za postupak detekcije odabran je slijedeći jednostavni postupak sastavljen od slijedećih faza:

- Učitavanje slike
- Kreiranje mape ispupčenosti
- Segmentacija *threshold* operacijom
- Filtiranje objekata s obzirom na veličinu korištenjem analize povezanih komponenti
- Očitavanje kontura i označavanje objekata

Kvalitativna analiza algoritama je izvršena na prijenosnom računalu Asus x55V kojeg pokreće Intel CoreI3 procesor sa dvije jezgre i 4GB RAM-a na. Programi su pokrenuti na Windows 7 operacijskom sustavu uz korištenje Visual Studio 2012 i Matlab programskih paketa. Kao relativno uniformne fotografije odabrali smo nekoliko fotografije slikane GoPro kamerom rezolucije 12 Mpx slikane iz UAV letjelice sa različitih visina i na različitim terenima.

Slika 6.3. Lokacija 1 na Mostarskom blatu (Mostar, BiH)

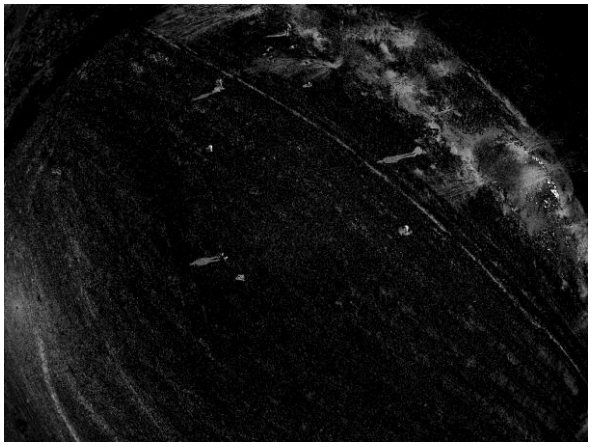
Slika 6.3. Lokacija 2 na Mostarskom blatu (Mostar, BiH)



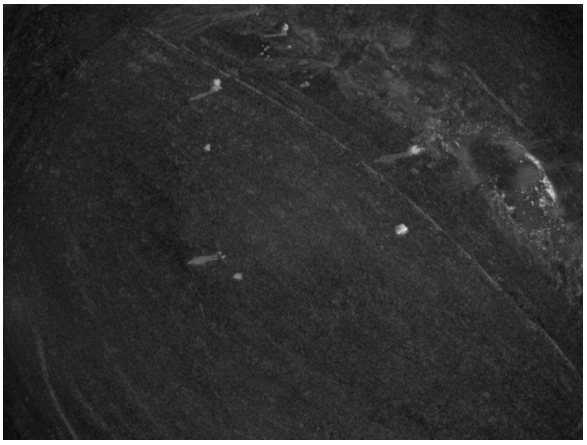
Slika 6.4. Lokacija 3 na Mostarskom blatu (Mostar, BiH)

Slika 6.5. Lokacija 4 na Mostarskom blatu (Mostar, BiH)

U slijedećem nizu sliku nalaze se mape ispupčenosti koje daju pojedini algoritmi za sliku 6.2.

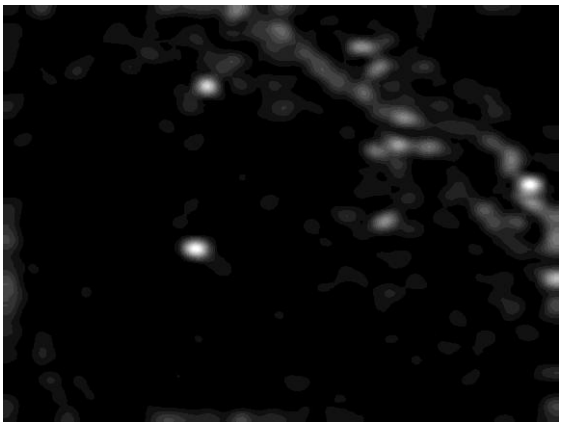


Slika6.5. a) Mapa ispupčenosti - LC algoritam



Slika 6.5. b) Mapa ispupčenosti - SW algoritam

Slika 6.5. c) Mapa ispupčenosti FT -algoritam



Slika 6.5 d) Mapa ispupčenosti SR algoritam

Iz priloženih slika 5. a), b),c),c) uočavaju se mape ispupčenosti koji su bitno pojednostavile početne slike i istaknule određene objekte od interesa.

Nakon *threshold* operacije u mogućnosti smo uraditi analizu povezanih komponenti, te isfiltirati premale i prevelike objekte i potom locirati i omeđiti objekte od interesa.

Slika 6.6. a) Locirani objekti FT -algoritam

Slika 6.6. b) Locirani objekti LC -algoritam

Slika 6.6. c) Locirani objekti SW -algoritam

Slika 6.6. d) Locirani objekti SR -algoritam

Slika 6.7. a) Locirani objekti FT -algoritam

Slika 6.7. b) Locirani objekti LC -algoritam

Slika 6.7. c) Locirani objekti SW -algoritam

Slika 6.7. d) Locirani objekti SR -algoritam

Iz slika 6.6. i slika 6.7. može se jasno uočiti da dva algoritma prednjače kada je riječ o detekciji objekata. To su prije svega FT i SW algoritam koji su pokazali odlične rezultate, a odmah zatim neznatno lošije je LC algoritam. Algoritam SR nije pokazao zadovoljavajuće rezultate u detekciji.

Detekcija objekata u heterogenom okruženju

Slijedeći niz slika predstavlja relativno teži problem jer je okruženje mnogo zahtjevnije. U ovom dijelu usporedio sam FT i SW algoritme.

Slika 6.8. Detekcija FT algoritmom – detektirani su svi objekti sa desne strane, a sa lijeve je detektirana osoba koja se skriva u grmlju

Slika 6.8b. Detekcija SW algoritmom – detektirani svi objekti uz znatno manje lažnih alarma

Slika 6.9a. Detekcija FT algoritmom – svi objekti detektirani

Slika 6.9b. Detekcija SW algoritmom – Detektirani svi traženi objekti uz vidljivo manje lažnih alarma

Nakon kvalitativne analize gdje je od važnosti bila točnost detekcije traženih objekata dva algoritma su iskazala veoma dobre performanse. Modeli FT i SW su pokazali odlične rezultate nad slikama u kojima je relativno uniformno okruženje (travnato područje), te u okruženju koje se može smatrati blago pošumljeno. Međutim, u veoma zahtjevnim slikama gdje su traženi objekti veoma male veličine SW model i algoritam je pokazao svoj puni potencijal. Naime na slici 6. 10.Model FT nije uspio detektirati objekte dok je model SW detektirao dva tražena objekta koja su se odlično uklopila u okruženje. Osim toga slikani su iz mnogo veće udaljenosti pa je čovjek na slici veličine svega 25*20 piksela.

Slika 6.10. Prikaz detekcije ljudi u ležećem i u položaju čučnja na čistini i u grmlju.

Kada je riječ o vremenu izvođenja u tablici 2 se nalazi vrijeme izvedbe svakog modela na slikama rezolucije 4000*3000 piksela odnosno 12 Mpiksela. Algoritmi FT, LC i SR su implementirani u C++ programskom jeziku dok je SW algoritam implementiran u Matlab programskom paketu.

Tablica 2. Performanse algoritama mjerene u sekundama

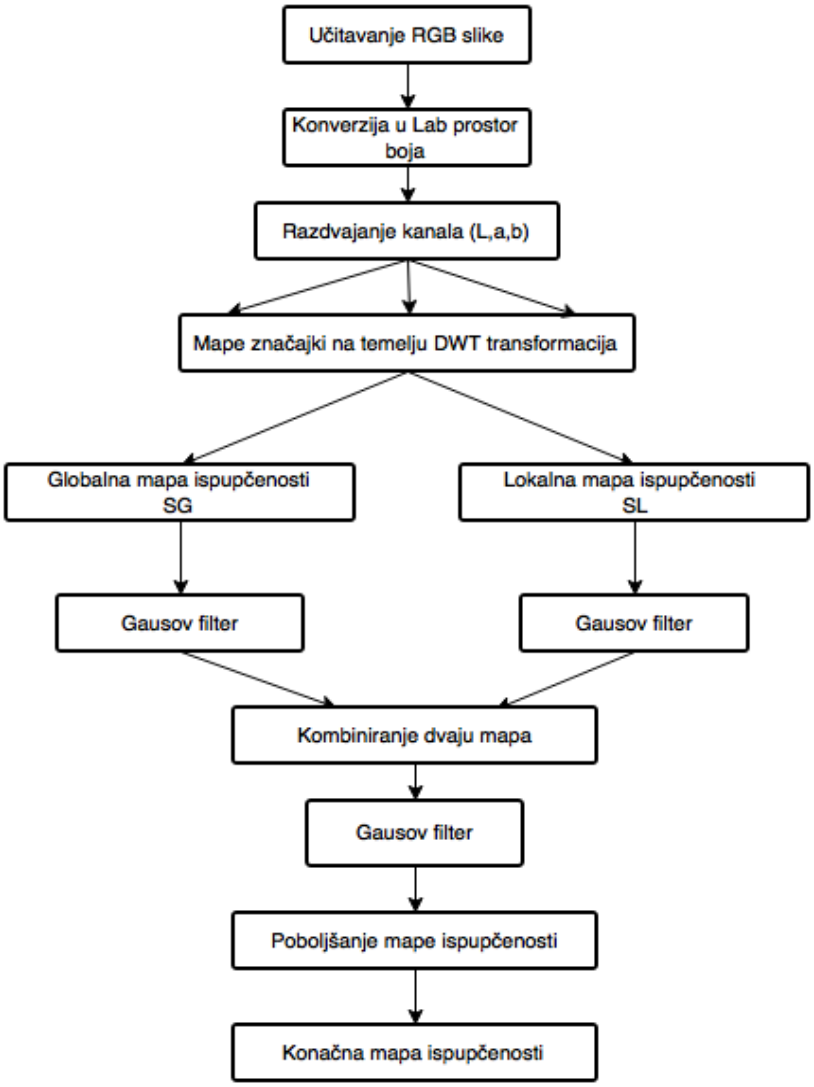
Algoritam	**FT (C++)**	**LC (C++)**	**SR (C++)**	**SW (Matlab)**
Vrijeme	5.4 s	1.21s	0.7s	600s

Prva tri algoritma pisana u C++ su iznimno brza s obzirom da je rađena detekcija na slikama rezolucije 4000 *3000 piksela. SW algoritam iako jako kvalitetan veoma je vremenski zahtjevan. Jedan od razlog je i to što je implementiran u Matlabu. U slijedećem poglavlju ispitati ću mogućnosti optimizacije ovog modela u C++ i CUDA programskom modelu.

6.3. Analiza mogućnosti optimizacije algoritma

U ovom poglavlju dati će se analiza mogućnosti optimiziranja modela detekcije ispupčenih objekata na temelju valića iz rada [24].

Sam algoritam se sastoji od nekoliko faza što je prikazano na slijedećem dijagramu.

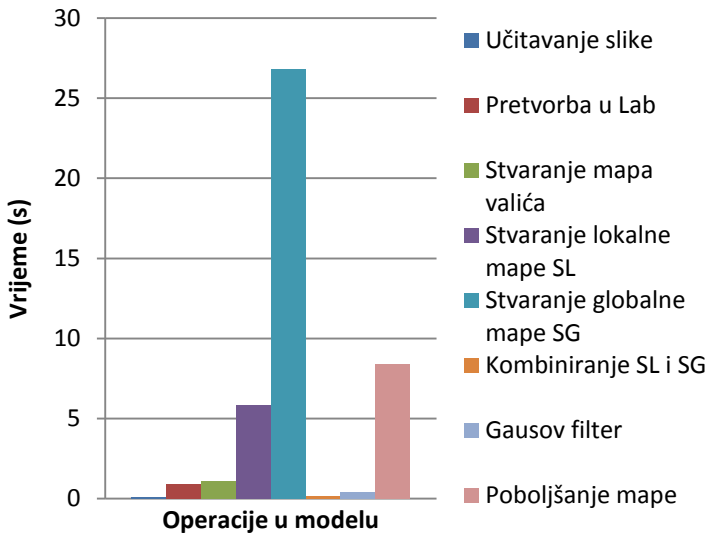


Slika 6.11. Faza izvedbe SW modela

Različite faze modela iziskuju i različito trajanje izvršavanja. U tablici 3 nalaze se rezultati izvedbe algoritma i svake pojedine faze mjerene u sekundama za sliku u boji (RGB) dimenzije 1000x750 piksela.

Tablica 3. Vrijeme trajanje svake faze SW modela

Operacija	Vrijeme (s)
Učitavanje slike	0.12
Pretvorba u Lab	0.88
Stvaranje mape valića	1.12
Stvaranje lokalnih značajki (SL)	5.8
Stvaranje globalnih značajki (SG)	26.79
Kombiniranje SL-a i SG-a	0.14
Gausovi filteri	0.38
Poboljašanje mape istaknutosti	8.38
Ukupno	**43.66**



Iz grafikona i tablice 3. može se iščitati da okosnicu algoritma čine operacije stvaranja mapa valića, lokalnih mapa značajki, globalnih mapa značajki i operacija koja poboljšava mapu istaknutosti. Operaciju poboljšavanja mapa istaknutosti možemo izostaviti u detekciji objekata za prirodne slike jer ne vrijedi princip centralnosti [72]što može dovesti do pogrešnih rezultata tako da ovu fazu nećemo razmatrati u ovoj analizi. Preostale tri faze čine 95% utroška vremena modela. U daljnjem radu ću testirati mogućnosti optimizacije koristeći C++ jezik, te CUDA programski model za rad na grafičkim procesorima.

6.3.1. Razmatranja CUDA programskog modela

Prije same optimizacije i rezultata pojedinih faza modela za detekciju istaknutih objekata razmotriti ću neke važnije postupke kod izrade CUDA programa. Algoritmi su testirani na testnom računalu sastavljeno od Intel Core i7-5930K procesora radnog takta 3.50GHz sa 32GB RAM memorije, te na grafičkom procesoru NVIDIA GF110GL Tesla C2050.

Grafički procesor za numerički računanje Tesla C2050 [73] predstavlja profesionalnu grafičku karticu zasnovanu Fermi arhitekturi [74] [75] koja je na tržište izbačena 2010. godine sa ciljem da donese performanse jednog klastera na desktop računalo za 20-tak puta nižu cijenu. Svojim specifikacijama i karakteristikama je zasigurno i uspjela. Sadrži 448 jezgri koje dostižu vršne performanse od 515 GFLOP/s nad podacima dvostruke preciznosti odnosno preko 1030 GFLOP/s za

podatke jednostruke preciznosti. Na ovaj model smješteno je 3,072 MB GDDR5 memorije, širina sabirnice 384 bit-a, a memorijska propusnost 144 GB/s. Osim glavne memorije sadrži i 64 KB L1 keš memorije po SM multiprocesoru i 768 KB L2 keš memorije. L1 memorija je konfigurabilna tako da može podržavati dijeljenu memoriju kao i operacije keširanja nad lokalnom i globalnom memorijom. 64 KB memorije se može konfigurirati na 48 KB dijeljene memorije i 16 KB L1 keš memorije u slučaju da se program intenzivno služi dijeljenom memorijom. S druge strane ako nam memorijski pristupi nisu unaprijed poznati možemo koristiti 48 KB L1 keš memorije sa 16 KB dijeljene memorije što uvelike može unaprijediti performanse u odnosu na direktni pristup memoriji.

Na veoma konkurentnom tržištu grafičkih kartica i industrije igara razdoblje od 5 godine se čini se kao vječnost budući je prožeto sa jako velikim brojem tehnoloških inovacija. Naime u 2012. godini Nvidia je izbacila grafičke kartice sa Kepler [76] arhitekturom, a 2014. godine Maxwel [77] arhitekturu. Svaka od novih arhitektura donosi poboljšanja u vidu broja CUDA jezgri, količine memorije kao i poboljšanja u samom programskom CUDA modelu koje rezultira jednostavnijim dizajniranjem i upravljanjem programa kao i boljoj iskoristivosti GPU-a. Primjerice Kepler arhitektura je donijela mogućnosti dinamičkog paralelizma, uspostavljanje direktne komunikacije i prijenos podataka između grafičkih kartica bez prethodne komunikacije sa CPU/radnom memorijom, mogućnost da više CPU procesora može istodobno pokrenuti posao na jednom GPU (HyperQ) itd [76]. S druge strane kada je riječ o Maxwell arhitekturi dizajneri su naglasak stavili na poboljšanje efikasnosti i smanjenju potrošnje samih uređaja. Grafički akcelerator Tesla K40 zasnovan na Kepler arhitekturi posjeduje 2880 jezgre, sadrži 12 GB GDDR5 memorije. Memorijska propusnost iznosi 288 GB/s, vršne performanse za operacije dvostruke preciznosti iznose 1680 GFLOP/s, a za operacije sa podacima jednostruke preciznosti čak 5040 GFLOP/s [78]. U odnosu na Teslu C2050 zasnovanu na Fermi arhitekturi to su impresivna poboljšanja, u nekim segmentima i za faktor 5.

Bez obzira na različite arhitekture pojedinih grafičkih akceleratora programi pisani u CUDA programskom modelu se ističu sa svojom skalabilnosti i međusobnom kompatibilnosti što znači da program pisan za Fermi arhitekturu će se bez potrebe za većim prilagodbama izvršiti i na novijim arhitekturama iskorištavajući veći broj jezgri na novijim modelima grafičkih procesora. Ipak za postizanje maksimalne učinkovitosti potrebno je voditi računa o detaljima svake pojedine arhitekture, a posebice o konfiguraciji *izvršnih niti* po blokovima za jezgrene funkcije te o količini upotrijebljene dijeljene memoriji po SM procesorima.

CUDA programski model je heterogeni model u kojem se koristi i CPU i GPU. U CUDA modelu, host se odnosi na CPU i radnu memoriju, dok se device odnosi na GPU i memoriju koja se nalazi na grafičkoj kartici. Kod koji izvršava host upravlja sa memorijom i na host i na device, te također

pokreće kernele ili jezgrene funkcije koji se izvršavaju na uređaju. Ove jezgrene funkcije izvode mnogo GPU izvršnih niti na paralelan način. Zbog svoje heterogene prirode CUDA programskog modela, tipična sekvenca operacija za CUDA C program je:

1. Deklaracija i alokacija host i device memorije
2. Inicijalizacija podataka na host-u
3. Transfer podataka iz host-a na device
4. Izvršavanje jedne ili više jezgrenih funkcija
5. Transfer rezultata sa device na host.

CUDA predstavlja proširenje C/C++ programskog jezika pri čijem dizajniranju se vodilo računa da bude što sličniji sa ANSI C standardom, a jedan od razloga je i brža krivulje učenja ovih proširenja. Tako su primjerice funkcija za alociranje memorije (*cudaMalloc*) na uređaju te funkcija za kopiranje i transfer podataka (*cudaMemcpy*) za uređaj slični standardnim C funkcijama (*malloc i memcpy*). Okosnicu proračuna čini jezgrena funkcija ili *kernel* u kojoj se definira onaj dio programskog koda koji će se i izvršavati na samom GPU.

U slijedećem odlomku se nalazi definicija funkcije pisane u C jeziku `transposeSek` te jezgrene funkcije `transposeNaiveCuda` koje vrši transponiranje kvadratne matrice s N redaka i N stupaca..

```
1.  void transposeSek(float*t_data,constfloat*data, int N)
2.  {
3.  intwidth=N;
4.
5.  for(int y=0; y<N;y++)
6.  for(int x=0; x<N;x++)
7.
8.      t_data[x*width+y]=data[y*width + x];
9.  }
```

```
1.  __global__ void transposeNaiveCuda(float*t_data,constfloat*data)
2.  {
3.  int x = blockIdx.x * TILE_DIM + threadIdx.x;
4.  int y = blockIdx.y * TILE_DIM + threadIdx.y;
5.  int width = gridDim.x * TILE_DIM;
6.
7.  for(int j =0; j < TILE_DIM; j+= BLOCK_ROWS)
8.      t_data[x*width +(y+j)]=data[(y+j)*width + x];
9.  }
```

Iako je riječ o jednostavnoj operaciji zamijene redaka sa stupcima na matrici *data,* ova implementacija nam može poslužiti kao vrlo ilustrativan primjer prilagodbi koje se moraju izvršiti na jezgrenim funkcijama. Ključna riječ `__global__` ispred jezgrene funkcije sugerira prevodiocu da je riječ o funkciji namijenjenoj za GPU. Primijećujemo da se u jezgrenoj funkciji za razliku od obične

funkcije *ne nalaze* dvije for petlje kojim iteriramo po elementima matrice *data*. Naime umjesto petlji koristimo ID vrijednosti izvršnih niti da predstavimo elemente u matrici. Pri pokretanju jezgrene funkcije na GPU cilj je pokrenuti što veći broj izvršnih niti, gdje svaka nit izvršava istu funkciju `transposeNaiveCuda`. Pokretanjem masivnog broja izvršnih niti osigurava se tzv. *skrivanje memorijske latencije* budući GPU po svojoj arhitekturi nema toliko razvijen upravljački dio kao što je slučaj kod CPU-a. Izvršne niti su organizirane u blokove, a ukupan broj niti po bloku ovisi o arhitekturi, pa je tako za Fermi arhitekturu broj niti ograničen na 512 izvršnih niti po jednom bloku. Kod Kepler i Maxwel arhitekture broj niti je ograničen na 1024 niti po jednom bloku. Broj blokova može biti maksimalno 65535 za *x* i *y* dimenziju. Svaka nit je jedinstveno određena u pojedinom bloku preko ID vrijednosti koju dohvaćamo preko strukture `threadIdx`. Kako nam je najčešće potrebno više blokova potrebno je dohvatiti i ID vrijednost bloka preko strukture `blockIdx`. Poznavanjem globalne varijable `TILE_DIM`, te vrijednosti ID niti i vrijednosti ID bloka može se jedinstveno odrediti svaka izvršna nit. Postupak određivanja jedinstvene ID vrijednosti izvršne niti na razini mreže po x i y dimenziji je navedeno u 1. i 2. liniji programskog odsječka funkcije `transposeNaiveCuda`.

Međutim sama konfiguriranje broja niti i blokova se ne vrši u jezgrenoj funkciji već prije samog poziva jezgrene funkcije. Za primjer kvadratne matrice veličine nx=1024 i ny=1024 .

```
const int nx=1024;const int ny =1024;
const intTILE_DIM=32;const intBLOCK_ROWS =8;
....
dim3dimGrid(nx/TILE_DIM, ny/TILE_DIM, 1);
dim3dimBlock(TILE_DIM, BLOCK_ROWS, 1);
```

U trodimenzionalnoj strukturi dim3 `dimBlock` se definira broj niti za jedan blok. U ovoj konfiguraciji broj niti za x dimenziju će biti 32, a za y će biti 8 što daje ukupno 256 niti po bloku. U strukturi dim3`dimGrid` se definira veličina mreže odnosno broj blokova pa tako za ovu veličinu matrice broj blokova za x i y dimenziju će biti po 32. Dimenzija z je oba slučaja suvišna pa se postavlja na vrijednost 1. Ukupno smo koristeći ovu konfiguraciju za matricu veličine 1024x1024 definirali 1024x256 niti što zapravo ne odgovara broju elemenata u matrici. U odnosu na broj elementa rezervirano je 4 puta manje izvršnih niti stoga će svaka izvršna nit biti zadužena za transponiranječetiri elementa matrice što je prikazano u liniji 7 i 8. Ovim pristupom donekle amortiziramo računanje ID niti te računanje širine matrice (1024) u jezgrenoj funkciji.

Kako i sam naziv funkcije `transposeNaiveCuda` sugerira, riječ je o naivnoj implementaciji jezgrene funkcije i ona ni blizu ne postiže performanse koje su teoretski moguće. Već sam istakao da GPU ne sadrži složenu upravljačku logiku kao na CPU što donekle prisiljava programere da koriste ručno upravljanu keš memoriju ili koriste mnogo bržu dijeljenu memoriju. U naivnoj implementaciji

podaci se nalaze smješteni u globalnoj GPU memoriji koja je i relativno brža od radne memorije na CPU, ali je za skoro dva reda veličine sporija od dijeljene memorije ili registara. Dijeljena memorija je ograničena na 48 KB za Fermi arhitekturu po SM-u te je dostupna svim izvršnim nitima u samo jednom bloku. Registri su vezani za samo jednu nit, a globalna memorija je dostupna svim nitima u svakom bloku. Osim relativno spore globalne memorije problem je i u samom pristupanju pojedinim elementima u matrici. Idealno bi bilo kada bi podaci kojima se pristupa bili relativno međusobno blizu odnosno da su *združeni (engl. coalesced)*. U funkciji `transposeNaiveCuda` podaci su združeni kod čitanja iz matrice *data*, ali pri upisu podataka u matricu *tdata* rade se preveliki koraci da se dohvati lokacija u memoriju kako bi se podatak upisao. Primjerice elemente koji se nalaze na mjestu 4, 5 i 6 u nizu *data* potrebno je spremiti na međusobno udaljene i raspršene lokacije 4,1028 i 2052. Ova raspršenost lokacija u memorijskom prostoru predstavlja faktor velike degradacije performansi kod GPU programa. Navedeni problemi su adresirani u implementaciji `transposeCoalesced`.

```
1.  __global__ void transposeCoalesced(float*t_data,constfloat*data)
2.  {
3.    __shared__ float tile[TILE_DIM][TILE_DIM+1];
4.
5.  int x = blockIdx.x * TILE_DIM + threadIdx.x;
6.  int y = blockIdx.y * TILE_DIM + threadIdx.y;
7.  int width = gridDim.x * TILE_DIM;
8.
9.  for(int j =0; j < TILE_DIM; j += BLOCK_ROWS)
10.       tile[threadIdx.y+j][threadIdx.x]= data[(y+j)*width + x];
11.
12.   __syncthreads();
13.
14.   x = blockIdx.y * TILE_DIM + threadIdx.x;
15.   y = blockIdx.x * TILE_DIM + threadIdx.y;
16.
17. for(int j =0; j < TILE_DIM; j += BLOCK_ROWS)
18.       t_data[(y+j)*width + x]= tile[threadIdx.x][threadIdx.y + j];
19. }
```

U prvom warp-u čitaju se susjedni podaci iz *data* u retke dijeljene memorije u varijabli tile. Nakon ponovnog izračunavanja indeksa polja, stupci u dijeljenoj memorije se zapisuju u susjedne lokacije u polju tdata. Budući niti zapisuju različite podatke u niz *tdat*a nego što čitaju iz niza data moramo koristiti barijernu sinkronizaciju __syncthreads(). Ovaj pristup nam daje veoma dobru propusnost algoritma te mnogo bolje peformanse u odnosu na naivnu implementaciju u što se možemo uvjeriti iz Tablice 4.

Tablica 4. Mjerenje memorijske propusnosti jezgrenih funkcija [79] u GB/s

Funkcija/Uređaj	Tesla M2050	Tesla K20c
transposeNaiveCuda()	18.8	55.3
transposeCoalesced()	99.5	144.3

Za mjeru performansi uzeta je memorijska propusnost budući operacija transponiranja ne radi nikakve proračune već je riječ o preslagivanju/prebacivanju podataka iz jednog u drugi niz. Metrika je izražena u GB/s, a eksperiment se izvodio na TeslaM2050 grafičkom procesoru zasnovanom na Fermi arhitekturi i Tesla K20c procesoru zasnovanu na novijoj Kepler arhitekturi. Teoretska vršna propusnost za TeslaM2050 iznosi 148 GB/s, a za Tesla K20c 208 GB/s. Iako nisu dostignute teoretski maksimalne moguće performanse vidljivo je da optimiziranje korištenjem dijeljenje memorije, združivanje podataka i sprječavanje konflikta drastično poboljšavaju performanse.
Detaljne informacije o CUDA programskom modelu, praktičnim programerskim savjetima se mogu pronaći u [80] [81].

6.3.2. Stvaranje mapa valića

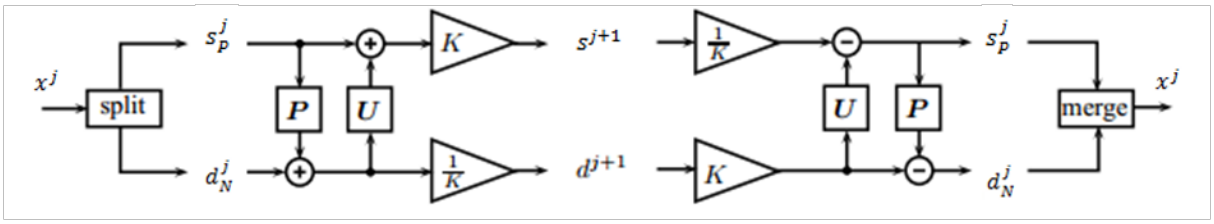
Mape valića ili Wavelet mape se kreiraju višerazinskom DWT transformacijom. DWT transformacija se tradicionalno implementira korištenjem konvolucije ili FIR bankefiltara [82]. Ovakve implementacije zahtijevaju veći broj aritmetičkih računanja i zahtijevaju veće spremnike što su svojstva koja nisu poželjna u aplikacijama obrade slike/videa pogotovo kada je riječ o aplikacijama koje zahtijevaju izvođenje u realnom vremenu ili koje se izvode na uređajima niske snage i potrošnje. Stoga je u relativno novije vrijeme predložena matematička formulacija koja se temelji na prostornoj konstrukciji valića i shemi faktorizacije [83]. Ovaj novi pristup se naziva transformacija valićima na temelju *lifting-a,* a zahtjeva manje računanja u odnosu na konvolucijski temeljene DWT [84]. Omogućuje manipulaciju sa podacima u mjestu i reducira memorijske među-ovisnosti.Postupak *lifting sheme* je slijedeći. Ulazni signal se dijeli u parne i neparne podsekvence koji se označavaju sa $\{s_i^0\}$ i $\{d_i^0\}$. Ove vrijednosti se dalje modificiraju uzastopnim korištenjem alternirajućih koraka predikcije (označeno sa *P*) i ažuriranja (označeno sa *U*). Operatori P i U su poznati kao koeficijenti Laurent-ovih polinoma nakon izvršenjaZ-transformacije [85]. U koraku predikcije, algoritam uzima neparne uzorke i oduzima ih od susjednih parnih uzoraka, te se formira predikcijska greška:

$$d_i^1 = d_i^0 - P(s_i^0 + s_{i+1}^0) \quad (1)$$

U koraku ažuriranja, modificirani susjedni neparni uzorci se zbrajaju i ažuriraju te se formira parna sekvenca$\{s_i^1\}$:

$$s_i^1 = s_i^0 + U(d_{i-1}^1 + d_i^1) \tag{2}$$

Broj koraka predikcije i ažuriranja ovisi o samoj faktorizaciji i obliku valića. Izlaz iz zadnje faze ažuriranja, $\{s_i^j\}$ je nisko-propusni izlaz DWT filtra, a slično tome zadnji korak predikcije $\{d_i^j\}$ je visoko-propusni izlaz filtera. Nakon faze ažuriranja izlaz$\{s_i^j\}$se množi sa faktorom skaliranja K, a $\{d_i^j\}$ sa faktorom skaliranja 1/K. Generalno se može reći da se nakon transformacije valićima signal dijeli u nisko propusne i visoko propusne kanale. Ovaj postupak se može ilustrirati na slici 6.12.



Slika 6.12: Jedna faza lifting sheme. Lijeva strana: lifting prema naprijed, Desna strana: invrerzni lifting

U radu je [83] detaljnije je opisan postupak faktorizacije wavelet transformacije kroz *lifting* korake.

Jedan od najkorištenijih valića odnosno filtera je CDF 9/7 valić (Cohen-Daubechies-Feauveau) koji se koristi u JPEG 2000 standardu za potrebe kompresiju sa gubicima. Koeficijenti faktorizacije ovog filtera iznose redom: $\alpha = -1.586134342, \beta = -0.05298011854,$ $\gamma = 0.8829110762$, $\delta = 0.4435068522$. Koeficijent skaliranja iznosi $k = 1.149604398$.Ovaj valić ima 2 faze predikcije i 2 faze ažuriranja, odnosno L=2.

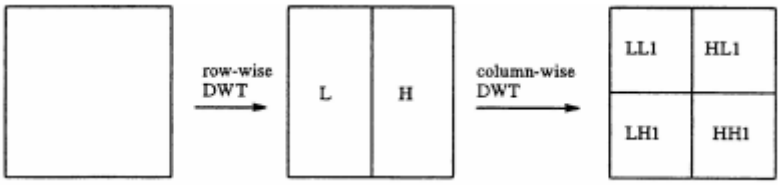
Postavljamo parametre Lifting sheme: $L = 2, P(0) \coloneqq \alpha, P(1) \coloneqq \gamma, U(0) = \beta, U(1) = \delta, K = k$.U slijedećem odlomku se nalazi pseudokod izvršavanja DWT (lijevo) transformacije i inverzne DWT (desno) transformacije.

DWT (signal x, lifting shema,signal s, signal d)	**IDWT (signal s, signal d, lifting shema,signal x)**
Razdvajanje: dijelimo signal na parni i neparni dio $s_P^j[n] = x[2n];$ $d_N^j[n] = x[2n+1];$	**Reskaliranje:** skalira svaki uzorak sa faktorom K ili 1/K. $s_P^j[n] \leftarrow s_P^j[n]/K;$ $d_N^j[n] \leftarrow d_N^j[n] * K;$
Ponavljaj za m< L	**Ponavljaj za m< L**
Predikcija: kombiniramo nekoliko parnih uzoraka i dodajemo neparnom dijelu $d_N^j[n] \leftarrow d_N^j[n] - P[m](s_P^j[n] + s_P^j[n+1]);$	**Ažuriranje:**kombiniramo nekoliko neparnih uzoraka i dodajemo parnom dijelu $s_N^j[n] \leftarrow s_N^j[n] - U[m]\ (d_N^j[n-1] + d_N^j[n]);$
Ažuriranje:kombiniramo nekoliko neparnih	**Predikcija:** kombiniramo nekoliko parnih uzoraka i

uzoraka i dodajemo parnom dijelu $s_N^j[n] \leftarrow s_N^j[n] + U[m]\ (d_N^j[n-1] + d_N^j[n]);$ **Kraj ponavljanja** **Skaliranje:** skalira svaki uzorak sa faktorom K ili 1/K. $s_P^j[n] = s_P^j[n] * K;$ $d_N^j[n] = d_N^j[n]\ /K;$	dodajemo neparnom dijelu $d_N^j[n] \leftarrow d_N^j[n] + P[m]\left(s_P^j[n] + s_P^j[n+1]\right);$ **Kraj ponavljanja** **Spajanje:** spajamo parni i neparni dio u izvorni signal x $x[2n] = s_P^j[n];$ $x[2n+1] = d_N^j[n];$

Ovaj pseudokod vrijedi za jednu razinu dekompozicije, a u slučaju višerazinske dekompozicije postupak se rekurzivno ponavlja nad dijelom signala koji predstavlja aproksimaciju izvornog signala (s_P^j). Inverzna transformacija(sinteza signala) je analogna transformaciji prema naprijed (analizi) uz razliku što se kod sinteze vrše inverzne operacije u obrnutom redoslijedu izvođenja.

U odnosu na jednodimenzionalne signale,2D signali (kao što su slike) uobičajeno se transformiraju u obje dimenzije. Prvo se primijeni DWT transformacija na sve retke pojedinačno, a potom na sve stupce čime dobivamo četiri *pod-banda*-a, LL,HL,LH i HH (Slika 6.13).



Slika 6.13. Prva razina DWT transformacija 2D signala

Pod-kanal LL predstavlja aproksimaciju izvornog signala i može poslužiti za daljnju transformaciju ukoliko želimo višerazinsku dekompoziciju valića (slika 6.14).

Slika 6.14. Druga i treća razina DWT transformacije 2D signala

Na slijedećoj ilustraciji može se vidjeti pseudokod algoritma računanja DWT–a za 2D signal na osnovu *lifting* sheme za prvu razinu. Koristi se funkcija za transformaciju jednodimenzionalnog signala iz prethodnog primjera na recima matrice. Nakon toga slijedi transponiranje izlazni podband-ova, te transformacija nad stupcima matrice.

2D_DWT (2D signal x,lifting shema)

Transformacija po retcima: *Za svaki pojedinačni redak slike izvršava se DWT*
for *i<retci*
data:=x[i];
DWT (signal data, lifting shema, signal s, signal d);
L[i]=s; H[i]=d;
i++;
end;
Transponiranje: *Izlazne signale L i H transponiramo kako bi izvršili DWT po stupcima*
transpose (L, LT);
transpose (H, HT);
Transformacija po stupcima: *Za svaki pojedinačni stupac slike izvršava se DWT nad LT i HT signalom*
for *i<stupci*
dataL:=LT[i];
dataH:=HT[i];
DWT (dataL, lift shema, signal sl, signal dl);
DWT (dataH, lift shema, signal sh, signal dh);
LLT[i]=sl; LHT[i]=dl;
HLT[i]=sh; HHT[i]dh;
i++;
end;
Transponiranje: *Nad izlazima LLT, LHT, HLT i HHT vršimo re-transponiranje*
transpose (LLT, LL);
transpose (LHT, LH);
transpose (HLT, HL);
transpose (HHT, HH);
KRAJ

Isto kao i u radu [24] korišten jeDaubechies valić koji sadrži pet *momenata iščezavanja* (*engl. vanishing moments*) ili *db5*.

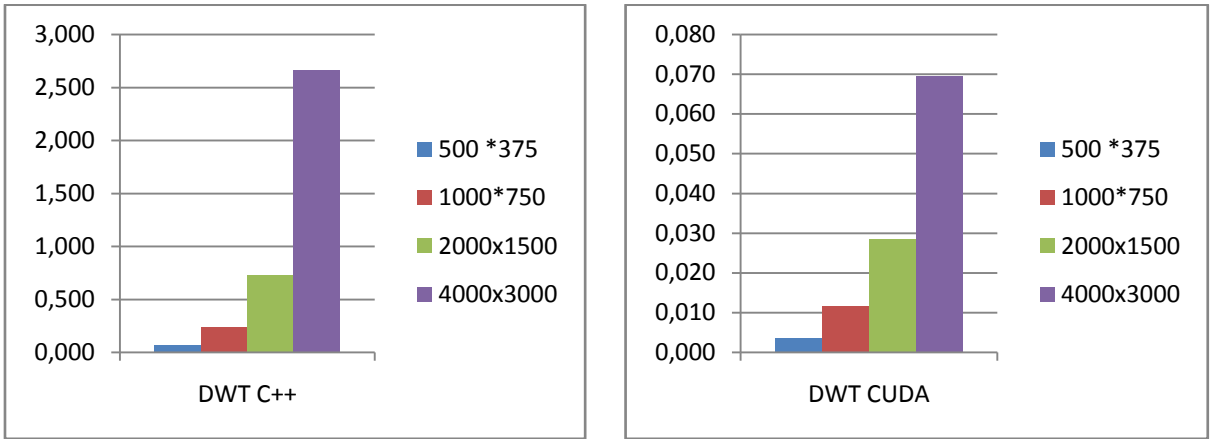
U slijedećoj tablici 5 se nalaze rezultate DWT transformacije naivne sekvencijalne implementacije u C++ programskom jeziku, kao i rezultati implementacije prilagođene za GPU. Pri mjerenju za CUDA model uračunat je prijenos podataka iz radne memorije u globalnu memorije, te prijenos rezultata iz globalne memorije u radnu. Korišteno je testno računalo Intel Core i7-5930K CPU 3.50GHz sa 32GB RAM memorije, te NVIDIA GF110GL Tesla C2050 grafičkom karticom sa 3GB GDDR memorije.

Tablica 5. Vrijeme trajanja DWT i inverzne DWT transformacije za 4 najveće razine, te ukupno trajanje za sve razine u sekundama

Dimenzije	DWT (C++)	DWT(Cuda)	Ubrzanje	IDWT (C++)	IDWT(Cuda)	Ubrzanje
500x375	0.067	0.004	16.75	0.066	0.004	16.50
1000x750	0.238	0.012	19.83	0.216	0.011	19.64
2000x1500	0.731	0.029	25.21	0.707	0.027	26.19
4000x3000	2.662	0.069	38.58	2.657	0.065	40.88
Ukupno	3.698	0.114	32.44	3.646	0.107	34.07

Osobina DWT transformacije je u tome što se može rekurzivno izvoditi za svaku novu razinu. Primjerice za sliku dimenzije 4000x3000 piksela transformacijom dobivamo 4 pod-banda koji su 4 puta manji od izvorne slike i imaju dimenzije 2000x1500. Pod-band LL predstavlja aproksimaciju izvorne slike, na kome se rekurzivno vrše daljnje DWT transformacije. Slika dimenzije 4000x3000 će imati 11 razina dekompozicije, a svaka nova razina sadrži 4 puta manju količinu podataka kada je riječ o 2D signalu. Trajanje izvršavanja će u najvećoj mjeri ovisiti o vremenu transformacije prve razine tj. nad najvećim signalom, u ovom slučaju nad slikom dimenzije 4000x3000. Na osnovi tablice 5 te grafa 2 dolazimo do zaključka da će trajanje višerazinske DWT transformacije nad 2D signalom biti manje od dvostrukog vremena trajanja izvođenja DWT-a prve razine.

Graf 2. DWT transformacija 4 razine i vrijeme izvođenja u sekundama za C++(lijevo) i CUDA(desno) izvedbu



Ako usporedimo sekvencijalnu C++ i CUDA implementaciju višerazinske DWT transformacije može se primijetiti da je ubrzanje veoma značajno te za sliku rezolucije 4000x3000 iznosi približno 39 puta. Slično vrijedi i za inverznu DWT transformaciju. Računanja su vrednovana samo za jedan kanal, a u modelu SW transformacija valićima se vrši nad tri kanala, pa stoga ukupne rezultate treba uvećati za faktor tri.

6.3.3. Stvaranje mape lokalnih značajki (SL)

Mape valića ili koeficijenata predstavljaju detalje slike za različite veličine slike. Iz ovih koeficijenata se može stvoriti veći broj mapa značajki preko inverznu DWT transformacije. Broj mapa valića ovisi pretežito o broju razina DWT transformacije odnosno o veličine ulaznog signala/slike. Primjerice za sliku u boji veličine 1000x750 piksela inverznu transformaciju odnosno rekonstrukciju slike možemo započeti od najdublje devete razine pa sve do prve razine čime dobivamo jednu mapu

značajki. Međutim, rekonstrukciju možemo započeti od osme razine i propagirati do početne prve razine čime dobivamo drugu mapu značajki. Ovaj postupak se ponavlja, a za svaki kanal dobivamo po devet mapa značajki pri čemu je svaka mapa rezolucije/veličine kao izvorna slika. Dakle ukupno ćemo dobiti 27 mapa značajki, količina podataka će iznositi 1000x750x27. Za slike velikih dimenzija riječ je o relativno velikom memorijskom zauzeću. U tablici 6 se nalaze podaci o broju razina i broju značajki za određene dimenzije slike te očekivano memorijsko zauzeće.

Tablici 6. Broj mapa značajki i memorijsko zauzeće

Dimenzije slike	Broj razina	Broj kanala	Broj mapa značajki	Ukupni broj piksela (pikseli * razine*kanali)	Ukupno memorijsko zauzeće 1 RGB slike
500 x 375	8	3	24	4.5 milijuna piksela	18.00 MB
750 x 562	9	3	27	11.38 milijuna piksela	45.52 MB
1000x750	9	3	27	20.25 milijuna piksela	81.00 MB
1500x1125	10	3	30	50.62 milijuna piksela	202.50 MB
2000x1500	10	3	30	90 milijuna piksela	360 MB
3000x2000	11	3	33	198 milijuna piksela	792 MB
4000*3000	11	3	33	396 milijuna piksela	1584 MB

Mapa lokalne istaknutosti se dobiva relativno jednostavno kombiniranjem svih mapa značajki na razini odgovarajućih piksela, a kao kriterij za odabir ispupčenog piksela koristi se najveća vrijednost značajki za određeni piksel. U tablici 7 se nalazi proračun za stvaranja mapa značajki i lokalne mape istaknutosti kombinirano.

Tablica 7. Vrijeme izračuna lokalne mape istaknutosti u sekundama

Dimenzije slike	Lokalna mapa (C++)	Lokalna mapa (CUDA)	Ubrzanje
500x375	2.03	0.14	15.04
1000x750	7.61	0.43	17.62
2000x1500	29.67	1.26	23.55
4000x3000	109.380	3.21	34.07

6.3.4. Stvaranje globalne mape ispupčenosti

Mapa globalne ispupčenosti se također računa na mapa značajkama dobivenih iz prethodno opisane rekonstrukcije preko inverzne DWT-a transformacije.

Globalna mapa ispupčenosti se računa kao normalna gustoća vjerojatnosti u multivarijatnom prostoru po slijedećoj jednadžbi.

$$p\big(f(x,y)\big) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} * e^{\left(-\frac{1}{2}(f(x,y)-\mu)\right)^{T}\Sigma^{-1}(\mathrm{f(x,y)}-\mu)} \quad (1)$$

Gdje je

$$\Sigma = \mathrm{E}[(f(x,y) - \mu)(f(x,y) - \mu)^T] \quad (2)$$

Vektor μ predstavlja srednju vrijednost svake mape značajki odnosno $\mu = E[f]$; *T* je operacija transponiranja; IzrazΣ u (2) je *n x n* matrica kovarijance; n=3 x N predstavlja broj vektora značajki,a uključuje 3 kanala boje i N mapa značajki za svaki kanal boja; $|\Sigma|$ je determinanta matrice kovarijance.

Jednadžbe (1) i (2) se zornije mogu prikazati preko slijedećeg pseudo-koda.

PDF (DATA data)
Ulaz u u funkciju je skup značajki data
Računamo srednju vrijednost značajki:
muDATA = mean(DATA);
Računamo kovarijancu na svim podacima:
Kov= cov(DATA);
Računamo determinantu kovarijance :
detKov=det(Kov);
Računamo inverz kovarijance
invKov= pinv(Kov);
Veličina L predstavlja broj uzorka, a D broj značajki za svaki piksel
[L, D] = size(DATA);
Odredimo konstantni dio iz jednadžbe 4:
*konstDio = (2*PI)^(D/2) * (detKov)^(1/2) ;*
U petljiračunamo viševarijabilnu normalnu gustoću distribuciju vjerojatnosti (PDF)
for *i = 1:L*
v = DATA(i,:);
*pdf(i) = exp(-(1/2) * (v-muDATA)*invKov*(v-muDATA)') ;*
pdf(i)=pdf(i) / konstDio;
end
Kraj

U navedenom pseudo-kodu vidljivi su izračuni za srednje vrijednosti, kovarijancu, inverz kovarijance, determinantu kovarijance, kao i petlju u kojoj se računa sama gustoća vjerojatnosti. U C++ implementaciji navedenih operacije koristio sam Armadillo biblioteku [86] za linearnu algebru koja podržava BLAS, LaPACK [87] standardizirane alate u matričnom računanju, a za CUDA izvedbu koristio sam cuBLAS biblioteku [88].

Sa stajališta proračuna najviše vremena je potrebno za računanje kovarijance, te za izvedbu petlje u kojoj se računa gustoća vjerojatnosti po jednadžbi (1). U slijedećoj tablici se nalaze izvedbe računanja proračuna za globalnu mapu istaknutosti.

Tablica 8. Vrijeme izračuna za globalne značajke u sekundama

Dimenzije slike	Globalna mapa (C++)	Globalna mapa (CUDA)	Ubrzanje
500x375	0.436	0.096	4.54
1000x750	1.415	0.152	9.31
2000x1500	6.339	0.468	13.54
4000x3000	25.550	1.884	13.56

Za sliku u boji veličine 4000x3000 piksela CUDA izvedba je brža oko 13.5 puta u odnosu na C++ izvedbu.

6.4. Očekivanja i rezultati

Na osnovu prethodnih pokazatelja, testiranja i simulacija može se pretpostaviti vrijeme izvođenja cjelokupnog modela za sliku u boji dimenzije 1000x750, kao i za sliku u boji veličine 4000x3000 piksela. Preko 90% modela SW čini stvaranje mapa značajki pomoću valića, stvaranje lokalnih značajki ispupčenosti i stvaranje globalne mape ispupčenosti. Ostale faze modela kao što su učitavanje slike, razdvajanje kanala, propuštanje kroz Gausove filtre nisu toliko značajni i vremenski zahtjevni pa su izostavljani iz *okvirne analize*, a optimizaciju svih faza modela ću ostaviti za budući rad. Također izostavljena je faza poboljšanja mape ispupčenosti budući je korisnija kod tzv. „subjekt slika“, a u prirodnim slikama može dovesti do pogrešnih rezultata.

Tablica 9. Rezultati izvedbe vremenski najintenzivnijih dijelova SW modela u sekundama

Proračun modela SW	C++ (1000x750)	CUDA (1000x750)	C++ (4000x3000)	CUDA (4000*3000)	Ubrzanje (4000x3000)
Mape valića	0.89	0.02	10.94	0.29	37.72
Lokalna mapa (SL)	7.61	0.43	109.38	3.21	34.07
Globalna mapa (SG)	1.42	0.15	25.55	1.88	13.56
Ukupno	**9.92**	**0.60**	**145.87**	**5.38**	**27.09**

Za sliku u boji veličine 12 Mpx vrijeme izvedbe testiranih faza iznosi 5.38 sekundi što predstavlja ubrzanje od preko 27 puta u odnosu na C++ izvedbu. Procjena očekivano trajanje izvedbe cjelokupnog modela uz korištenje grafičke kartice kao akceleratora iznosi između 6 i 7 sekundi čime se ovaj algoritam približava primjeni za one probleme koji zahtijevaju razumno vrijeme izvođenja. Međutim treba naglasiti da je ova implementacija u nekim svojim fazama veoma naivna te da nisu iskorišteni puni potencijali kako opće-namjenskog procesora tako i grafičkog procesora. Naivna implementacija DWT transformacije za C++ nije koristila vektorizaciju podataka, a nije primijenjena

niti kakva tehnika paralelizacije za višejezgrene procesore korištenjem nekih od standardnih alata (C11 Threads, OpenMP, Cilk, TBB). Također, u članku [89] autori navode da su uspjeli izvršiti tro-razinsku DWT transformaciju na GPU za 2.05 ms na slici veličine 1920x1080 što je 8 puta brže u odnosu na implementaciju (17 ms) primijenjenu u ovom kvalifikacijskom radu.

U daljnjem radu pokušati ću se približiti ovim rezultatima koristeći prednosti registara, brze priručne i dijeljenje memorije na GPU,te s druge strane u potpunosti iskoristiti potencijale standardnih višejezgrenih procesora.

.

Zaključak

U ovom kvalifikacijskom doktorskom radu obrađen je pregled literature veoma zanimljivog biološki inspiriranog područja računalnog vida koji je vezan za detekciju istaknutih objekata (*engl. saliency detection*). Ovi modeli se najčešće koriste u detektiranju objekata koji su vizualno upadljivi, a u svom radu pokušavaju simulirati princip tzv. *rang predstavljanja.* Ovaj princip po kojem funkcionira selektivna vizualna pažnja u ljudi se sastoji od vizualnih mapa koje uključuju elementarne značajke kao što su orijentacija rubova, boja, disparitet i smjer pokreta. Ovi modeli se najčešće koriste u kompleksnim sustavima računalnog vida kao početna, pripremna faza za daljnju obradu pa moraju biti relativno brzi. Međutim postoje i modeli koji se iskazuju kvalitetom, ali njihova brzina izvođenja predstavlja ograničavajući faktor u primjeni za aplikacije za rad u realnom ili blizu realnom vremenu. Jedna od takvih aplikacija je i detekcija objekata u prirodnim slikama za potrebe pretrage i spašavanje. Nakon kvalitativne analize nekoliko modela na fotografijama prikupljenih iz bespilotne letjelice kao najuspješniji se pokazao algoritam opisan u radu [24]. Međutim sa stajališta brzine izvođenja pokazao se kao neadekvatan za prirodne slike velike rezolucije.

Stoga sam u drugom dijelu opisao mogućnosti paralelizacije na hibridnim arhitekturama kao što je korištenje grafičkog procesora kao koprocesor glavnom procesoru. Ovakva arhitektura zaokuplja akademsku zajednicu već duži niz godina budući je lako dostupna i povoljna, a osigurava iznimne performanse. Pokazalo se da su najznačajniji dijelovi gore navedenog algoritma veoma pogodni za paralelizaciju na grafičkim procesorima uz CUDA programski model te su postignuta ubrzanja i do 40 puta u odnosu na sekvencijalnu izvedbu na C++ programskom jeziku. Ustanovljeno je da za testnu konfiguraciju brzina izvođenja za sliku rezolucije 4000x3000 iznosi oko 5 sekundi. U daljnjem radu istražit ću daljnje korake optimizacije ovog algoritma budući nisu iskorišteni svi potencijali grafičkog procesora kao i integraciji svih dijelova modela u dobivanju cjelovitog modela optimiziranog za grafičke procesore.

Bibliografija

[1] J. M. Wolfe and T. S. Horowitz., "What attributes guide the deployment of visual attention and how do they do it?," *Nature Reviews Neuroscience*, pp. 1-7, 2004.

[2] C. Kennard, and M. Husain S. K. Mannan, "The role of visual salience in directing eye movements in visual object agnosia," *Current biology*, vol. 19, no. 6, pp. 247-248, 2009.

[3] C. Koch, and E. Niebur L. Itti, "A model of saliency-based visual attention for rapid scene analysis," *IEEE TPAMI*, vol. 20, no. 11, pp. 1254-1259, 1998.

[4] K. Ngan, M. Li, and H. Zhang J. Han, "Unsupervised extraction of visual attention objects in color images," *IEEE TCSV*, vol. 16, no. 1, pp. 141–145, 2006.

[5] B. Ko and J. Nam, "Object-of-interest image segmentation based on human attention and semantic region clustering," *J Opt Soc Am*, vol. 23, no. 10, p. 2462, 2006.

[6] D. Walther, C. Koch, and P. Perona. U. Rutishauser, "Is bottom-up attention useful for object recognition?," in *CVPR*, 2004, pp. 37-44.

[7] A. Skodras, and T. Ebrahimi C. Christopoulos, "The JPEG2000 still image coding system: an overview," *IEEE Trans. on Consumer Electronics*, vol. 46, no. 4, pp. 1103-1127, 2002.

[8] M.-M. Cheng, P. Tan, A. Shamir, and S.-M Hu. T. Chen, "Sketch2photo: Internet image montage. ," *ACM TOG*, vol. 28, no. 5, pp. 1-10, 2009.

[9] M.-M. Cheng, S.-M. Hu, and R. R. Martin G.-X. Zhang, "A shape-preserving approach to image resizing," *Comput.Graph. Forum*, vol. 28, no. 7, pp. 1897-1906, 2009.

[10] G-X Zhang, M-M Cheng, "Global Contrast based Salient Region Detection".

[11] J. Sun, N. Zheng, X. Tang, and H.-Y. Shum T. Liu, "Learning to detect salient object," in *CVPR*, 2007, pp. 1-8.

[12] C. Koch and S. Ullman, "“Shifts in selective visual attention: towards the underlying neural circuitry," *Matters of Intelligence*, pp. 115-141, 1987.

[13] X. Hou and L. Zhang, "Saliency detection: A spectral residual approach," in *CVPR*, 2007, pp. 1-8.

[14] J. Harel, and C. Koch X. Hou, "Image signature: Highlighting sparse salient regions," *IEEE TPAMI*, vol. 34, no. 1, 2012.

[15] X. Jiang, Y. Sun, and J. Wang H.-D. Cheng, "Color image segmentation: advances and prospects," *Pattern Recognition*, vol. 34, no. 12, pp. 2259-2281, 2001.

[16] Z. Zhang, W.-Y. Lin, and P. H. S. Torr M.-M. Cheng, "“BING: Binarized normed gradients for objectness estimation at 300fps," in *CVPR*, 2014.

[17] J. Warrell, and P. H. Torr, Z. Zhang, "Proposal generation for object detection using cascaded ranking svms," in *CVPR*, 2011, pp. 1497-1504.

[18] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *CVPR*, vol. 1, pp. 886-893, 2005.

[19] R. B. Girshick, D. McAllester, and D. Ramanan P. F. Felzenszwalb, "Object detection with discriminatively trained part based models," *IEEE TPAMI*, pp. 1627-1645, 2010.

[20] S.M. Culhane, W.Y.K Wai, Y.H. Lai, N. Davis, F. Nuflo J.K. Tsotsos, "Modelling visual attention via selective tuning," *Artifical Intelligence*, vol. 78, no. 1-2, pp. 507-545, 1995.

[21] X.Zhang,B.Wandell S. Engel, "Colour tuning in human visual cortex measured with functional magnetic resonance imaging," *Nature*, vol. 388, no. 6637, pp. 68-71, July 1997.

[22] X. Hou and L. Zhang, "Saliency detection: A spectral residual approach," *IEEE CVPR*, pp. 1-8, 2007.

[23] S. Hemami, F. Estrada, and S. Susstrunk R. Achanta, "Frequency-tuned salient region detection," *CVPR*, 2009.

[24] Weisi Lin Nevrez İmamoğlu, "A Saliency Detection Model Using Low-Level Features Based on Wavelet Transform," *Multimedia, IEEE Transactions*, vol. 15, no. 1, pp. 96 - 105, January 2013.

[25] N. Sebe, M. S. Lew, E. Loupias, and T. S. Huang Q. Tian, "Image retrieval using wavelet-based salient points," *J. Electron. Imag*, vol. 10, no. 4, pp. 835–849, 2001.

[26] M. Vanrell, X. Otazu, and C. A. Parraga N. Murray, "Saliency estimation using a non-parametric low-level vision model," *Proc. IEEE Int. Conf. Comput. Vision and Pattern Recognition*, 2011.

[27] R. J. E. Merry, "Wavelet Theory and Application: A Literature Study ," Eindhoven Univ Eindhoven, The Netherlands: Eindhoven Univ., Eindhoven, The Netherlands, DCT 2005.53 2005.

[28] A. Torralba, M. S. Castelhano, and J. M. Henderson A. Oliva, "Topdown control of visual attention in object detection," *Proc. IEEE Int. Conf. Image Processing*, vol. vol.1, pp. 253–256, 2003.

[29] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, 4th, Ed. London, UK: Academic/Elsevier, 2009.

[30] L. Zelnik-Manor, and A. Tal S. Goferman, "Context-aware saliency detection," *IEEE TPAMI*, vol. 34, no. 10, 2012.

[31] M. Gleicher F. Liu, "Region enhanced scale-invariant saliency detection," *Proceedings of the ICME*, pp. 1477–1480, 2006.

[32] F. Estrada, P. Wils, and S. Susstrunk R. Achanta, "Salient region detection and segmentation," *Comp. Vis. Sys.*, 2008.

[33] N. RICHE, M. MANCAS, B. GOSSELIN, T. DUTOIT J. LEROY, "SuperRare: an Object-oriented Saliency Algorithm Based on Superpixels Rarity," in *IEEE International Conference on Robotics and Automation (ICRA 2014)*, Hong Kong, 2014.

[34] A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Su R. Achanta, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. vol. 34, no. no. 11, pp. 2274–2282, 2012.

[35] M. Mancas, M. Duvinage, M. Mibulumukini, B. Gosselin, T. Dutoit N. Riche, "A multi-scale rarity-based saliency detection," *Sig. Proc.: Image Comm*, vol. 28, no. 6, pp. 642–658, 2013.

[36] L. Itti and C. Koch, "A comparison of feature combination strategies for saliency-based visual attention systems," *Journal of Electronic Imaging*, vol. 10, pp. 161–169, 1999.

[37] N. J. Mitra, X. Huang, P. H. S. Torr, and S.-M. Hu M.-M. Cheng, "Global contrast based salient region detection," *IEEE TPAMI*, 2014.

[38] D. Huttenlocher P. Felzenszwalb, "Efficient graph-based image segmentation," *IJCV*, vol. vol. 59, no. no. 2, pp. 167–181, 2004.

[39] J. Wang, Z. Yuan, T. Liu, N. Zheng, and S. Li H. Jiang, "Automatic salient object segmentation based on context and shape prior," *BMVC,* pp. pp. 1–12, 2011.

[40] N. Bruce and J. Tsotsos, "Saliency, attention, and visual search: An information theoretic approach," *Journal of Vision*, vol. 9, 2009.

[41] M. Tong, T. Marks, H. Shan, and G. Cottrell L. Zhang, "SUN:A bayesian framework for saliency using natural statistics," *Journal of Vision*, vol. 8, no. 7, pp. 1-20, 2008.

[42] C. Koch, and P. Perona J. Harel, "Graph-based visual saliency," *NIPS*, pp. 545-552, 2007.

[43] Y. Zhai and M. Shah, "Visual attention detection in video sequences using spatiotemporal cues," *ACM Multimedia*, 2006.

[44] Ming–Ming Cheng, Huaizu Jiang, Jia Li Ali Borji, "Salient Object Detection: A Survey," *arXiv eprint*, 2014.

[45] V. Kolmogorov, and A. Blake C. Rother, ""“GrabCut”– Interactive foreground extraction using iterated graph cuts," *ACM TOG*, vol. 23, no. 3, pp. 309–314, 2004.

[46] N. J. Mitra, X. Huang, and S.-M. Hu M.-M. Cheng, "SalientShape:Group Saliency in Image Collections," *The Visual Computer*, 2013.

[47] M. Vanrell, X. Otazu, and C. A. Parraga N. Murray, "Saliency estimation using a non-parametric low-level vision model," *IEEE CVPR*, pp. 433-440, 2011.

[48] R. Achanta and S. Susstrunk, "Saliency detection using maximum symmetric surround," *IEEE ICIP*, pp. 2653-2656, 2010.

[49] J. Kannala, M. Salo, and J. Heikkila E. Rahtu, "Segmenting salient objects from images and videos," *ECCV*, 2010.

[50] H. Seo and P. Milanfar, "Static and space-time visual saliency detection by self-resemblance," *Journal of vision*, vol. 9, 2009.

[51] C. Wu, J. Miao, L. Qing, and Y. Fu L. Duan, "Visual saliency detection by spatially weighted dissimilarity," *IEEE CVPR*, pp. 473-480, 2011.

[52] T. Narumi, R. Yokota, K. Yasuoka, K. Nitadori, and M. Taiji T. Hamada, "42 tflops hierarchical n-body simulations on GPUs with applications in both astrophysics and turbulence," in *SC*, 2009.

[53] A. Di Serio and M. B. Ibanez, "Evaluation of a nearest-neighbor load balancing strategy for parallel molecular simulations in mpi environment," in *PVM/MPI*, 2002, pp. 226-233.

[54] J. J. Tapia and R. D'Souza, "Data-parallel algorithms for large-scale real-time simulation of the cellular Potts model on graphics processing units," in *IEEE International Conference on Systems Man and Cybernetics*, 2009, pp. 1411–1418.

[55] C. H. C. Leung, W. Rahayu, and S. Goel D. Taniar, "High-Performance Parallel Database Processing and Grid Databases," in *Wiley Series on Parallel and Distributed Computing*, 2008.

[56] A. Maximo, L. Velho, H. Hnaidi, and M.-P. Cani A. Bernhardt, "Real-time terrain modeling using cpu-GPU coupled computation," in *In Proceedings of the 2011 24th SIBGRAPI Conference on Graphics, Patterns and Images, SIBGRAPI '11*, 2011, pp. 64-71.

[57] Y. Matias and U. Vishkin, "On parallel hashing and integer sorting," in *In Michael Paterson editor, Automata, Languages and Programming*. Heidelberg: Springer Berlin , 1990.

[58] A. Koch, and W. Straßer S. Pabst, "Fast and scalable CPU/GPU collision detection for rigid and deformable surfaces," in *Computer Graphics Forum*, 2010, pp. 1605-1612.

[59] N. K. Nousis, and M. N. Vrahatis V. P. Plagianakos, "Locating and computing in parallel all the simple roots of special functions using pvm," *J. Comput. Appl. Math.*, vol. 133, pp. 545-554, August 2001.

[60] Y. Shiloach and U. Vishkin, "An o(log n) parallel connectivity algorithm," *J. Algorithms*, vol. 3, no. 1, pp. 57–67, 1982.

[61] J. H. Hoover, and W. L. Ruzzo R. Greenlaw, *Limits to Parallel Computation: P-Completeness*

Theory. USA: Oxford University Press, 1995.

[62] H. A. Peelle, "To teach Newton's square root algorithm," *SIGAPL APL Quote Quad*, vol. 5, no. 4, pp. 48-50, 1974.

[63] C. P. Breshears, *The Art of Concurrency – A Thread Monkey's Guide to Writing Parallel Applications*.: O'Reilly, 2009.

[64] Guy Blelloch, "Programming Parallel Algorithms," *Communications of the ACM*, vol. 39, no. 3, pp. 85–97.

[65] J. L. Gustafson, "Fixed time, tiered memory, and superlinear speedup," in *In Proceedings of the Fifth Distributed Memory Computing Conference (DMCC5*, 1990.

[66] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing," in *spring joint computer conference*, New York, 1967, pp. 483-485.

[67] P. E. Ross, "Why cpu frequency stalled," *IEEE Spectr.*, pp. 72-72, 2008.

[68] Sonya Geis. (2007, September) Rescue Crews Find No Sign Of Missing Adventurer. [Online]. http://www.washingtonpost.com/wp-dyn/content/article/2007/09/05/AR2007090502116.html

[69] Simon Hradecky. (2009, June) Crash: Air France A332 over Atlantic on Jun 1st 2009, aircraft impacted ocean. [Online]. http://avherald.com/h?article=41a81ef1/0022&opt=0

[70] Sokalski Jan and Toby P. Breckon, "Automatic Salient Object Detection In UAV Imagery," in *25th International UAV Systems Conference*, 2010, pp. 11.1-11.12.

[71] S. Hemami, F. Estrada, and S. Susstrunk R. Achanta, "Frequency tuned salient region detection," *IEEE CVPR*, 2009.

[72] F. Durand, and A. Torralba T. Judd, "A benchmark of computationa lmodels of saliency to predict human fixations," 2012.

[73] Nvidia. (2010, July) TESLA™ C2050 / C2070 GPU Computing Processor Supercomputing at 1/10th the Cost. [Online]. http://www.nvidia.com/docs/IO/43395/NV_DS_Tesla_C2050_C2070_jul10_lores.pdf

[74] Peter N. Glaskowsky. (2009, Rujan) NVIDIA's Fermi: The First Complete GPU Computing Architecture. [Online]. http://www.nvidia.com/content/pdf/fermi_white_papers/p.glaskowsky_nvidia's_fermi-the_first_complete_gpu_architecture.pdf

[75] NVIDIA. (2011) NVIDIA's Next Generation CUDA Compute Architecture: Fermi, Whitepaper. [Online]. www.nvidia.com/object/IO_89570.html

[76] Nvidia. (2012) NVIDIA's Next Generation CUDA Compute Architecture: Kepler GK110,Whitepaper. [Online]. https://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf

[77] Nvidia. (2014) Nvidia GTX 980, Featuring Maxwell, The most advanced GPU ever made. Whitepaper. [Online]. http://international.download.nvidia.com/geforce-com/international/pdfs/GeForce_GTX_980_Whitepaper_FINAL.PDF

[78] Nvidia. (2013, Novembar) TESLA K40 GPU Accelerator. [Online]. http://www.nvidia.com/content/PDF/kepler/Tesla-K40-PCIe-Passive-Board-Spec-BD-06902-001_v05.pdf

[79] Mark Harris. (2013, February) An Efficient Matrix Transpose in CUDA C/C++. [Online]. http://devblogs.nvidia.com/parallelforall/efficient-matrix-transpose-cuda-cc/

[80] Kirk & Hwu, *Programming Massively Parallel Processors,A Hands-on Approach*, 2nd ed., Morgan Kaufmann, Ed.: Elsevier, 2012.

[81] Nvidia. (2015, September) CUDA C PROGRAMMING GUIDE. [Online]. http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf

[82] ACHARYA,C. CHAKRABARTI T., "A Survey on Lifting-based Discrete Wavelet Transform Architectures," *Journal of VLSI Signal Processing 42*, pp. 321–339, 2006.

[83] I. Daubechies and W. Sweldens, ""Factoring Wavelet Transforms into Lifting Schemes"," *The J. of Fourier Analysis and Applications*, vol. 4, pp. 247–269, 1998.

[84] W. Swelden, ""The Lifting Scheme:A Custom-Design Construction of Biorthogonal Wavelets," *Applied and Computationaltional Harmonic Analysis"*, vol. 3, no. 15, pp. 186–200, 1996.

[85] Rajmic P. Prusa Z., "Real-Time lifting wavelet transform algorithm," *Elektrorevue*, vol. 2, no. 3, SEPTEMBER 2011.

[86] Conrad Sanderson, "Armadillo: An Open Source C++ Linear Algebra Library for Fast Prototyping and Computationally Intensive Experiments," Technical Report, NICTA, Australia 2010.

[87] Bai Z., Bischof C., Blackford S., Demmel J., Dongarra J. Anderson E. (1999) LAPACK Users' Guide. [Online]. http://www.netlib.org/lapack/lug/

[88] Nvidia. (2013, July) CUBLAS LIBRARY User Guide. [Online]. http://docs.nvidia.com/cuda/cublas/index.html

[89] Andrei C. Jalba, and Jos B.T.M. Roerdink Wladimir J. van der Laan, "Accelerating Wavelet Lifting on Graphics Hardware Using CUDA," *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS*, vol. VOL. 22, no. NO. 1, JANUARY 2011.

[90] P. Krahenbuhl, Y. Pritch, and A. Hornung F. Perazzi, "Contrast based filtering for salient region detection," *CVPR*, pp. 733-740, 2012.

[91] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph based based image segmentation," pp. 167-181, 2004.

[92] L. Xu, J. Shi, and J. Jia Q. Yan, "Hierarchical saliency detection," *CVPR*, pp. 1155-1162, 2013.

[93] H. Lu, L. Zhang, X. Ruan, and M.-H. Yang X. Li, "Saliency detection via dense and sparse reconstruction," *ICCV*, 2013.

[94] Y. Li, C. Shen, A. R. Dick, and A. van den Hengel, X. Li, "Contextual textual hypergraph modeling for salient object detection," *ICCV*, pp. 3328-3335, 2013.

PRILOG

Skraćenice

FT	Frequency-tuned salient region detection
AIM	Saliency, attention, and visual search: An information theoretic approach.
MSS	Saliency detection using maximum symmetric surround
SEG	Segmenting salient objects from images and videos
SeR	Static and space-time visual saliency detection by self-resemblance
SWD	Visual saliency detection by spatially weighted dissimilarity
SUN	SUN: A bayesian framework for saliency using natural statistics
IM	Saliency estimation using a non-parametric low-level vision model
IT	A model of saliency-based visual attention for rapid scene analysis
GB	Graph-based visual saliency
SR	Saliency detection: A spectral residual approach
CA	Context-aware saliency detection
LC	Visual attention detection in video sequences using spatiotemporal cues
AC	Salient region detection and segmentation
CB	Automatic salient object segmentation based on context and shape prior
LP	Learning to predict where humans look