Pregled novijih algoritama za stereo viziju

Tisaj, Matea

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

PREGLED NOVIJIH ALGORITAMA ZA STEREO VIZIJU

Diplomski rad

Matea Tisaj

Osijek, 2018.

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 26.03.2018.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Matea Tisaj
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina	D822 R, 27.09.2017.
OIB studenta:	72080802770
Mentor:	Doc.dr.sc. Emmanuel Karlo Nyarko
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Doc.dr.sc. Damir Filko
Član Povjerenstva:	Prof.dr.sc. Robert Cupec
Naslov diplomskog rada:	Pregled novijih algoritama za stereo viziju
Znanstvena grana rada:	**Obradba informacija (zn. polje računarstvo)**
Zadatak diplomskog rada:	U klasičnoj stereo viziji se pomoću dvije kamere dobiva dva pogleda na istu scenu te mogućnost dobivanja informacije o dubini scene. Najveći problem kod stereo vizije je određivanje korespodencija između točaka slike snimljene jednom kamerom i točaka slike snimjene drugom kamerom, odnosno pronalazak slike dispariteta čije vrijednosti predstavljaju razlike u koordinatama projekcija iste točke u prostoru na lijevu odnosno desnu sliku. Potrebno je napraviti pregled i usporedbu nekoliko novijih stereo algoritama za generiranje slike dispariteta.
Prijedlog ocjene pismenog dijela ispita (diplomskog	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	26.03.2018.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT

IZJAVA O ORIGINALNOSTI RADA

Osijek, 18.04.2018.

Ime i prezime studenta:	Matea Tisaj
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D822 R, 27.09.2017.
Ephorus podudaranje [%]:	3

Ovom izjavom izjavljujem da je rad pod nazivom**: Pregled novijih algoritama za stereo viziju**

izrađen pod vodstvom mentora Doc.dr.sc. Emmanuel Karlo Nyarko

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Robotika, a samim time i robotski vid, jedno je od područja tehnologije koje se danas najviše i najbrže razvija. Koristeći robotski vid, inženjeri pokušavaju na razne načine doći do što više informacija iz dobivenih slika kako bi roboti lakše obavljali svoje zadatke. Primjerice u industriji, robotski vid može pomoći robotskom manipulatoru da lakše detektira predmete od interesa u nestrukturiranoj okolini. Slike dobivene stereo vizijom omogućavaju dobivanje informacije o njezinoj trećoj dimenziji – dubini. Kako bi se dobila udaljenost pojedinih točaka na slici od kamere, računa se slika dispariteta čije vrijednosti predstavljaju razlike u koordinatama projekcija iste točke u prostoru na lijevu, odnosno desnu sliku. Upravo najveći problem stereo vizije leži u tome što je računalu teško pridružiti točke s jedne slike točkama koje pripadaju drugoj slici i prema kojima se računa slika dispariteta [1].

Cilj diplomskog rada je usporediti nekoliko novijih algoritama za stereo viziju kako bi se vidjelo u kojim uvjetima daju najtočnije podatke. Algoritmi su pisani u programskom jeziku C++, a korištena je biblioteka OpenCV.

Poglavlje 2 donosi pregled korištenih tehnologija, dok poglavlje 3 detaljno opisuje stereo viziju i probleme pri određivanju slike dispariteta, a poglavlje 4 donosi pregled odabranih novijih algoritama, te je u poglavlju 5 opisan proces predobrade podataka, primjena algoritama i njihova usporedba nakon provedenih testova.

1.1. Zadatak diplomskog rada

Zadatak diplomskog rada jest napraviti pregled novijih algoritama za stereo viziju i usporediti ih pomoću generiranih slika dispariteta s obzirom na brzinu izvođenja i točnost. U teorijskom dijelu rada potrebno je navesti i opisati algoritme za izračun slika dispariteta, te način usporedbe dobivenih podatka.

2. PREGLED KORIŠTENIH TEHNOLOGIJA

U ovom diplomskom radu je korišten Visual Sudio Enterprise 2015 . Visual Studio je proizvod tvrtke Microsoft. Visual Studio je integrirano razvojno okruženje (eng. *Integrated Development Environment*) za razvoj aplikacija na Android operativnom sustavu, iOS-u i Windows operativnom sustavu. Koristi se i za programiranje mrežnih (eng. *web*) aplikacija, te rješenja u oblaku (eng. *cloud*). Omogućava korištenje mnogih dodataka (eng. *plug-in*) koji olakšavaju izradu aplikacija u programa. Postoji više inačica koje se uglavnom naplaćuju, kao što je Visual Studio Professional ili Visual Studio Enterprise, međutim postoji i besplatna verzija razvojnog okruženja pod nazivom Visual Studio Community [2].

Budući da se tema diplomskog rada temelji na robotskom, odnosno računalnom vidu, za rad nad slikama korištena je biblioteka OpenCV. OpenCV je biblioteka otvorenog koda (eng. *open-source*) koja sadrži nekoliko stotina algoritama za računalni vid. Omogućava manipulaciju nad slikama i videom. OpenCV je napisan u programskom jeziku C++ koji je opisan dalje u poglavlju, a može se koristiti s drugim programskim jezicima kao što su Python ili Java. Neke funkcionalnosti OpenCV biblioteke su [3, 4]:

- procesiranje slika - omogućava filtriranje i transformaciju slika kao što je na primjer rezanje slike (eng. *crop*)
- video analiza - upravljanje videom tako što se može upravljati pozadinom, detektirati predmete koji se kreću itd.
- kalibraciju 3D kamera - sadrži jednostavne algoritme za scene iz više pogleda, kalibraciju pomoću jednog ili više pogleda i algoritme stereo korespondencije što je izrazito važno za ovaj diplomski rad
- detekcija objekata - objekti se prepoznaju na osnovi unaprijed određenih klasa
- GPU akceleracija – moguće je ubrzati izvođenje algoritama tako da se kod izvršava na grafičkoj kartici

Za izradu diplomskog rada korištena verzija biblioteke je OpenCV 2.4.13.6.

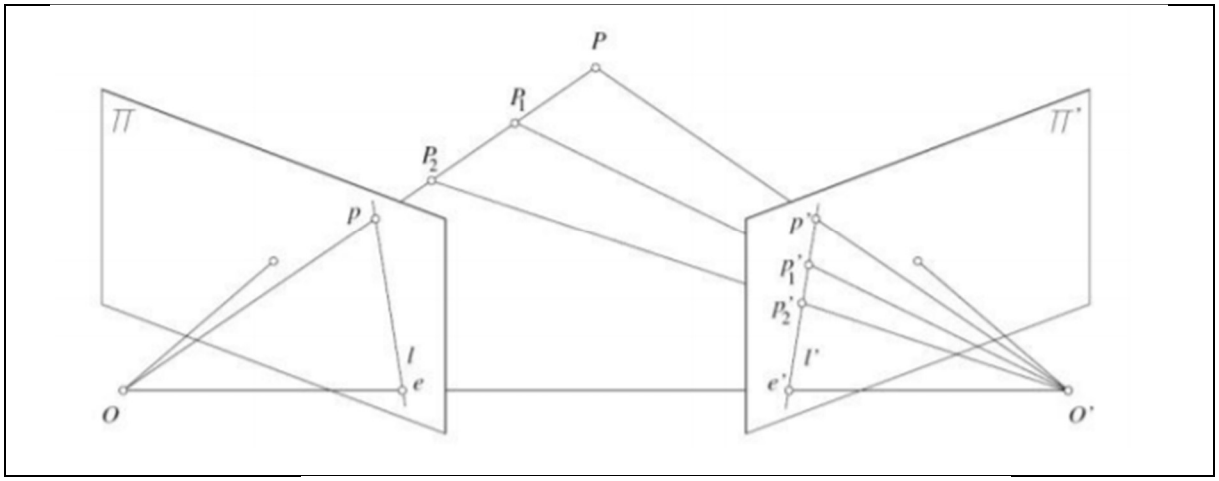
C++ jedan je od najčešće korištenih programskih jezika u svijetu. Objektno je orijentiran i koristi se za različite vrste programa kao što su izrade jednostavnih skripta do izrade aplikacija ili video igara. Kompatibilan je i s programskim jezikom C, drugim riječima kod napisan u

programskom jeziku C može se pokrenuti s C++ interpreterom budući da C++ može koristiti biblioteke napisane za programski jezik C [5]. Standardiziran je prema ISO standardu 1998. godine. C++ je utjecao na razvoj mnogih programskih jezika kao što su C# i Java [6]. Algoritmi korišteni u ovom diplomskom radu napisani su u C++ programskom jeziku.

Programski jezik Python je napravljen početkom 90-ih godina 20. stoljeća u Nizozemskoj. Otvorenog je koda i visoke razine (eng. *high level programming language*) [7, 8]. Python se lako nadovezuje na druge programske jezike kao što su C, C++ ili Java, te omogućava laku migraciju projekata na drugi programski jezik odnosno prelazak na korištenje programskog jezika Python. Izrazito je jednostavan za uporabu i omogućeno je korištenje na Windows, Mac OS X i Unix operacijskom sustavu [9]. U ovom diplomskom radu programski jezik Python je korišten za predobradu podataka, te analizu i vizualizaciju prikupljenih rezultata. Korištena je inačica 3.4.4.1.

3. STEREO VIZIJA

Stereo vizija dio je robotskog vida kod kojeg se, uz pomoć analize dvaju ili više slika koje gledaju istu scenu iz različitih kutova dobiva udaljenost točaka sa slike, kao što je vidljivo na slici 3.1. Stereo vizija opisana je epipolarnom geomerijom. Točka *p* na ravnini π predstavlja projekciju točke *P* iz prostora na lijevu sliku lijeve kamere, dok točke *p'*, *p1'* i *p2'* na ravnini π' predstavljaju moguće projekcije točke *P* na desnu kameru. Određivanje pravca na kojem leži projekcija točke, tj. epipolarna linij, znatno skraćuje vrijeme pretraživanja i pridruživanja točaka jedne slike drugoj [10]. Epipolarne linije, tj. epipolarno ograničenje moguće je odrediti određivanjem esencijalne $\boldsymbol{E}$ i fundamentalne matrice $\boldsymbol{F}$ prema formulama (3-1) i (3-2) [11].



Slika 3.1. Položaj dvaju kamera koje gledaju istu scenu [10]

$$\boldsymbol{E} = \left[{}^{R}\boldsymbol{t}_{L}\right]_{x} \cdot \boldsymbol{R}\left({}^{R}\boldsymbol{\phi}_{L}\right) \tag{3-1}$$

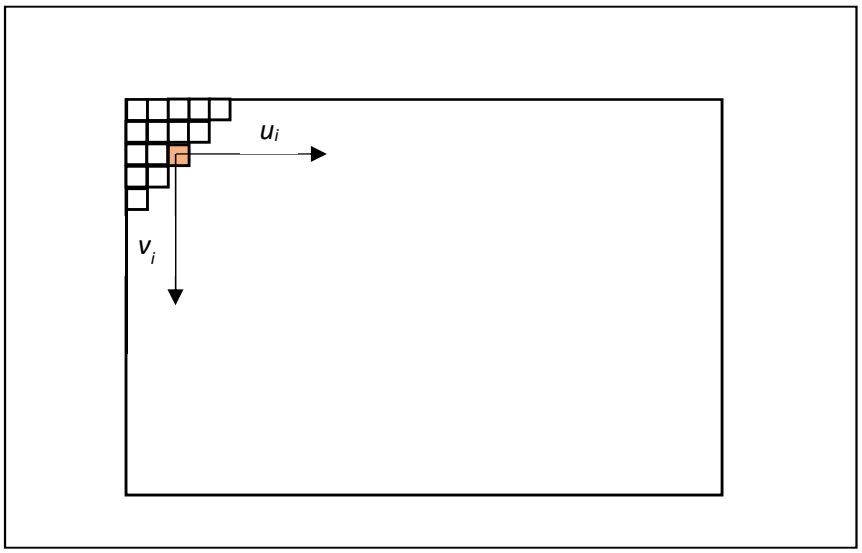
$$\boldsymbol{F} = \boldsymbol{P}_{R}^{-T}\left[{}^{R}\boldsymbol{t}_{L}\right]_{x} \cdot \boldsymbol{R}\left({}^{R}\boldsymbol{\phi}_{L}\right) \cdot \boldsymbol{P}_{L}^{-1} \quad , \tag{3-2}$$

gdje je $\boldsymbol{P_L}$ projekcijska matrica na prvoj kameri, a $\boldsymbol{P_R}$ projekcijska matrica na drugoj kameri. $\boldsymbol{t}$ je vektor translacije, a $\boldsymbol{R({}^{R}\phi_L)}$ je matica odnosa koordinatnog sustava $\boldsymbol{S_L}$ u odnosnu na sustav $\boldsymbol{S_R}$, odnosno njihova rotacijska matrica [11].

Za određivanje fundamentalne matrice potrebno znati ekstrinsične i intrinsične parametre kamera kojim su slike slikane, točnije njihov međusobni položaj, podatke o lećama kao što su njihov fokus i distorzije. Proces određivanja tih parametara se naziva kalibracija. Dobiveni parametri koriste se kasnije prilikom obrade slike. Kalibracija se provodi tako da se kamerom snime jednostavni objekti poznatih dimenzija te se prema tome izračunaju parametri. Primjer

kalibracije je slikanje iz različitih uglova niza crno bijelih kvadrata kao što su na šahovskoj ploči. Dimenzije i broj kvadrata na slici moraju biti poznati. U nekim slučajevima nije moguće napraviti kalibraciju. Osim podataka o kamerama i njihovom položaju za rekonstrukciju 3D scene, potrebno je združiti točke, odnosno piksele s jedne slike točkama (pikselima) druge slike, te nakon toga izračunati sliku dispariteta [12].

Ako slika lijeve kamere ima koordinatni sustav s koordinatama u_l i v_l , a desna slika ima svoj koordinatni sustav u kojem su koordinate označene s u_r i v_r , slika dispariteta se odredi tako da se izračunaju apsolutne udaljenosti između pridruženih piksela jedne slike, $p(u_l$ i $v_l)$ u odnosu na drugu $p'(u_r$ i $v_r)$. Na slici 3.2. je prikazan primjer koordinatnog sustava slike [11].



Slika 3.2. - Primjer koordinatnog sustava slike

Budući da pomak pridruženih piksela može biti i po *u* i *v* osi, sliku je potrebno ispraviti, odnosno prikazati tako da postoji samo pomak po jednoj osi. Takav par slika zove se ispravljeni par slika, a za njega vrijedi [11]:

- Intrinsični parametri kamera su jednaki
- Osi sustava lijeve i desne slike su paralelne
- Optički centri kamera udaljeni su samo po *x* osi

U slučaju ispravljenog para slika puno je lakše izračunati slike dispariteta, međutim slike korištene u ovom diplomskom radu nisu ispravljene kako bi se algoritmi testirali u uvjetima koji su realni i češći [11]. Algoritme koji daju dobre rezultate kod slika koje nisu ispravljene i ne ovise o tome da su kamere pričvršćene za nepomičnu konstrukciju lakše je implementirati na sustave kao što su autonomna vozila.

Glavni problem u stereo viziji predstavlja upravo određivanje slike dispariteta, odnosno pridruživanja točaka s jedne slike točkama druge slike. Problem se javlja kod slika koje imaju velike površine bez teksture kao što su, na primjer, zidovi u prostoriji. Zahtjevi tehnologije su takvi da podatke o dubini slike treba dostaviti u što kraćem vremenu, gotovo u stvarnom vremenu kako bi ih se moglo koristiti [12].

4. ALGORITMI U STEREO VIZIJI

Znanstveno - istraživački tim s Middlebury College napravio je razvojno okruženje i program za evaluiranje algoritama za generiranje slika dispariteta [13]. Rezultati su objavljeni na njihovim stranicama, te su prema tim rezultatima odabrana tri algoritama koji su testirani u diplomskom radu, a njihovi su rezultati prikazani u idućem poglavlju.

Algoritmi stereo vizije se uglavnom sastoje od četiri koraka čiji se redoslijed mijenja ovisno o algoritmu. Ta četiri koraka su:

- izračun funkcije cilja (eng. *cost function*)
- agregacija vrijednosti dobivenih funkcijom cilja
- izračunavanje i optimizacija dispariteta
- poboljšanje dobivenih slika dispariteta

Algoritmi mogu biti lokalni i globalni. Lokalni algoritmi se uglavnom bave izračunom funkcije cilja i agregacijom vrijednosti funkcija cilja, a zatim se jednostavnim pridruživanjem piksela s najmanjom vrijednosti funkcije cilja dobije slika dispariteta. Za određivanje iznosa funkcije cilja može se na primjer koristiti normalizirana unakrsna korelacija. Kod optimizacije uzimaju se u obzir lokalni minimumi što može rezultirati outlierima u dobivenoj slici dispariteta. Ograničenje ovog pristupa je što su pikseli jedinstveno pridruženi samo na jednoj slici, dok na drugoj jedan piksel može biti sparen s više točaka. Kod globalnih metoda se stavlja naglasak na izračunavanje slika dispariteta, a agregacija vrijednosti funkcije cilja se često preskače. Globalne metode su najčešće formulirane u okviru minimiziranja energije, te nakon što je definirana globalna energija slika traže se lokalni minimumi. Lokalne metode daju manje točne rezultate od globalnih metoda, ali globalne metode su značajno sporije od lokalnih [14].

4.1. ELAS

ELAS (eng. *Efficient Large-scale Stereo Matching*) je C++ biblioteka za računanje slika dispariteta na ispravljenim slikama sivih tonova (eng. *greyscale*). Za korištenje ove datoteke potrebno je imati CMAKE [15].

Glavna prednost ovog algoritma je brzina i to što radi sa slikama visoke rezolucije. Kod ovakvih je algoritama brzina izrazito važna kako bi se mogli primjenjivati kod sustava u stvarnom vremenu (eng. *real-time systems*). Rad sa slikama visoke rezolucije omogućava smanjenje greške koja nastaje kod izračunavanja slike dispariteta i koja kvadratno raste s udaljenošću. [16]

ELAS je generativni probabilistički model za sparivanje točaka u stereo viziji. Algoritam koristi mnogo malih prozora prilikom procesiranja slike kako bi se smanjio broj točaka koje nisu mogle biti jasno sparene. Prije svega se, uz pomoć prozora koji prolazi slikom, odrede "pomoćne točke" (eng. *support points*), tj. točke parova koji su robusno spojeni. Te točke obično predstavljaju jedinstvene teksture na slici koje je lako lokalizirati. Delaunay triangulacijom se nad "pomoćnim točkama", prema kojima se sparuju ostale točke na slici, stvori 2D mreža. Kako bi se osigurala robusnost algoritma, provjerava se jesu li parovi spareni i s lijeva na desno i s desna na lijevo. Razlika udaljenosti točaka mora biti unutar određene tolerancije inače se taj par eliminira. Zbog pomičnog se prozora, rub slike dispariteta određuje tako da se tim točkama dodijeli vrijednost najbližeg susjeda. Lijeva slika se koristi kao referentna slika. "Pomoćne točke" se određuju uz pomoć Sobelovog filtera dimenzija 3x3 koji imaju fiksni pomak od 5 piksela [16]. Prema Middlebury Stereo Vision popisu algoritama, algoritam je jako dobro pozicioniran s obzirom na vrijeme, dok je s obzirom na srednju pogrešku mjerenja prosječan [13]. Algoritam izvodi do 300 MDE/s (eng. *million disparity evaluations per second*) na procesoru s jednom jezgrom [16]. Algoritam testiran u diplomskom radu preuzet je sa stranice koje je navedena pod literaturom [17].

4.2. SGM – Census

SGM - Census ili Semi-Global Matching algoritam je algoritam za pridruživanje točaka para slika dobivenih stereo kamerom. Algoritam dobiva rezultate bolje nego algoritmi koji koriste lokalne metode, točnije, rezultati su približni onima koji koriste globalne metode, ali vrijeme izvođenja algoritma je puno brže od algoritama s globalnim metodama. Kako bi rezultati bili dobri uzeta su u obzir tri slučaja kod kojih se javljaju problemi pri izračunu slika dispariteta:

- točno određivanje rubova objekata
- promjene u osvjetljenju
- slike velikih dimenzija

Slike moraju biti ispravljene, mora im biti poznata epipolarna geometrija, te se često koristi za 3D rekonstrukciju [1]. Postupak se temelji na sparivanju piksela koristeći uzajamne informacije (eng. *Mutual information*). Uzajamne informacije su se pokazale korisne jer su robusne na promjene u osvjetljenju, a čak i u rješavanju problema reflektirajućih površina. Postavljena je globalna funkcija cilja koja je optimizirana na način da prolazi u 8 smjerova preko slike. Budući da se na slikama javljaju diskonuiteti koji se na slici dispariteta pojavljuju kao glatke površine korištene su metode kako bi se ta pogreška smanjila.

Uzajamne informacije definirane su entropijama prve i druge slike, te entropijom njihove zajedničke slike. Iterativnim postupkom se izračunava slika dispariteta uzimajući u obzir uzajamne informacije. Svakom iteracijom izračunata slika dispariteta je bolja, a i postoji više uzajamnih informacija.

Nakon izračuna funkcije cilja vrši se provjera piksel po piksel kako bi se utvrdili lokalni minimumi. Budući da inicijalni lokalni minimumi mogu biti netočno spareni zbog šuma na slici penaliziraju se promjene kod dispariteta susjednih piksela. Na taj način riješen je i problem diskonuiteta [17].

Zbog jednostavnih operacija koje koristi moguće ju je implementirati i na FPGA [1]. Metoda korištena u ovom radu je napisana u C++ programskom jeziku, a izvršena je na operacijskom sustavu Ubuntu 16.04. Detalji o algoritmu mogu se naći na [17], a kod algoritma preuzet je s [19].

4.3. mc-cnn

Metoda mc-cnn (eng. *Matching cost with a convolutional neural network*) koristi se za sparivanje točaka ispravljenog para slika, te za izračun njihove slike dispariteta. Fokus je stavljen na izračunavanje dobre funkcije cilja sparenih slika.

Konvolucijske neuronske mreže su istrenirane na parovima slika gdje je poznat njihov pravi disparitet. Izlaz iz konvolucijske neuronske mreže se koristi kao inicijalizacija slike dispariteta. Budući da to nije dovoljno kako bi se dobili dobri rezultati pogotovo u područjima gdje manjka teksture ili gdje su objekti na slici zaklonjeni koriste dodatne metode za obradu dobivene slike dispariteta. Metode korištene kod ovog algoritma su unakrsna agregacija, SGM, provjera konzistentnosti lijeve slike u odnosu na desnu, poboljšanje podpiksela (eng. *subpixel*), median i bilateralni filter. Poboljšavanjem podpiksela te izračunavanjem mediana i primjenom bilateralnog filtera izračunava se konačna slika dispariteta. Od svih metoda korištenih za obradu dobivene slike dispariteta, metoda SGM najviše utječe na validacijsku pogrešku [20].

Kod konvolucijskih neuronskih mreža korištene se dvije arhitekture - brzu (eng. *fast*) i sporu (eng. *slow*). Prva arhitektura je značajno brža, ali i daje lošije rezultate od one sporije. Analogno tomu sporija arhitektura daje točnije rezultate, ali zahtjeva i veću računalnu moć jer je kompleksnija. Ulaz u konvolucijsku neuronsku mrežu je par slika, dok je izlaz iz mreže rezultat njihove sličnosti - slika dispariteta. Obje arhitekture su istrenirane kako bi izlučile značajke pojedinih slika i spremile ih u vektor značajki (eng. *feature vector*). Slika dispariteta se izračunava upravo prema vektoru značajki, a ne prema intenzitetu slika, kako je slučaj u nekim algoritmima. Zbog toga algoritam je robustan na osvjetljenje. Brza arhitektura koristi unaprijed određene mjere sličnosti kako bi usporedila vektore dok sporija arhitektura pokušava naučiti što je slično između vektora.

Predstavljene su dvije mreže, jedna je naučena na KITTI skupu podataka, a druga na Middlebury skupu podataka [20]. U ovom diplomskom radu testirane mreže su:

- KITTI fast (brza arhitektura)
- KITTI slow (spora, precizna arhitektura)
- Middlebury fast (brza arhitektura)
- Middlebury slow (spora, precizna arhitektura)

Kod algoritma preuzet je s [21], dok se više informacija o algoritmu može pronaći pod [20].

5. PRIMJENA ALGORITAMA I USPOREDBA REZULTATA

Testirane su gotove implementacije kratko objašnjenih algoritama iz prethodnog poglavlja. Prije samog testiranja potrebno je napraviti predobradu podataka kako bi ih se prilagodilo različitim ulazima spomenutih algoritama. Testiranje algoritama je izvršeno na dva operacijska sustava - Windows 10 Pro i Ubuntu 16.04. Specifikacije računala na kojemu su vršena testiranja su sljedeća:

- Procesor: Intel Core i7 2.6 GHz
- RAM: 8 GB
- Tip sustava: 64-bitni
- Grafička kartica: NVIDIA GeForce GTX 950M (2 GB)

Vizualizacija i obrada prikupljenih podataka napravljena je u programskom jeziku Python.

5.1. Predobrada podataka

Podaci korišteni u diplomskom radu preuzeti su sa stranice pod [22]. Napravljen je skup podataka od 10 parova slika i njihovih *ground truth* slika (tablica 5.1.). Kod izbora slika birane su raznolike slike kao što su slike s dominantnom ravnom površinom bez teksture (slika *Reindeer* ili *Monopoly*), zatim slika koja prikazuje šiljaste predmete (slika *Moebuis* ili *Cones*), slika s različitim teksturama (slika *Cones* ili *Art)*, slika s teksturom koja se ponavlja (slika *Aloe*) itd. Slike su odabrane iz različitih skupova podataka [14, 23-25]. Podatci o slikama se nalaze u tablici 5.2.

Lijeva slika	**Desna slika**	**Lijeva slika**	**Desna slika**
Tsukuba		Cones	
			

Teddy	Art
 	 
Moebius	Reindeer
 	 
Aloe	Baby3
 	 
Bowling1	Monopoly
 	 

Tablica 5.1. Prikaz odabranih slika

Naziv slike	*Format slike lijeve i desne kamere*	*Format slike dispariteta*	*Veličina slike s obzirom na originalne*	*Rezolucija slike*
***Tsukuba* [14]**	.ppm	.pgm	polovina veličine	384x288
***Cones* [23]**	.ppm	.pgm	četvrtina veličine	450x375
***Teddy* [23]**	.ppm	.pgm	četvrtina veličine	450x375
***Art* [24]**	.png	.png	polovina veličine*	695x555*
***Moebius* [24]**	.png	.png	polovina veličine*	695x555*
***Reindeer* [24]**	.png	.png	polovina veličine*	671x555*
***Aloe* [25]**	.png	.png	polovina veličine*	641x555*
***Baby3* [25]**	.png	.png	polovina veličine*	656x555*
***Bowling1* [25]**	.png	.png	polovina veličine*	626x555*
***Monopoly* [25]**	.png	.png	polovina veličine*	665x555*

Tablica 5.2. Podatci o slikama

Važno je napomenuti da ćelije u tablici 5.2. označene s * označavaju slike koje su bile prevelike kvalitete pri korištenju algoritma mc-cnn s konvolucijskim neuronskim mrežama KITTI slow i mc-cnn Middlebury slow, tj. nije bilo dovoljno računalne moći za njihovu obradu, korištena grafička kartica nije bila dovoljno velika (2 GB) te je samo u tom slučaju korištena veličina slika koja je trećina većine originalnih slika. Budući da algoritmi koriste različite ulazne i izlazne podatke, kao što je vidljivo u tablici 5.3., a i formati slika iz tablice 5.2. imaju također više vrsta formata (.ppm, .png i .pgm), prije implementacije algoritama bilo je potrebno napraviti konverziju slika. Promjena formata slika napravljena je u programskom jeziku Python u skripti kao što je prikazano u prilogu P1.1.

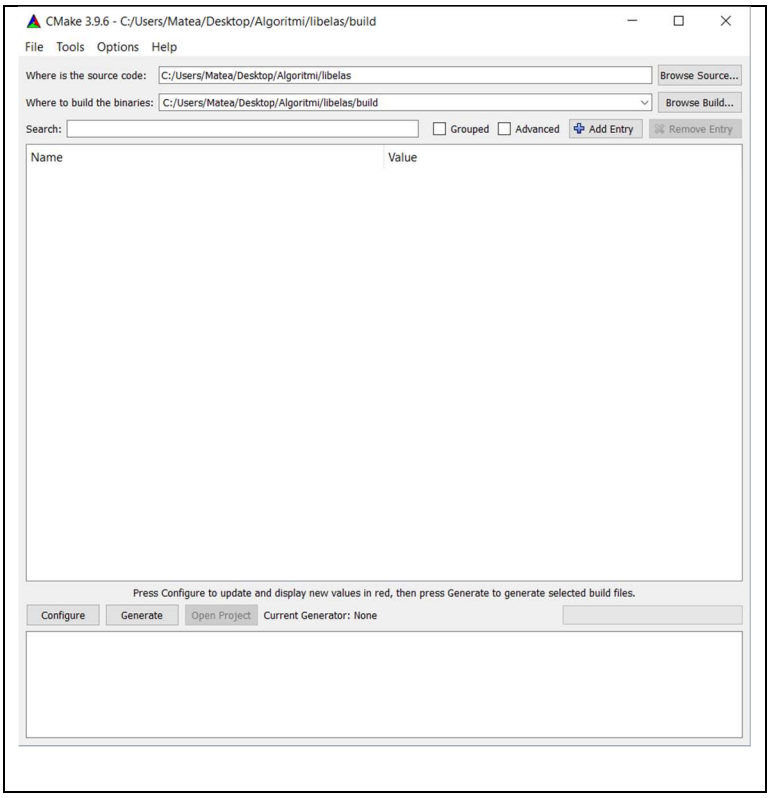
Algoritam	*Format slike ulaza*	*Format slike izlaza*
ELAS	.pgm	.pgm
SGM – Census	.png	.png
mc - cnn	.png	.png

Tablica 5.3. Formati slika ulaza i izlaza za pojedine algoritme

5.2. Testiranje algoritama

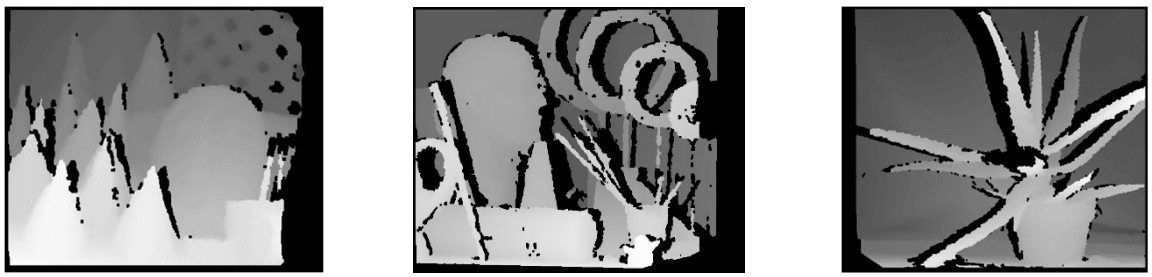
5.2.1. ELAS

Kako je već navedeno u poglavlju 4, ELAS je algoritam napisan u C++ programskom jeziku. Za njegovo pokretanje potrebno je imati C++ kompajler, zbog toga je, uz pomoć CMAKE alata za izradu rješenja neovisno o operacijskom sustavu koji je korišten, izrađen projekt u programskom okruženju Visual Studio Enterprise 2015 na operacijskom sustavu Windows 10. Nakon preuzetog koda sa stranice [17], sadržaj je raspakiran na disk te je pokrenut program CMAKE s GUI sučeljem. Na slici 5.1. prikazano je kako su postavljeni podatci. Nakon što je odabran izvorni kod i mapa u koju će biti generiran projekt Visual Studia, potrebno je pritisnuti tipku *Configure*, te nakon što je konfiguracija obavljena, pritisnuti *Generate*. Visual Studio projekt je generiran i moguće ga je pokrenuti.



Slika 5.1. Postavljanje podataka u CMAKE programu

Nakon pokretanja programa u Visual Studio programskom okruženju, kreira se datoteka *elas.exe* do koje je potrebno doći koristeći naredbu "cd putanja/do/libelas" u CMD-u. Nakon što je promijenjena mapa, program se pokreće nad dvije slike sljedećom naredbom: "*elas.exe left.pgm right.pgm*". Budući da ELAS koristi samo slike sivih tonova, ulazne slike su formata .pgm. Primjeri rezultata slika dispariteta dobivenih ovom metodom nalaze se na slici 5.2. Promatranjem primjera jasno je vidljivo kako ima dosta piksela kojima disparitet nije određen; ti pikseli su prikazani crnom bojom. Obrisi predmeta jasno su vidljivi. Sve slike dispariteta dobivene ovim algoritmom nalaze se u prilogu P2.1.



Slika 5.2. Primjeri slika dispariteta dobivenih ELAS metodom

5.2.2. SGM-Census

Algoritam SGM-Census preuzet je sa stranice [19]. Za pokretanje je korišten operacijski sustav Ubuntu 16.04 s gcc verzijom 6.3 i g++ verzijom 6.3. Također, potrebno je imati instaliran Git – program za verzioniranje koda. Naredbe za implementaciju i pokretanje SGM-Census algoritma su:

- git clone https://github.com/epiception/SGM-Census.git
- cd ~/putanja/do/SGM-Census
- make
- time ./sgm <desna slika> <lijeva slika> <naziv izlazne slike> <raspon dispariteta>

Vrijeme za procesiranje pojedine slike izmjereno je tako da se prije naredbe postavila naredba *time*. Iz prethodne je naredbe vidljivo kako je za ovaj algoritam potrebno poznavati raspon dispariteta. Kako je izlazna slika dispariteta slika sivih tonova, svaki piksel označava jedan broj u rasponu 0-255, ali budući da nisu korištene slike u najvišoj kvaliteti, taj je broj potrebno podijeliti s obzirom na korištenu kvalitetu. Middlebury kod svakog skupa podataka navodi kako odrediti raspone dispariteta [22]. Tablica raspona dispariteta po slikana dana je u tablici 5.4., a na slici 5.3. su dani primjeri slike dispariteta dobivenih metodom SGM–Census. Obrisi predmeta na primjerima su manje izraženi, međutim nema puno piksela s neodređenim disparitetom u usporedbi s prethodnom metodom. Uz desni rub slike javlja se šum. Sve slike dispariteta dobivene ovim algoritmom nalaze se u prilogu P2.2.

Naziv slike	***Tsukuba***	***Cones***	***Teddy***	***Art***	***Moebius***	***Reindeer***	***Aloe***	***Baby3***	***Bowling1***	***Monopoly***
Raspon dispariteta	32	63	63	127	127	127	127	127	127	127

Tablica 5.4. Raspon dispariteta po slikama

Slika 5.3. Primjeri slika dispariteta dobivenih SGM-Census metodom

5.2.3. mc-cnn

mc-cnn algoritam je algoritam koji zahtijeva najveću računalnu moć u usporedbi s ostalim algoritmima navedenim u ovom diplomskom radu. Kako je već navedeno, prilikom korištenja dvije verzije ovog algoritma bilo je potrebno smanjiti kvalitetu pojedinih slika. Za pokretanje metode potrebno je imati Torch, OpenCV 2.4. i png++ [21]. Budući da za Torch postoji podrška samo na operacijskom sustavu Linux, algoritam je implementiran na Ubuntu verzije 16.04. Nakon što je kod preuzet uz pomoć programa Git, potrebno ga je kompilirati koristeći naredbe:

- cp Makefile.proto Makefile
- make

Zatim je potrebno uz pomoć naredbe *wget* preuzeti s mreže već istrenirane konvolucijske neuronske mreže. Primjer naredbe za KITTI fast neuronsku mrežu slijedi:

- wget -P net/ https://s3.amazonaws.com/mc-cnn/net_kitti_fast_-a_train_all.t7

Za pokretanje iste konvolucijske neuronske mreže nad parom slika treba izvršiti naredbu:

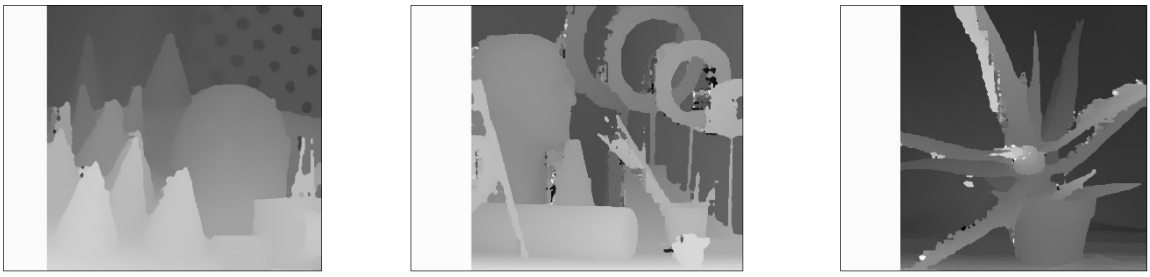
- time ./main.lua kitti fast -a predict -net_fname net/net_kitti_fast_-a_train_all.t7 -left input/left.png -right input/right.png -disp_max 127

U naredbi *time* služi za mjerenje vremena potrebnog da se naredba izvrši. *kitti* i *fast* označavaju argumente o vrsti mreže, *-left* argument označava putanju do lijeve slike, dok *-right* putanju do desne slike. *-disp_max* označava raspon dispariteta. Zbog toga što se nad konvolucijskom neuronskom mrežom s argumentima *slow* nisu mogle obraditi slike visoke kvalitete, rasponi dispariteta se razlikuju od raspona dispariteta kod metode SGM-Census. Rasponi dispariteta se nalaze u tablici 5.5., a prikaz primjer slika dispariteta po vrstama konvolucijskih neuronskih mreža se nalaze na slikama 5.4.-5.7. Sve slike dispariteta dobivene ovim algoritmom nalaze se u prilogu P2.3.-P2.6.

Naziv slike		***Tsukuba***	***Cones***	***Teddy***	***Art***	***Moebius***	***Reindeer***	***Aloe***	***Baby3***	***Bowling1***	***Monopoly***
Raspon dispariteta	**fast**	32	63	63	127	127	127	127	127	127	127
	slow	32	63	63	85	85	85	85	85	85	85

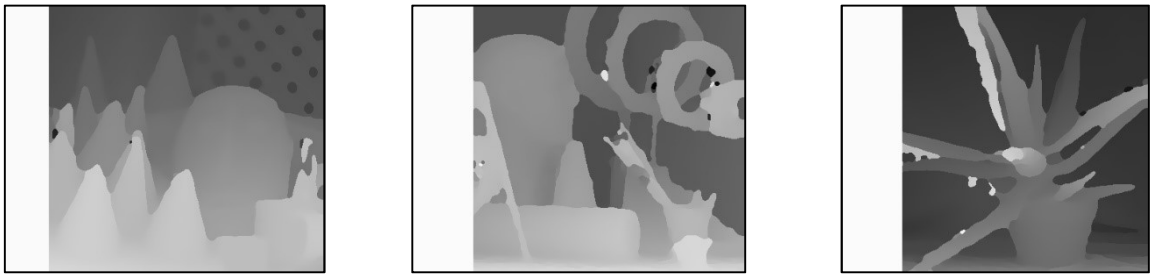
Tablica 5.5. Rasponi dispariteta za metodu mc-cnn

Slika 5.4. Primjeri slike dispariteta za algoritam mc-cnn KITTI fast



Slika 5.5. Primjeri slike dispariteta za algoritam mc-cnn Middlebury fast

Slika 5.6. Primjeri slike dispariteta za algoritam mc-cnn KITTI slow



Slika 5.7. Primjeri slike dispariteta za algoritam mc-cnn Middlebury slow

Važno je naglasiti kako nakon izvršenja prethodne naredbe metoda generira slike kao binarne datoteke formata .bin. Kako bi se te datoteke prebacile u .png format, potrebno je pozvati skriptu *bin2png.lua*.

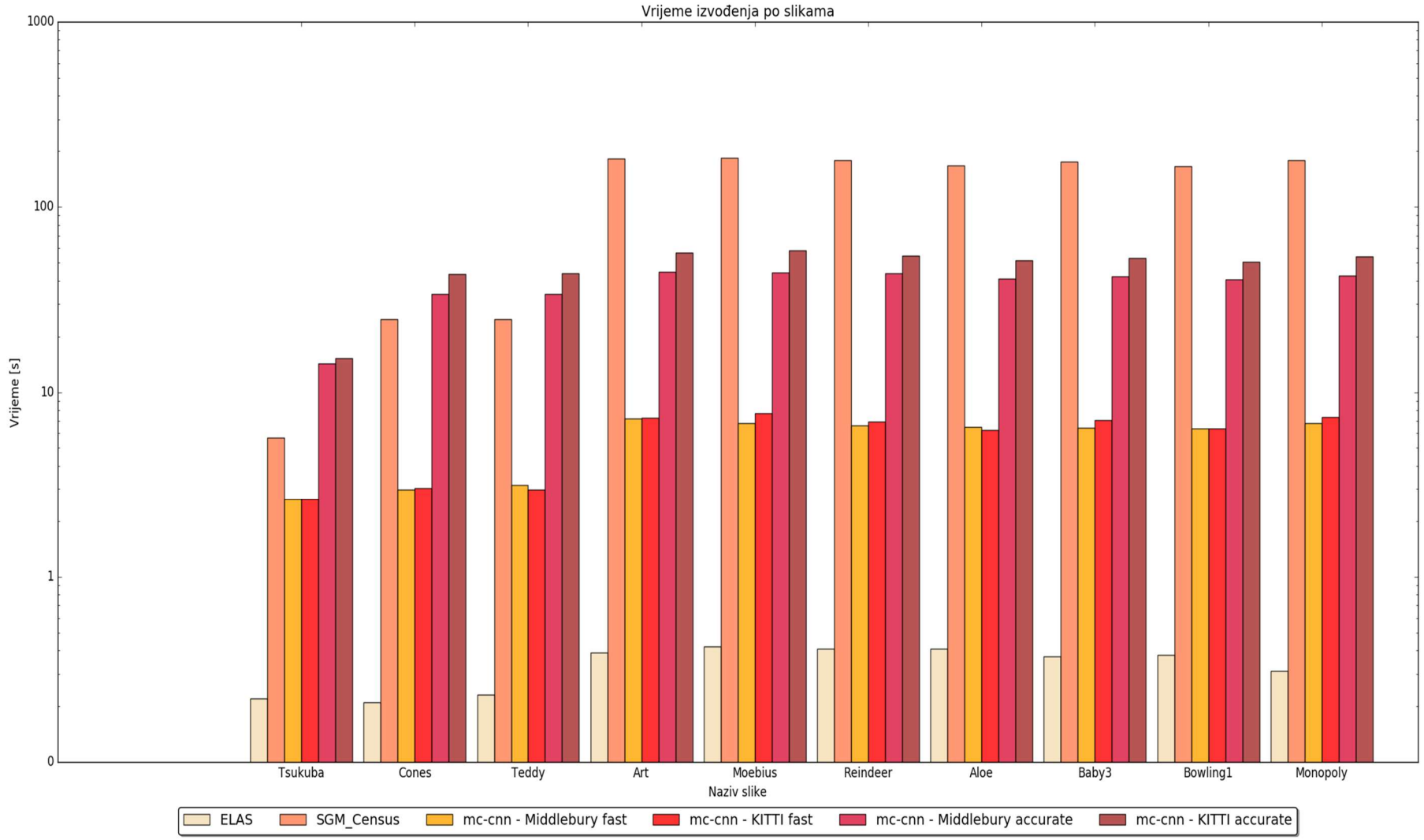
5.3. Usporedba rezultata po vremenu izvođenja

Za izračun vremena koliko je potrebno pojedinom algoritmu da obradi par slika korištena je naredba *time* ugrađena u operacijski sustav Ubuntu. Rezultati su izraženi u sekundama, a i potrebno je naglasiti kako su za mc-cnn Middlebury accurate (slow) i mc-cnn KITTI accurate (slow) korištene slike manje rezolucije: *Art*, *Moebius*, *Reindeer*, *Aloe*, *Baby3, Bowling1* i *Monopoly*. Umjesto da je kvaliteta smanjena na pola, ona je smanjena na trećinu (eng. *third size*) početne vrijednosti. U tablici 5.6. prikazan je odnos algoritama i slika s obzirom na vrijeme izvođenja.

Algoritam / Naziv slike	*ELAS*	*SGM-Census*	*mc-cnn Middlebury fast*	*mc-cnn KITTI fast*	*mc-cnn Middlebury accurate*	*mc-cnn KITTI accurate*
Tsukuba	0.22	5.707	2.624	2.630	14.284	15.260
Cones	0.21	24.875	2.989	3.043	33.755	43.215
Teddy	0.23	24.777	3.164	2.982	34.072	43.830
Art	0.39	182.797	7.217	7.245	44.675*	56.706*
Moebius	0.42	184.176	6.797	7.694	44.316*	58.221*
Reindeer	0.41	179.070	6.619	6.962	43.863*	54.362*
Aloe	0.41	167.629	6.467	6.259	40.831*	51.668*
Baby3	0.37	174.711	6.455	7.075	42.067*	52.929*
Bowling1	0.38	165.529	6.386	6.353	40.544*	50.586*
Monopoly	0.31	178.527	6.776	7.324	42.688*	53.940*

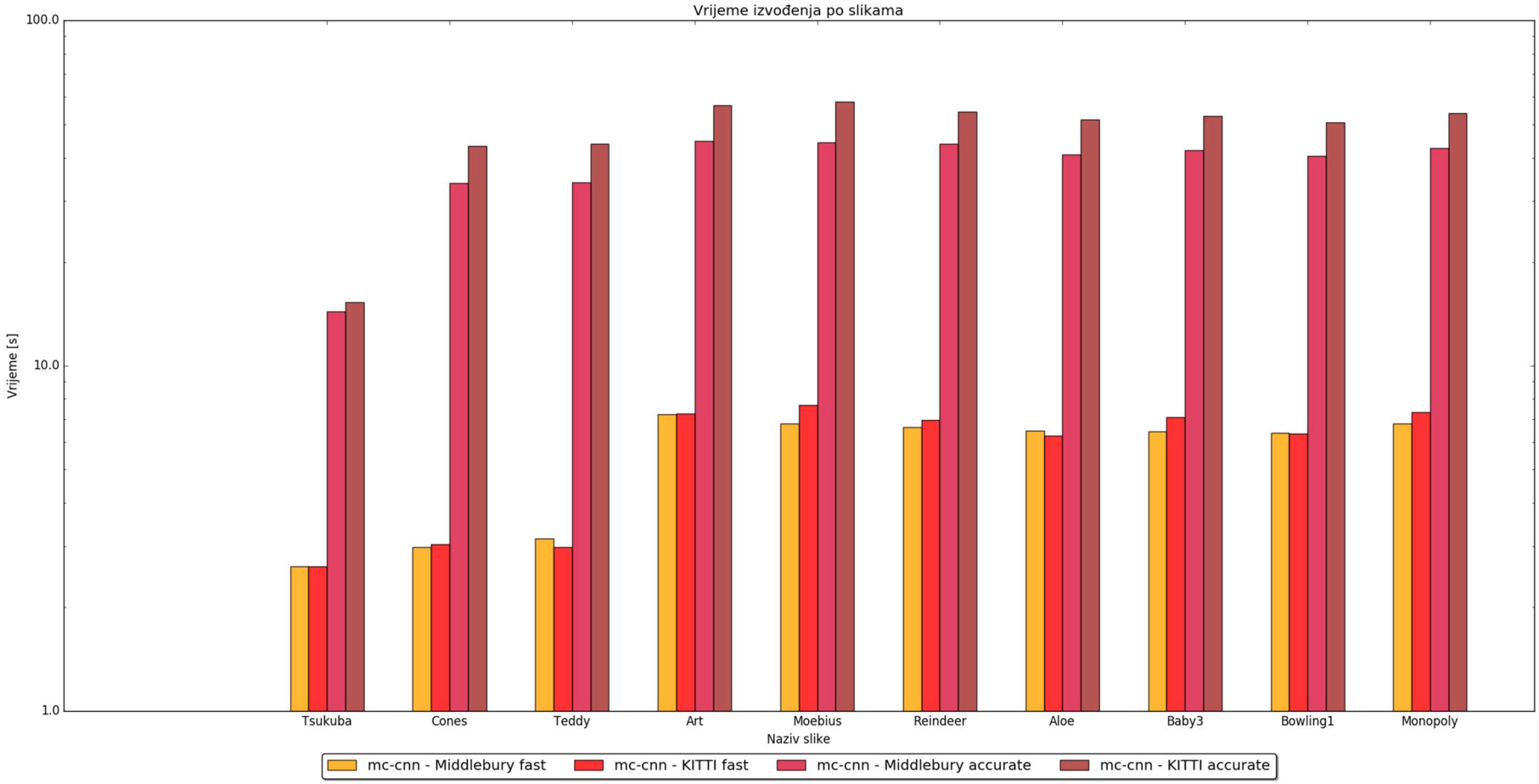
Tablica 5.6. Prikaz vremena izvođenja algoritma po slikama [s] (označava slike za koje su korištene slike manjih dimenzija kako je objašnjeno u prošlom poglavlju)*

Na slici 5.8. ovisnost algoritma o slici po vremenu izvođenja prikazana je stupičastim dijagramom. Budući da su razlike između vremena izvođenja SGM-Census algoritma i algoritma ELAS jako velike, korištena je logaritamska skala za vrijeme kako bi sve vrijednosti bile dobro vidljive.



Slika 5.8. Stupičasti dijagram vremena izvođenja algoritama po slikama

Zato što metoda mc-cnn koristi različito istrenirane konvolucijske neuronske mreže istrenirane na dva različita skupa podataka treba obratiti pažnju i na njihove odnose. Tako je vidljivo da je mreža KITTI accurate (slow) uvijek sporija u odnosu na Middlebury accurate (slow) mrežu. Za te mreže treba uzeti u obzir kako su korištene slike manjih dimenzija pa iznosi vremena nisu relevantni u odnosu s drugim algoritmima i mrežama. Mreže KITTI fast i Middlebury fast imaju približno jednaka vremena izvođenja (Slika 5.9.). Skripta u programskom jeziku Python za vizualizaciju stupičastog dijagrama sa slike 5.7. nalazi se u prilogu u prikazu koda P1.2.



Slika 5.8. Stupičasti dijagram vremena izvođenja algoritma mc-cnn po slikama

5.4. Usporedba rezultata po točnosti

Usporedba rezultata po točnosti se vrši uz pomoć dvije metode – BMP metode i Metode srednje kvadratne pogreške.

5.4.1. Metoda pogrešno sparenih piksela

Metoda pogrešno sparenih piksela (eng. *Bad Matched Pixel, BMP*) je jednostavna metoda koja provjerava postotak krivo spojenih piksela. Ne penalizira pogreške s obzirom na veličinu pogreške. Računa se prema formuli (5-1) [26].

$$BMP = \frac{1}{N}\sum_{(x,y)} \varepsilon_{(x,y)};\ \varepsilon_{(x,y)} = \begin{cases} 1, & |D_{GT} - D_E| > \delta \\ 0, & |D_{GT} - D_E| < \delta \end{cases}, \qquad (5\text{-}1)$$

gdje je *N* broj piksela, D_{GT} iznos stvarnog dispariteta, D_E je iznos izračunatog dispariteta i δ je određena tolerancija. Skripta pomoću koje se izračunava BMP se nalazi u prilogu P1.3. Svaki algoritam je testiran tako da δ iznosi 1, 2 ili 10, a rezultati mjerenja s obzirom na iznos tolerancije su prikazani u tablicama 5.7-5.9. Tablice 5.7-5.9 prikazane su dijagramima na slikama 5.9-5.11.

Algoritam / Naziv slike	*ELAS*	*SGM-Census*	*mc-cnn Middlebury fast*	*mc-cnn KITTI fast*	*mc-cnn Middlebury accurate*	*mc-cnn KITTI accurate*
Tsukuba	0.7051	0.6768	0.7860	0.7811	0.7870	0.7801
Cones	0.7587	0.6546	0.7897	0.7567	0.7905	0.5774
Teddy	0.7853	0.6776	0.8000	0.7983	0.7989	0.6036
Art	0.6667	0.6167	0.9010	0.8435	0.9059	0.5471
Moebius	0.7845	0.5187	0.7014	0.6636	0.7031	0.5151
Reindeer	0.7596	0.5564	0.8075	0.7215	0.7910	0.4951
Aloe	0.7685	0.4602	0.6496	0.5690	0.6583	0.3703
Baby3	0.7473	0.4992	0.6597	0.5702	0.6614	0.3962
Bowling1	0.7501	0.6179	0.8098	0.7590	0.8047	0.5848
Monopoly	0.7437	0.6046	0.8398	0.9250	0.8539	0.4652

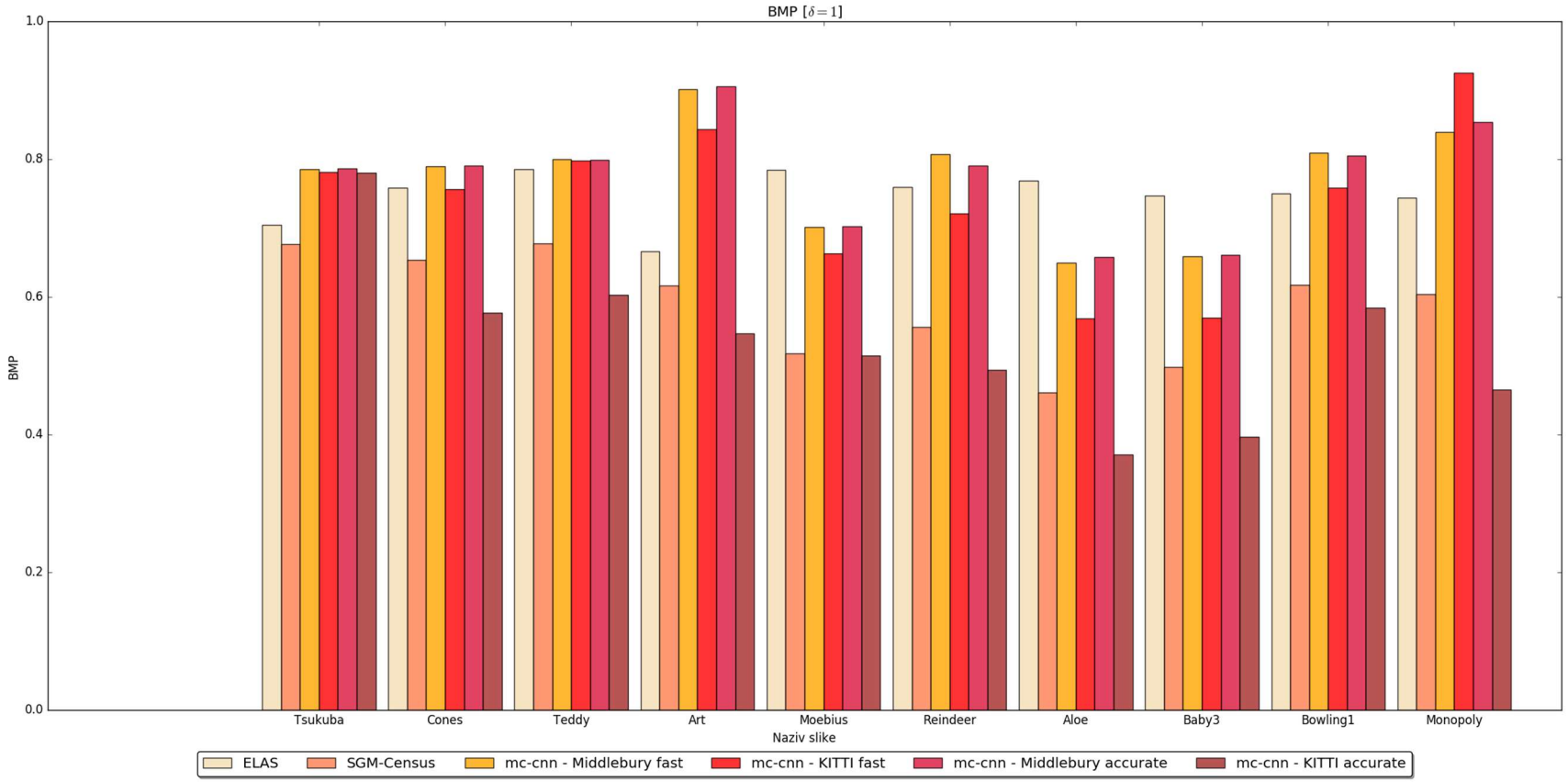
Tablica 5.7. BMP [δ=1]

Algoritam / Naziv slike	*ELAS*	*SGM-Census*	*mc-cnn Middlebury fast*	*mc-cnn KITTI fast*	*mc-cnn Middlebury accurate*	*mc-cnn KITTI accurate*
Tsukuba	0.6774	0.6768	0.7828	0.7733	0.7847	0.7734
Cones	0.7500	0.4941	0.6829	0.6104	0.6771	0.3835
Teddy	0.7853	0.5147	0.6340	0.6022	0.6395	0.4229
Art	0.6582	0.4837	0.8618	0.7955	0.8780	0.4092
Moebius	0.7770	0.3675	0.6236	0.5347	0.6333	0.3534
Reindeer	0.7557	0.3967	0.7467	0.6398	0.7301	0.3549
Aloe	0.7666	0.3046	0.5953	0.4869	0.6043	0.2687
Baby3	0.7473	0.3208	0.6175	0.5073	0.6138	0.2530
Bowling1	0.7020	0.4684	0.7429	0.6814	0.7339	0.3727
Monopoly	0.7437	0.4655	0.8299	0.9020	0.8276	0.3978

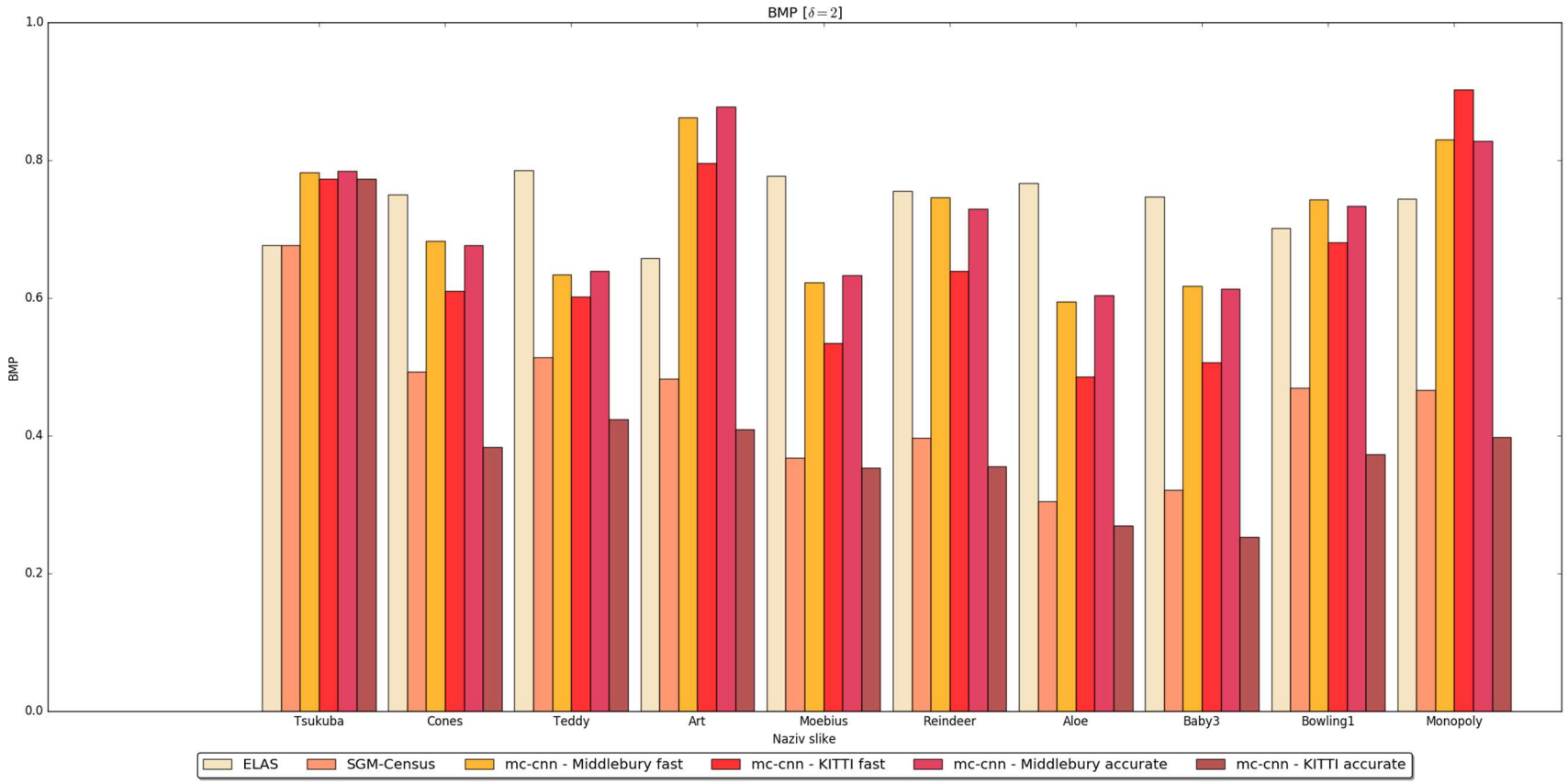
Tablica 5.8. BMP [δ=2]

Algoritam / Naziv slike	*ELAS*	*SGM-Census*	*mc-cnn Middlebury fast*	*mc-cnn KITTI fast*	*mc-cnn Middlebury accurate*	*mc-cnn KITTI accurate*
Tsukuba	0.4758	0.6067	0.5203	0.5233	0.5073	0.6391
Cones	0.7011	0.1326	0.4573	0.3299	0.4558	0.1114
Teddy	0.7841	0.1411	0.3306	0.2194	0.3311	0.0965
Art	0.6129	0.2826	0.6352	0.5523	0.6467	0.2546
Moebius	0.7098	0.1700	0.4844	0.3433	0.4864	0.0990
Reindeer	0.7163	0.2301	0.5775	0.4232	0.5827	0.1231
Aloe	0.7390	0.1600	0.4904	0.3079	0.4924	0.1090
Baby3	0.7473	0.1206	0.4069	0.2715	0.4129	0.0697
Bowling1	0.6127	0.2166	0.4174	0.2919	0.4119	0.1433
Monopoly	0.7127	0.2598	0.4094	0.4527	0.3968	0.3307

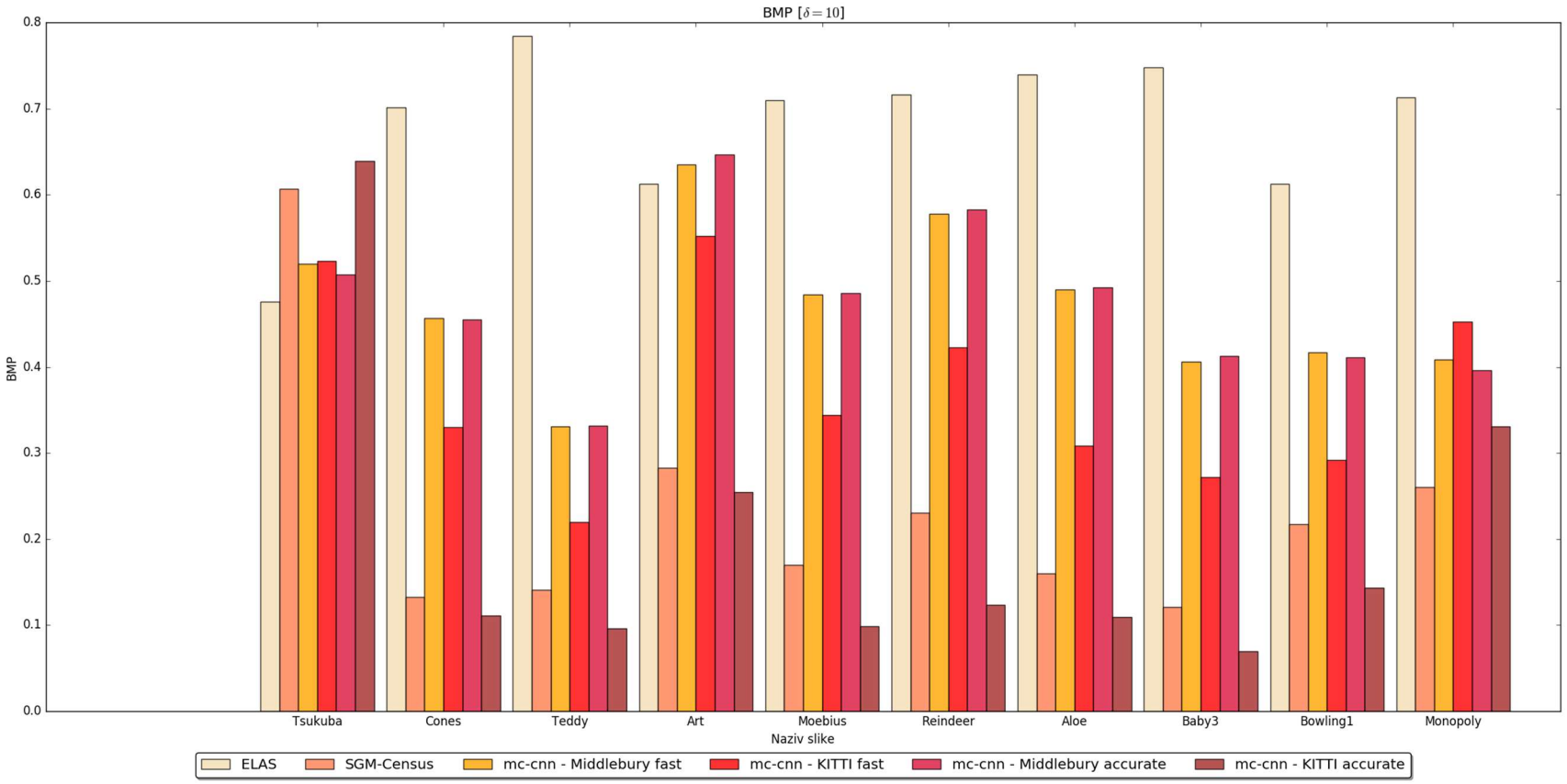
Tablica 5.9. BMP [δ=10]



Slika 5.9. BMP [δ=1]



Slika 5.10. BMP [δ=2]



Slika 5.11. BMP [δ=10]

BMP daje poprilično velike pogreške za navedene algoritme, KITTI slow daje najbolje rezultate, a što je iznenađujuće budući da su slike korištene iz Middleburyevog skupa podataka. Međutim, postoji greška u kodu koja još nije ispravljena od strane autora i prilikom korištenja Middlebury konvolucijske neuronske mreže, na izlaznoj slici dispariteta se pojavljuje velika bijela margina kao što je vidljivo na slikama 5.5 i 5.7. Također treba uzeti u obzir da su korištene slike manje rezolucije nego one predstavljene u znanstvenom radu [19]. Izlazne slike dispariteta nisu manipulirane, što znači da većina ima marginu koja utječe na točnost. Dijagrami po pojedinim algoritmima prikazani su u prilogu P3.1.-P3.6.

5.4.2. Metoda srednje kvadratne pogreške

Metoda srednje kvadratne pogreške (eng. *Root Mean Squared Error*) je također jedna od metoda pri evaluaciji Middlebury instituta. Ako je *N* broj piksela slike, D_{GT} iznos stvarnog dispariteta, a D_E je iznos izračunatog dispariteta, RMS se može izračunati prema jednadžbi (5-2) [14].

$$RMS = \sqrt{\frac{1}{N}\sum_{(x,y)}|D_{GT} - D_E|^2}\,, \tag{5-2}$$

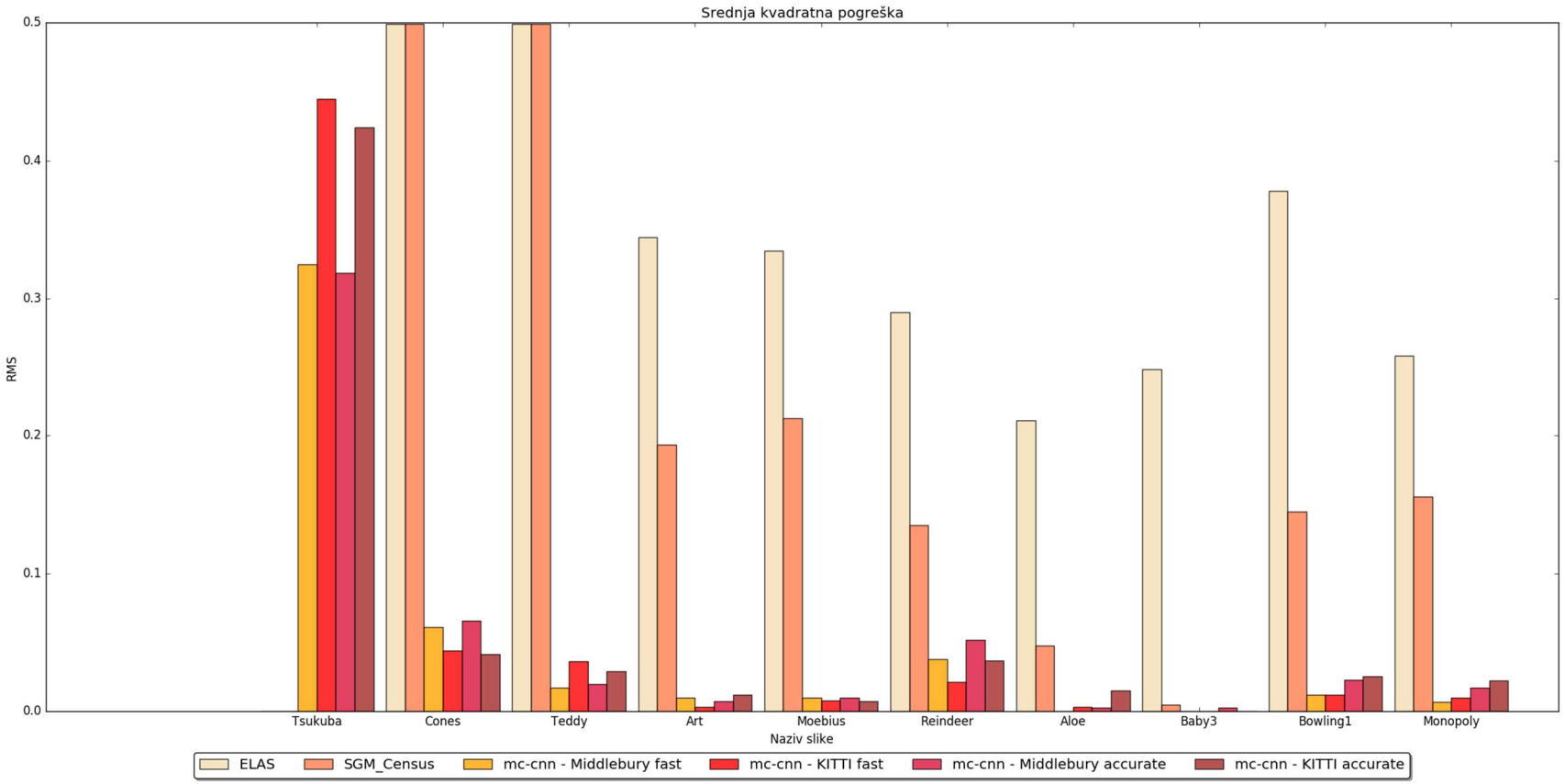
Python skripta za izračunavanje srednje kvadratne pogreške prikazana je u prilogu P1.4.

U tablici 5.10 nalaze se iznosi srednje kvadratne pogreške po algoritmima i slikama. Srednja kvadratna pogreška izražena je u jedinicama dispariteta u intervalu [0,1]. [14]

Algoritam / Naziv slike	*ELAS*	*SGM-Census*	*mc-cnn Middlebury fast*	*mc-cnn KITTI fast*	*mc-cnn Middlebury accurate*	*mc-cnn KITTI accurate*
Tsukuba	0.0	0.0	0.3247	0.4450	0.3187	0.4239
Cones	0.4990	0.4990	0.0608	0.0438	0.0657	0.0413
Teddy	0.4990	0.4990	0.0170	0.0365	0.0194	0.0292
Art	0.3445	0.1932	0.0096	0.0032	0.0072	0.0120
Moebius	0.3349	0.2126	0.0096	0.0080	0.0096	0.0072
Reindeer	0.2900	0.1352	0.0376	0.02130	0.0516	0.0368
Aloe	0.2112	0.0478	0.0	0.00335	0.0025	0.0150
Baby3	0.2485	0.0049	0.0	0.0	0.0024	0.0
Bowling1	0.3783	0.1451	0.0118	0.0118	0.0229	0.0254
Monopoly	0.2584	0.1556	0.0065	0.0098	0.0172	0.0222

Tablica 5.10. Prikaz srednje kvadratne pogreške po algoritmima i slikama

Za lakšu analizu rezultati su prikazani i na slici 5.12.



Slika 5.12. Stupičasti dijagram srednje kvadratne pogreške

Kao što je vidljivo iz tablice 5.10, ali i iz dijagrama (slika 5.12.) za prvu sliku, odnosno Tsukubu, bez obzira koja se vrsta konvolucijske neuronske mreže koristila, rezultati će biti znatno lošiji nego u odnosu na ostale slike. Za Tsukubu je karakteristično da je tamnija slika puno manje kvalitete nego ostale. Nadalje, algoritam ELAS gotovo na svim slikama ima nekoliko puta veću kvadratnu pogrešku od ostalih, a slijedi ga SGM-Census koji je po BMP-u davao najbolje rezultate. Rezultate mc-cnn algoritma na posljednjih 7 slika ne treba međusobno uspoređivati nego po argumentima *fast* i *slow* budući da slike korištene u ta dva slučaja nisu iste rezolucije, nego su za *slow* konvolucijske neuronske mreže korištene slike lošije kvalitete. U ovom slučaju mreža koja je učena na Middlebury skupu podataka ima bolje rezultate.

6. ZAKLJUČAK

Najveći problem u stereo viziji predstavlja određivanje slike dispariteta. Kako bi se taj problem riješio, intenzivno se radi na izradi novih načina kako mu pristupiti. Kroz ovaj diplomski rad odabrano je nekoliko novijih algoritama za stereo viziju koji se bave određivanjem slika dispariteta, te su opisane tehnologije koje su potrebne za njihovu implementaciju i analizu. Osim teorijske podloge o pojedinim algoritmima, opisan je i proces odabira podataka za testiranje, njihova predobrada i sama primjena algoritama nad podatcima. Dva važna čimbenika u robotskom vidu su vrijeme i točnost; stoga su upravo ta dva argumenta uzeta u razmatranje prilikom testiranja.

Po vremenu je najbolje rezultate imao algoritam ELAS koji koristi slike visoke rezolucije. Najlošije rezultate po vremenu imao je algoritam SGM - Census, što i ne čudi budući da se kod izvršava na procesoru, a ne na grafičkoj kartici kao kod mc-cnn algoritma. mc-cnn algoritam podijeljen je prema podatcima na kojima su konvolucijske neuronske mreže istrenirane - KITTI i Middlebury, te prema brzini izvođenja pa se tako dijeli na brzi (eng. *fast*) i spori (eng. *slow*). Rezultati testiranja po vremenu kod ovih algoritama odgovaraju njihovim nazivima - brze mreže su brže od sporih. Što se tiče testiranja vremena i korištenih slika, prema očekivanjima slike s većom rezolucijom trebaju više vremena za obradu.

Što se tiče analize na osnovu točnosti, korištene su dvije metode - metoda pogrešno uparenih piksela (BMP) i metoda srednje kvadratne pogreške. Kod BMP metode rezultati su loši kod odabira $\delta=1$ i $\delta=2$, što je za očekivati budući da je ta metoda dosta osjetljiva. Uz to, zbog smanjene računalne moći korištene slike nisu u punoj rezoluciji kako se preporuča. Važno je i napomenuti kako svaki od navedenih algoritama radi s ispravljenim slikama, dok slike korištene u diplomskom radu nisu ispravljene kako bi prikazivale realne uvjete. Kod usporedbe točnosti pomoću BMP metode najbolje rezultate ima algoritam SGM-Census koji je bio zadnji po vremenu izvođenja.

Međutim analizom srednje kvadratne pogreške najbolje rezultate ima algoritam mc-cnn bez obzira na korištenu vrstu konvolucijske neuronske mreže. Rezultati te metode su dosta dobri. Gledajući slike i metode za analizu točnosti, vidljivo je kako je prva slika Tsukuba problematična gotovo svim algoritmima. Tsukuba je slika najlošije kvalitete u skupu podataka i predmeti na njoj nisu jako osvjetljeni, uz to po rezultatima s obzirom na točnost slike lošije kvalitete imaju i lošije rezultate od onih bolje kvalitete. To je logično jer što je slika veće rezolucije, sadrži više piksela, a samim time i više podataka. Važno je i napomenuti kako slike s izraženom teksturom imaju bolje rezultate od slika na kojima nema objekata s teksturom. Nad slikama koje su dobivene algoritmima

nije vršena nikakva manipulacija, čak nije ni uklonjena izražena crna ili bijela margina, što je još jedan od razloga zašto su rezultati usporedbe rezultata pomoću metode BMP lošiji.

LITERATURA

[1] Stereo Vision, *DLR - Institute of Robotics and Mechatronics - Stereo Vision,* http://www.dlr.de/rm/en/desktopdefault.aspx/tabid-9389/16104_read-39811/, ožujak 2018.

[2] Visual Studio, *Visual Studio IDE*, https://www.visualstudio.com/vs/, ožujak 2018.

[3] OpenCV, *Introduction — OpenCV 2.4.13.6 documentation*, https://docs.opencv.org/2.4.13.6/modules/core/doc/intro.html, ožujak 2018.

[4] OpenCV, *OpenCV: Introduction*, https://docs.opencv.org/3.4.1/d1/dfb/intro.html, ožujak 2018.

[5] C++, *A Brief Description - C++ Information*, http://www.cplusplus.com/info/description/, ožujak 2018.

[6] C++, *History of C++ - C++ Information*, http://www.cplusplus.com/info/history/, ožujak 2018.

[7] Python, *History and License — Python 3.6.4 documentation*, https://docs.python.org/3/license.html, ožujak 2018.

[8] Python, *The Python Tutorial — Python 3.6.4 documentation*, https://docs.python.org/3/tutorial/index.html, ožujak 2018.

[9] Python, *1. Whetting Your Appetite — Python 3.6.4 documentation,* https://docs.python.org/3/tutorial/appetite.html, ožujak 2018.

[10] K. Banožić, *Analiza scene iz više pogleda,* https://www.fer.unizg.hr/_download/repository/KDI_Kata_Banozic.pdf, 2012.

[11] R. Cupec, *6. Trodimenzionalna rekonstrukcija scene na temelju više slika*, https://loomen.carnet.hr/pluginfile.php/312992/mod_resource/content/1/PRED/RV_PR06.pdf, Osijek 2010.

[12] DLR, *Robotics and Mechatronics Center - Stereo Vision*, http://www.dlr.de/rmc/rm/en/desktopdefault.aspx/tabid-9389/16104_read-39811/, 2017.

[13] Middlebury College , *Stereo*, http://vision.middlebury.edu/stereo/, 2017.

[14] D. Scharstein, R. Szeliski, *A taxonomy and evaluation of dense two-frame stereo correspondence algorithms*, International Journal of Computer Vision, 47(1/2/3):7-42, travanj-lipanj 2002.

[15] Autonomus Vision Group, *ELAS*, http://www.cvlibs.net/software/libelas/, 2017.

[16] A. Geiger, M. Roser, R. Urtasun, *Efficient Large-Scale Stereo Matching*, http://www.cvlibs.net/publications/Geiger2010ACCV.pdf, 2010.

[17] ELAS, Download ELAS, http://www.cvlibs.net/download.php?file=libelas.zip, ožujak 2018.

[18] H. Hirschmuller, Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information, http://www.dlr.de/rm/en/PortalData/3/Resources/papers/modeler/cvpr05hh.pdf, 2005.

[19] SGM-Census , *GitHub - SGM-Census: Semiglobal Matching with Census Matching Cost*, https://github.com/epiception/SGM-Census, veljača 2018.

[20] J.Žbontar, Y.LeCun, *Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches,* https://arxiv.org/pdf/1510.05970.pdf, 2016.

[21] J.Žbontar, *Git - Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches*, http://jzbontar.github.io/mc-cnn/, 2016.

[22] *Middlebury Stereo Datasets*, http://vision.middlebury.edu/stereo/data/, veljača 2018.

[23] D. Scharstein, R. Szelisk,. *High-accuracy stereo depth maps using structured light*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003), izdanje 1, stranice 195-202, Madison, WI, lipanj 2003.

[24] D. Scharstein, C. Pal, *Learning conditional random fields for stereo*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), Minneapolis, MN, lipanj 2007.

[25] H. Hirschmüller, D. Scharstein, *Evaluation of cost functions for stereo matching*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), Minneapolis, MN, lipanj 2007

[26] I. Cabezas, V. Padilla, M. Trujillo, A Measure for Accuracy Disparity Maps Evaluation, https://pdfs.semanticscholar.org/37ba/e1d6361f0f543868f160325646fc6a797df1.pdf, 2011.

SAŽETAK

U diplomskom radu odabrano je nekoliko novijih algoritama za stereo viziju koji su postigli dobre rezultate na Middlebury Stereo Vision stranici. Algoritmi su korišteni za određivanje slike dispariteta što je inače problem u stereo viziji. U praktičnom su dijelu rada algoritmi implementirani prema uputama i odabran je set slika nad kojima je izvršeno testiranje i usporedba algoritama. U teorijskom dijelu rada opisane su korištene tehnologije, odabrani algoritmi i metode uz pomoć kojih su rezultati analizirani, te su rezultati analizirani.

Ključne riječi: stereo vizija, robotski vid, slika dispariteta, epipolarna geometrija

ABSTRACT

In the graduate thesis, several recent algorithms for stereo vision have been selected that have achieved good results on the Middlebury Stereo Vision page. Algorithms have been used to determine the disparity matrix which is usually a problem in stereo vision. In the practical part of thesis, the algorithms were implemented according to the instructions and tested on selected data set. Obtained results were analyzed. In the theoretical part of the paper used technology was described, same as the selected algorithms and methods that are used to analyze the results.

Keywords: stereo vision, robotic vision, disparity matrix, epipolar geometry

ŽIVOTOPIS

Matea Tisaj rođena je rođena je 20. svibnja 1993. godine u Osijeku. Pohađa Osnovnu školu Tina Ujevića u Osijeku koju upisuje 2000. godine, a završava 2008. godine. Nakon toga upisuje III. gimnaziju Osijek, koju 2012. završava polaganjem državne mature. Zbog odličnog uspjeha, te brojnih izvannastavnih aktivnosti i natjecanja dobiva izravan upis na Elektrotehnički fakultet Osijek na Sveučilištu Josipa Juraja Strossmayera u Osijeku, te na istom upisuje preddiplomski smjer računarstva 2012. godine. Preddiplomski studij završava 2015. te upisuje Diplomski sveučilišni studij računarstva izborni blok DRB – Procesno računarstvo na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Aktivna je članica IEEE-a od 2013. godine.

Matea Tisaj

PRILOG

P1. Python skripte za obradu i testiranje dobivenih podataka

P1.1. Primjer koda za konverziju slika

```
for i in range(0, 9):
	im1 = Image.open(str(i+1)+'/left.ppm')
	im1 = im1.convert('RGB')
	im1.save('input/left'+str(i+1)+'.png')
	im1.save('input/left'+str(i+1)+'.pgm')
	im1.save('input/left'+str(i+1)+'.ppm')

	im2 = Image.open(str(i+1)+'/right.ppm')
	im2 = im2.convert('RGB')
	im2.save('input/right'+str(i+1)+'.png')
	im2.save('input/right'+str(i+1)+'.pgm')
	im2.save('input/right'+str(i+1)+'.ppm')

	im3 = Image.open(str(i+1)+'/right_disp.ppm')
	im3.save('input/right_disp'+str(i+1)+'.png')
	im3.save('input/right_disp'+str(i+1)+'.pgm')

	im4 = Image.open(str(i+1)+'/left_disp.ppm')
	im4.save('input/left_disp'+str(i+1)+'.png')
	im4.save('input/left_disp'+str(i+1)+'.pgm')
```

Prikaz koda P1.1. Kod za konverziju .ppm slike u .png i .pgm

P1.2. Primjer koda za prikaz podataka u stupičastom dijagramu

```
import numpy as np
import matplotlib.pyplot as plt

n_groups = 10
time_libelas = (0.22, 0.21, 0.23, 0.39, 0.42, 0.41, 0.41, 0.37, 0.38, 0.31)
time_mb_fast = (2.624, 2.989, 3.164, 7.217, 6.797, 6.619, 6.467, 6.455, 6.386, 6.776)
time_mb_slow=(14.284, 33.755, 34.072, 44.675, 44.316, 43.863, 40.831, 42.067, 40.544, 42.688)
time_kitti_fast=(2.630, 3.043, 2.982, 7.245, 7.694, 6.962, 6.259, 7.075, 6.353, 7.324)
time_kitti_slow=(15.260, 43.215, 43.830, 56.706, 58.221, 54.362, 51.668, 52.929, 50.586, 53.940)
time_sgm_census=(5.707, 24.875, 24.777, 182.797, 184.176, 179.070, 167.629, 174.711, 165.529,
178.527)

fig, ax = plt.subplots()
index = np.arange(n_groups)
bar_width = 0.15
opacity = 0.8
```

```
rects1 = plt.bar(index-2*bar_width, time_libelas, bar_width,
            alpha=opacity,
            color='wheat',
            label='ELAS')
rects2 = plt.bar(index-bar_width, time_sgm_census, bar_width,
            alpha=opacity,
            color='coral',
            label='SGM_Census')
rects3 = plt.bar(index, time_mb_fast, bar_width,
            alpha=opacity,
            color='orange',
            label='mc-cnn - Middlebury fast')
rects4 = plt.bar(index+bar_width, time_kitti_fast, bar_width,
            alpha=opacity,
            color='red',
            label='mc-cnn - KITTI fast')
rects5 = plt.bar(index+2*bar_width, time_mb_slow, bar_width,
            alpha=opacity,
            color= 'crimson',
            label='mc-cnn - Middlebury accurate')
rects6 = plt.bar(index+3*bar_width, time_kitti_slow, bar_width,
            alpha=opacity,
            color='brown',
            label='mc-cnn - KITTI accurate')

plt.yscale('log')
plt.xlabel('Naziv slike')
plt.ylabel('Vrijeme [s]')
plt.title('Vrijeme izvođenja po slikama')
plt.xticks(index + bar_width, ('Tsukuba', 'Cones', 'Teddy', 'Art', 'Moebius', 'Reindeer', 'Aloe', 'Baby3',
'Bowling1', 'Monopoly'))
plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.05),
        fancybox=True, shadow=True, ncol=10)
from matplotlib.ticker import ScalarFormatter
for axis in [ax.yaxis]:
   axis.set_major_formatter(ScalarFormatter())
plt.tight_layout()
plt.show()
```

Prikaz koda P1.2. Skripta za vizualizaciju podataka u stupičastom dijagramu

P1.3. Primjena BMP algoritma

```
for i in range(0,9):
    img=scipy.misc.imread('kitti slow/'+str(i+1)+'/disp.png')
    img2=scipy.misc.imread('disp'+str(i+1)+'.png')
    c=img.shape
    delta=2
    suma=0
    print(str(i+1))
    for k in range(0, c[0]):
```

```
      for j in range(0, c[1]):
         N=img.size
         if(abs(img.item(k,j)-img2.item(k,j))>delta):
            suma=suma+1

  BMP=suma/N
  print(BMP)
```

Prikaz koda P1.3. Primjena algoritma BMP

P1.4. Primjena metode srednje kvadratne pogreške

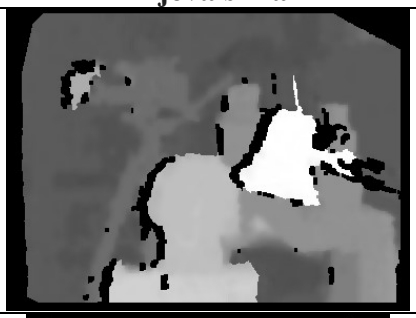
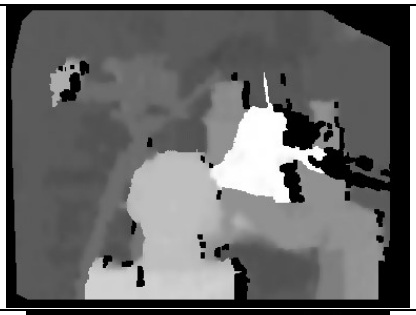
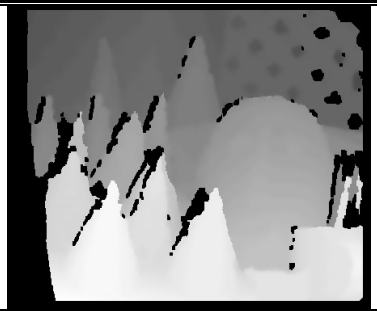
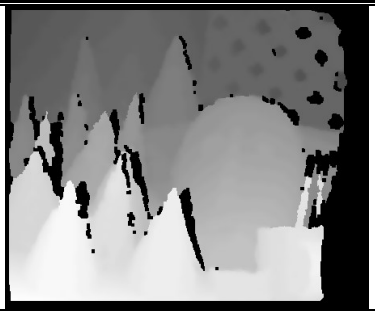
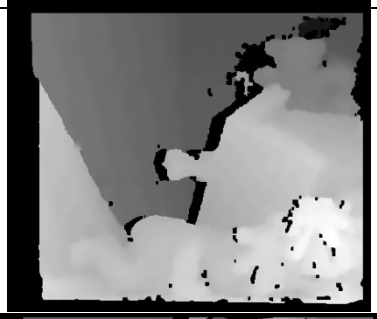
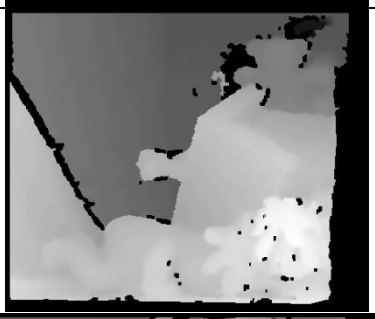
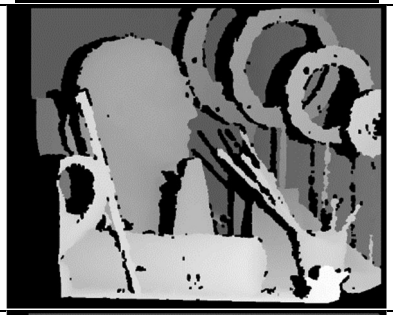
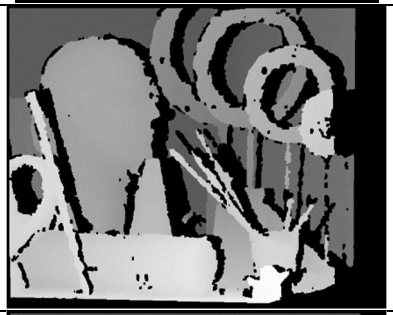
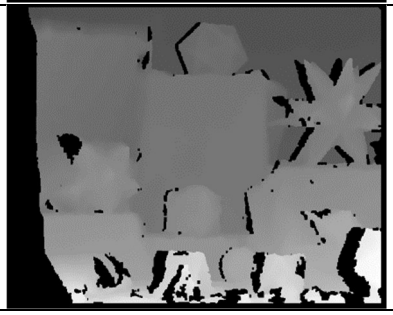
```
for i in range(0,10):
   img=scipy.misc.imread('kitti slow/'+str(i+1)+'/left.png')
   img2=scipy.misc.imread('right_disp'+str(i+1)+'.png')
   c=img.shape
   suma=0
   for k in range(0, c[0]):
      for j in range(0, c[1]):
         N=img.size
         suma=abs(img.item(k,j)-img2.item(k,j))**2

   RMS=(suma/N)**0.5
   print(RMS)
```

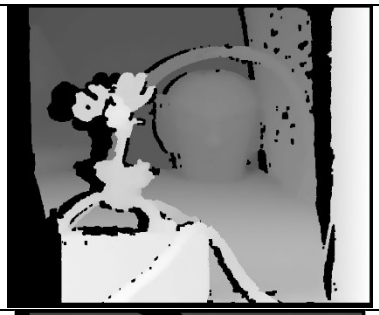
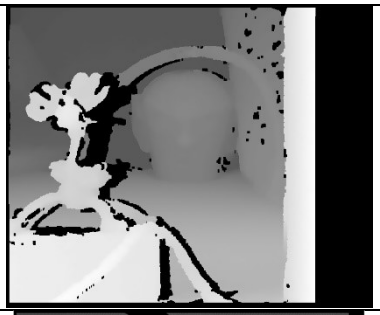
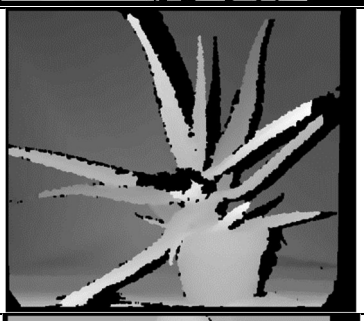
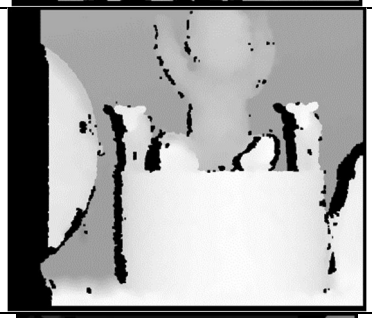
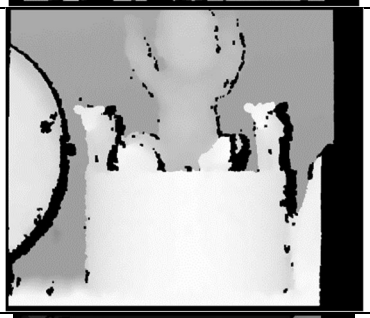
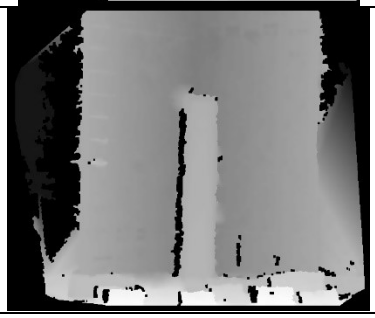
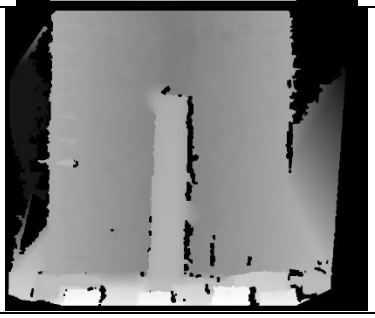
Prikaz koda P1.4. Skripta za izračunavanje RMS-a

P2. Slike dispariteta

Kod slika dispariteta po algoritmima, lijeva slika označava sliku dispariteta kada je lijeva slika korištena kao referentna, dok desna označava sliku kada je desna korištena kao referentna.

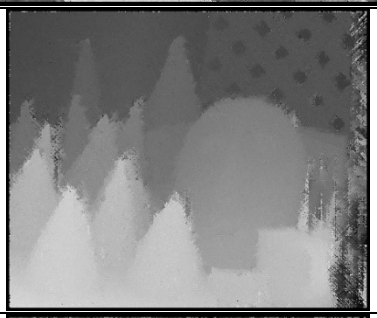
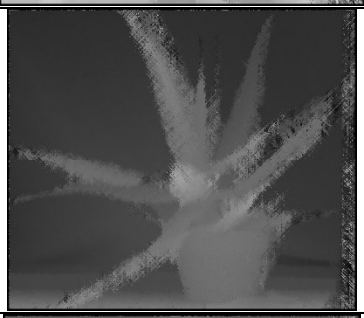
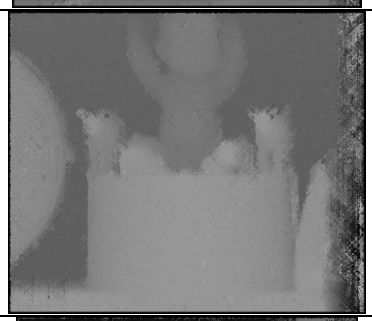
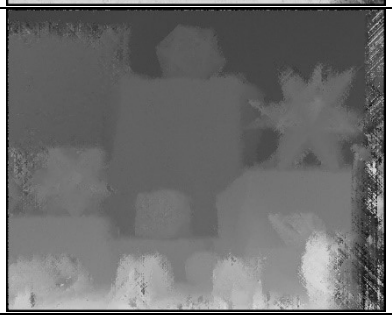
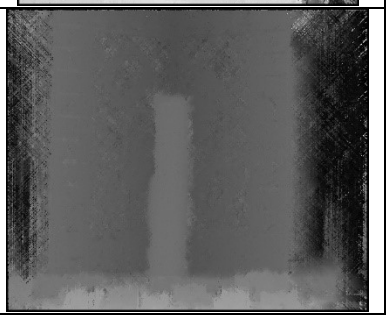
P2.1. ELAS slike dispariteta

Naziv slike	Lijeva slika	Desna slika
Tsukuba		
Cones		
Teddy		
Art		
Moebius		

Reindeer		
Aloe		
Baby3		
Bowling1		
Monopoly		

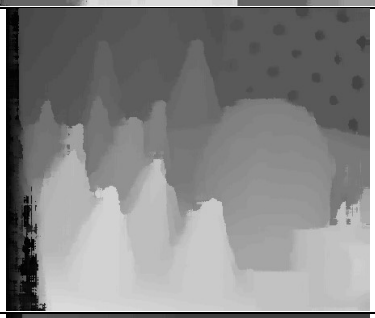
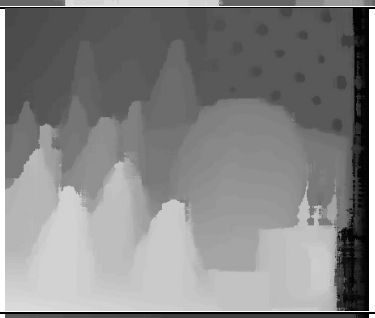
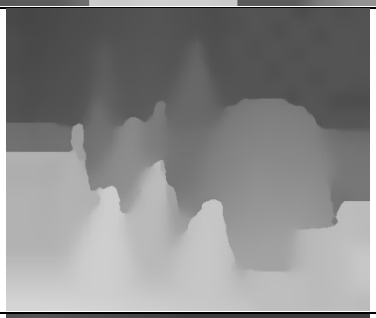
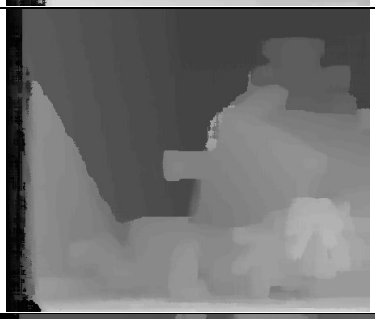
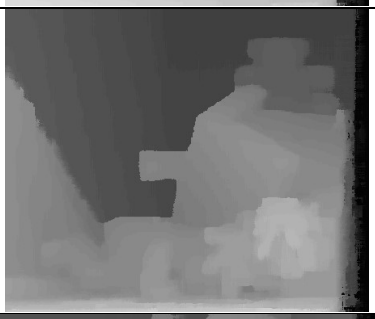
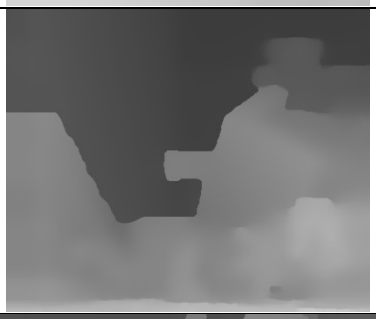
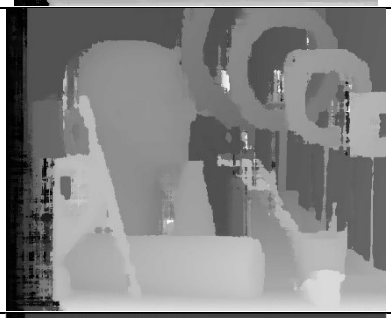
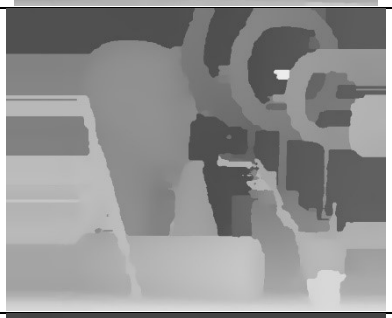
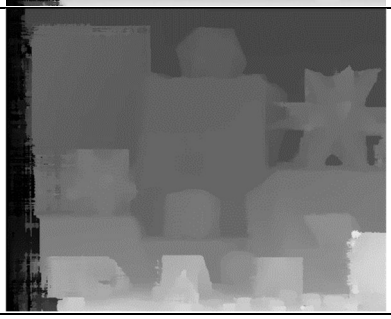
Tablica P2.1. Algoritam ELAS – slike dispariteta

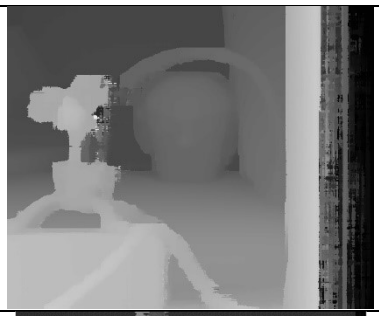
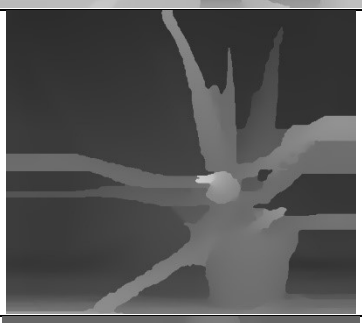
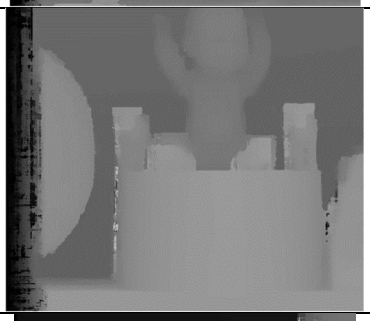
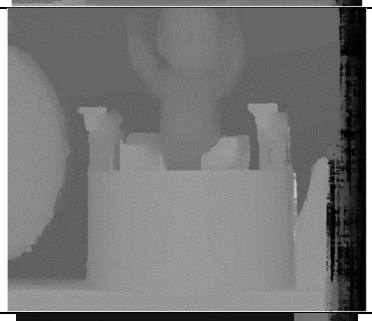
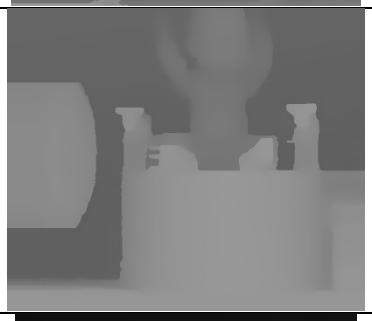
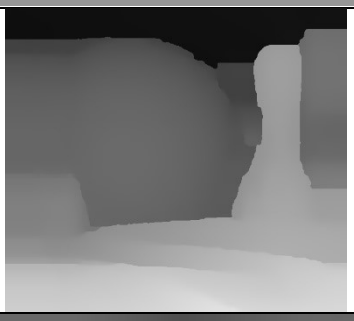
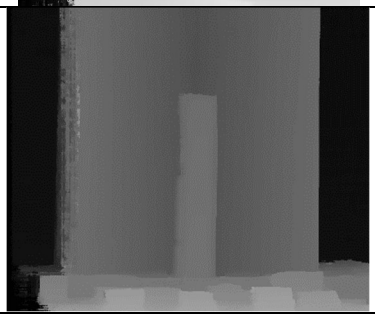
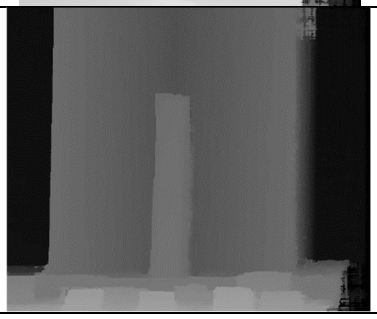
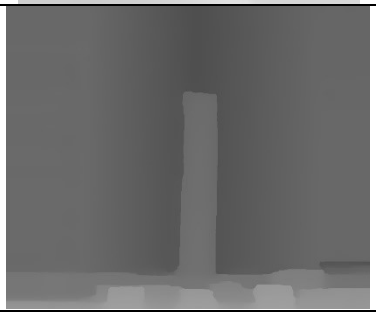
P2.2. SGM – Census slike dispariteta

Naziv slike	Slika dispariteta	Naziv slike	Slika dispariteta
Tsukuba		**Reindeer**	
Cones		**Aloe**	
Teddy		**Baby3**	
Art		**Bowling1**	
Moebius		**Monopoly**	

Tablica P2.2. Algoritam SGM-Census – slike dispariteta

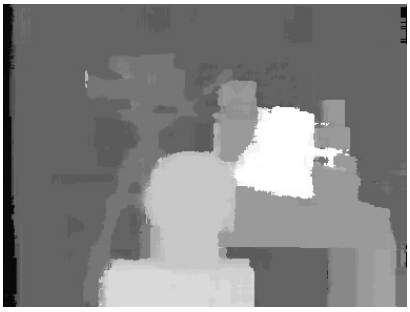
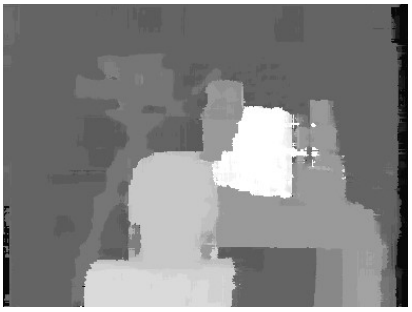
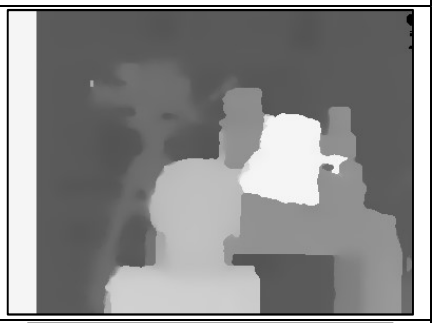
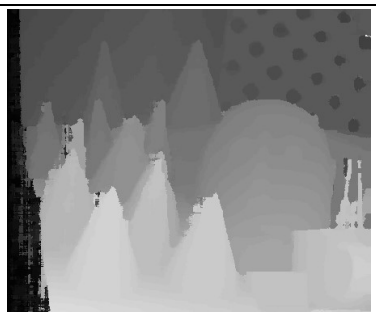
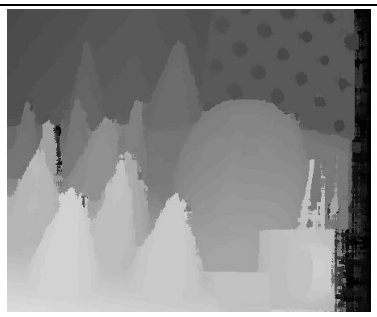
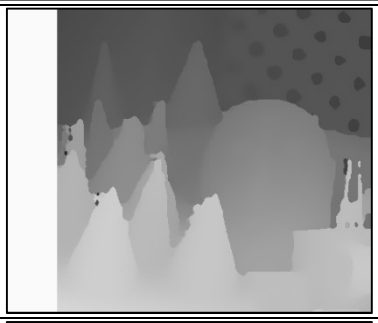
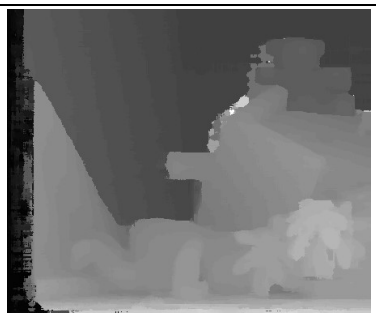
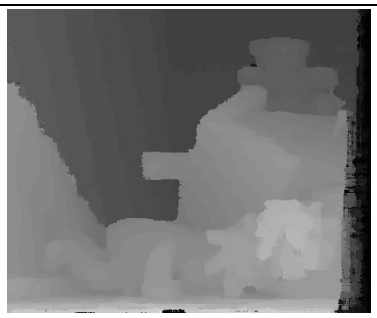
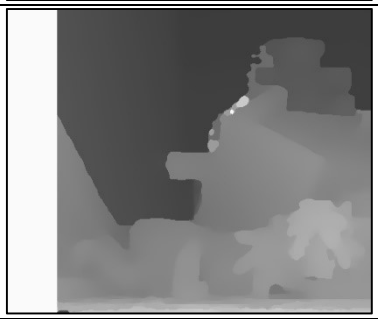
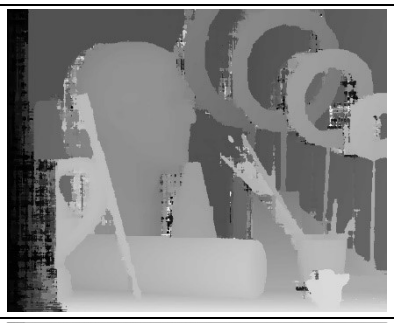
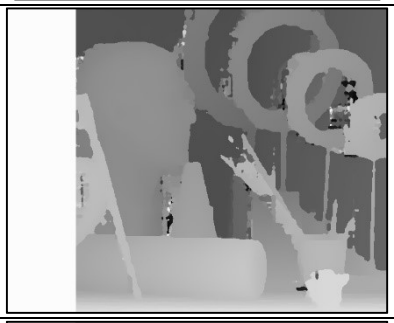
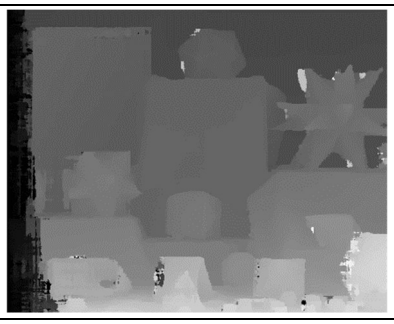
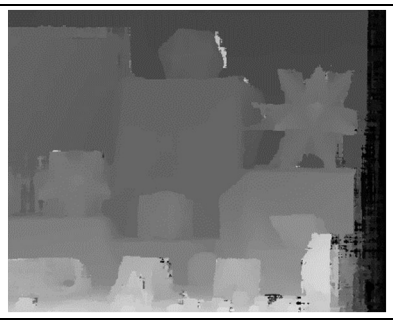
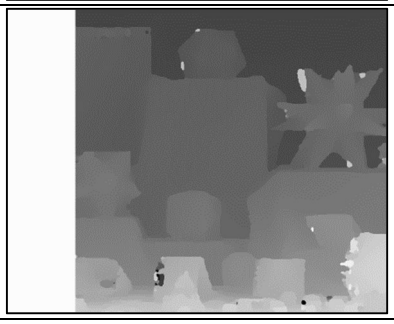
P2.3. mc-cnn KITTI fast slike dispariteta

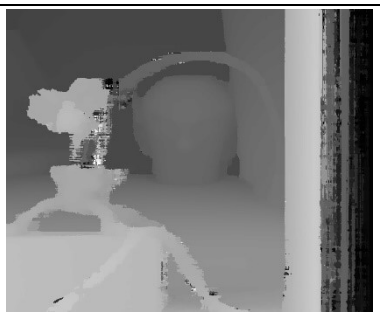
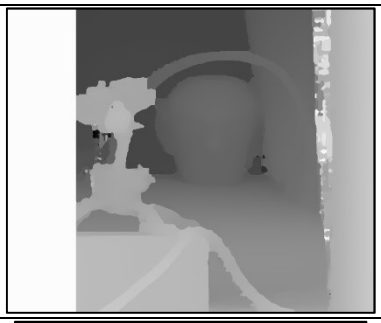
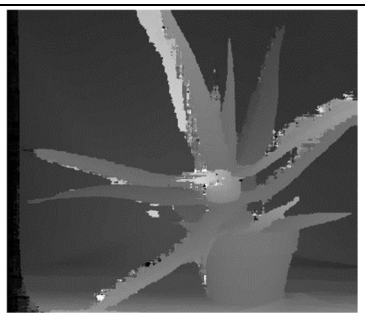
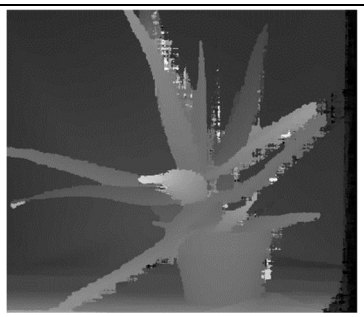
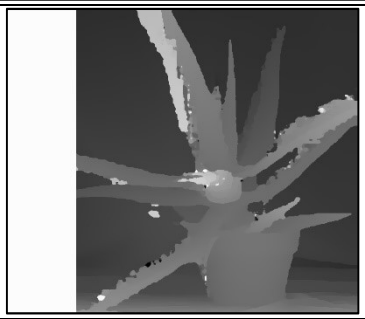
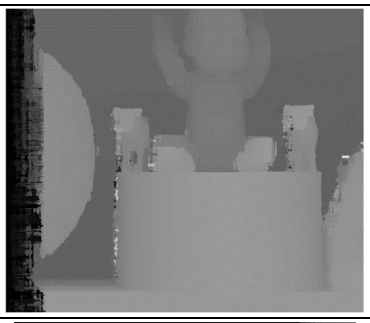
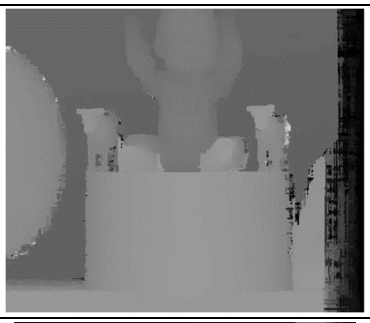
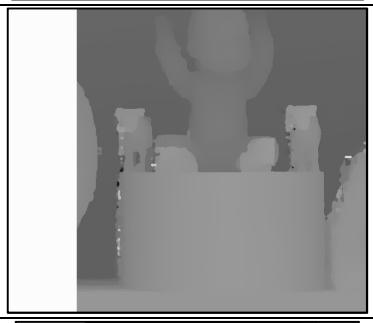
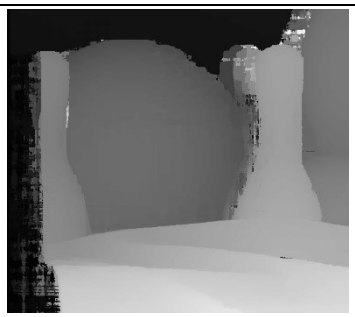
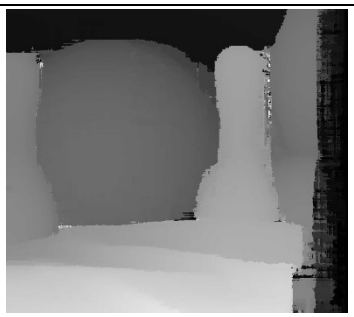
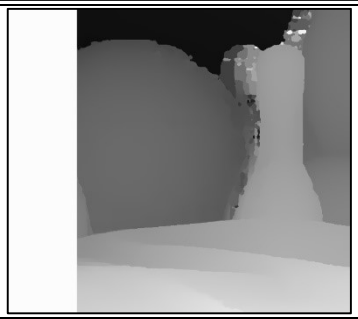
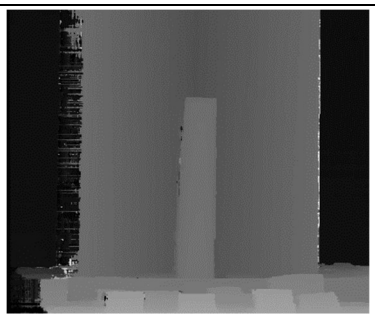
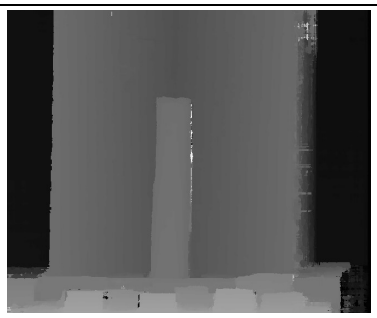
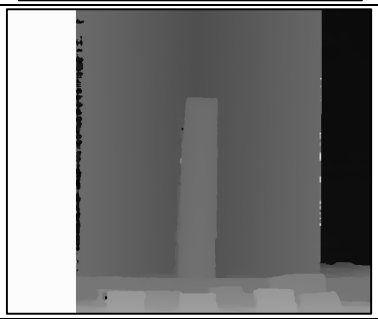
Naziv slike	Lijeva slika	Desna slika	Slika dispariteta
Tsukuba			
Cones			
Teddy			
Art			
Moebius			

Reindeer			
Aloe			
Baby3			
Bowling1			
Monopoly			

Tablica P2.3. Algoritam mc-cnn KITTI fast – slike dispariteta

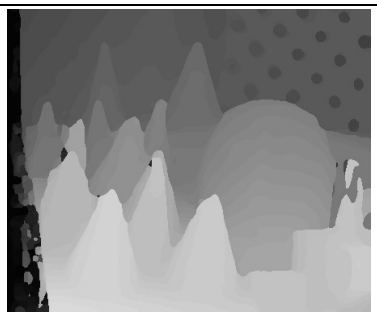
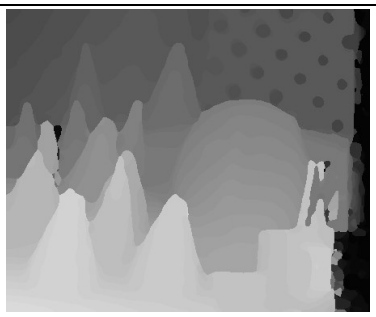
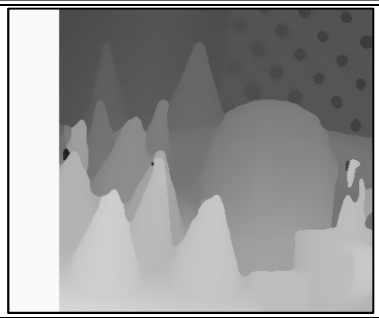
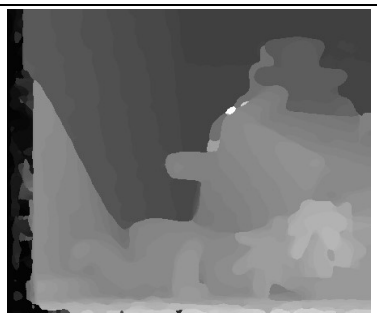
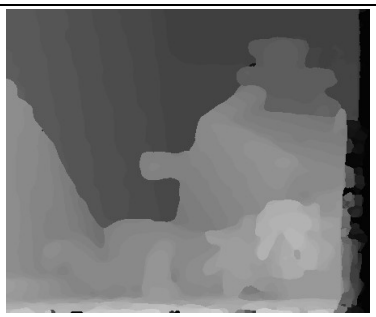
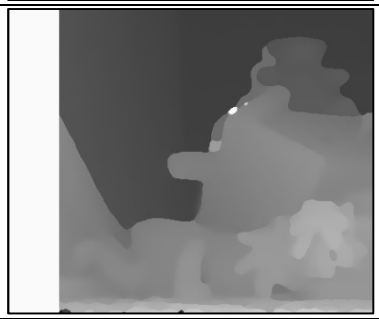
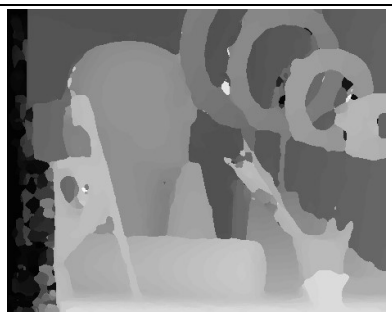
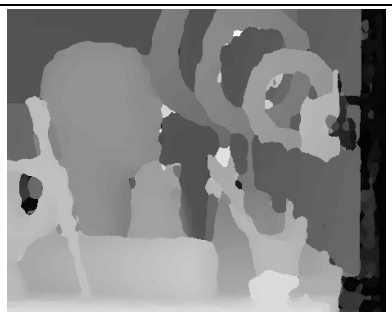
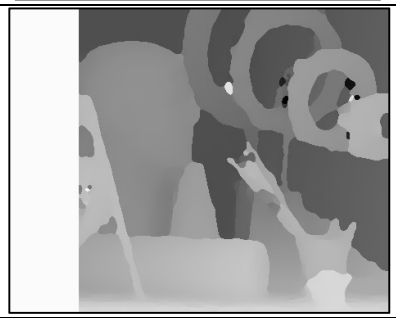
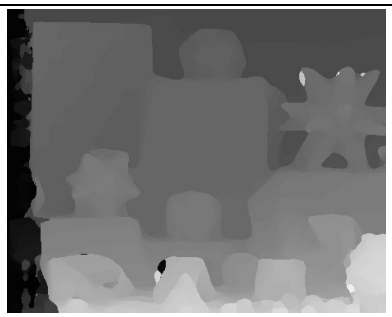
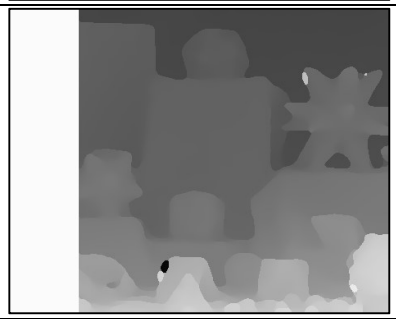
P2.4. mc-cnn Middlebury fast slike dispariteta

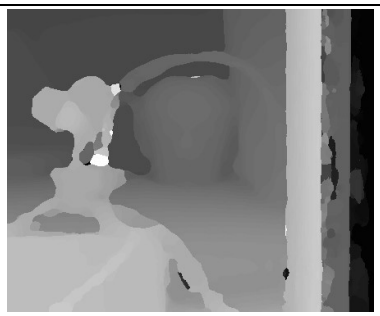
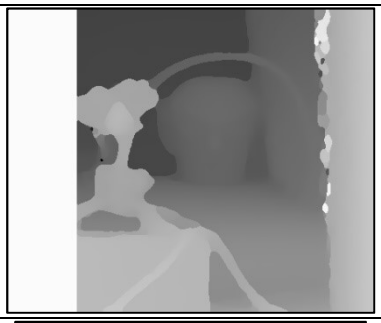
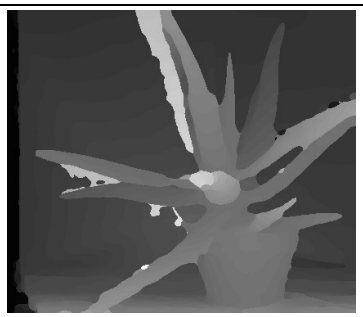
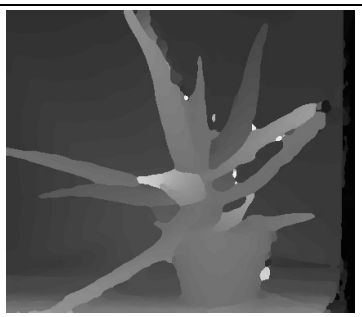
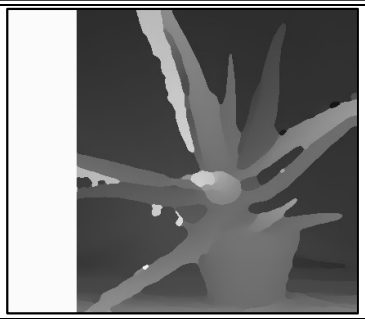
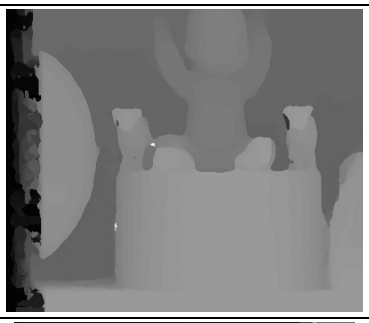
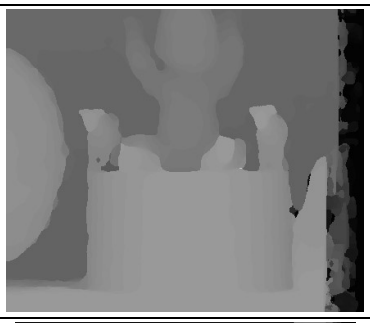
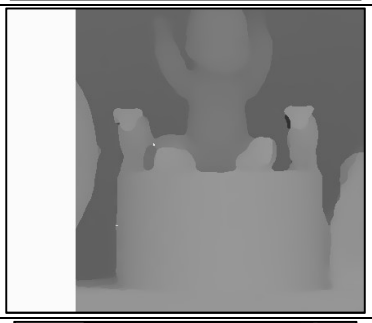
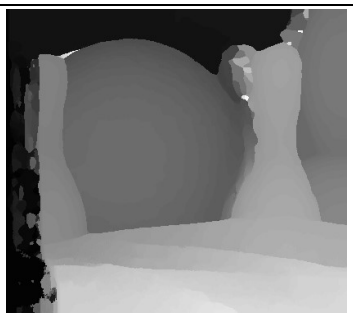
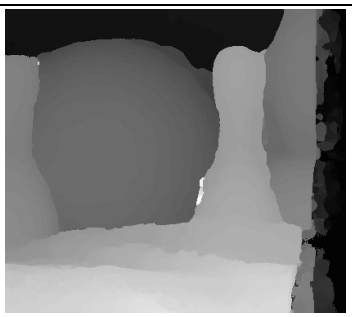
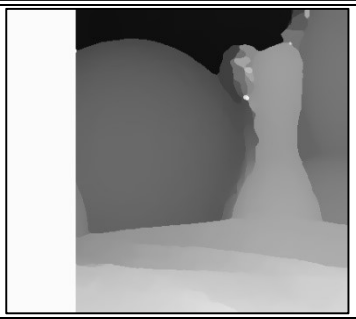
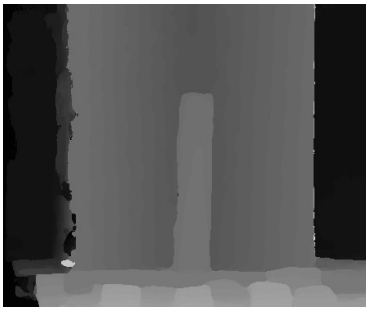
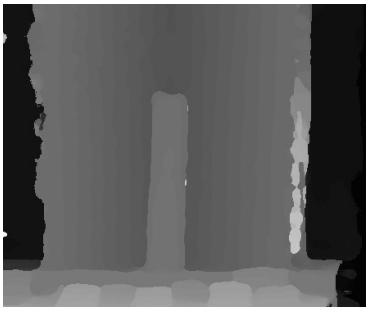
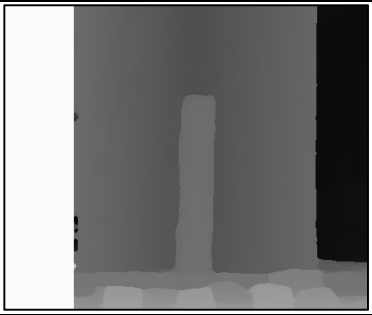
Naziv slike	Lijeva slika	Desna slika	Slika dispariteta
Tsukuba			
Cones			
Teddy			
Art			
Moebius			

Reindeer			
Aloe			
Baby3			
Bowling1			
Monopoly			

Tablica P2.4. Algoritam mc-cnn Middlebury fast– slike dispariteta

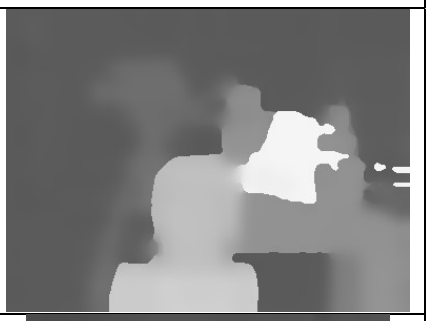
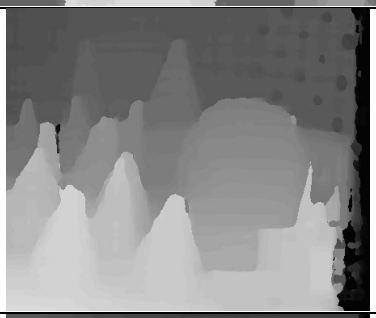
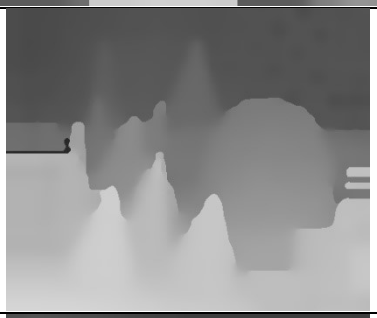
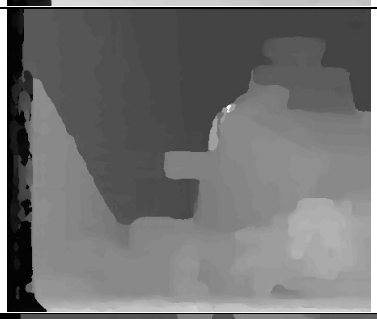
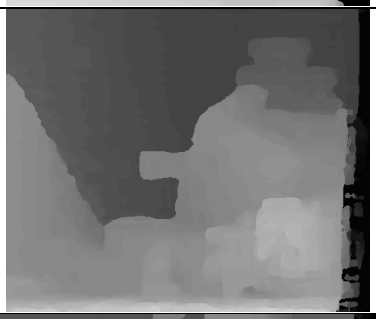
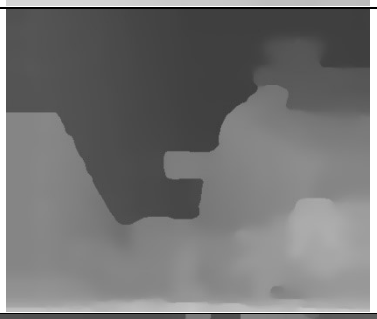
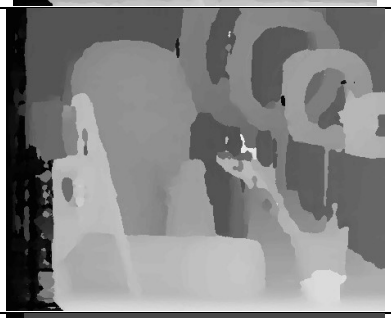
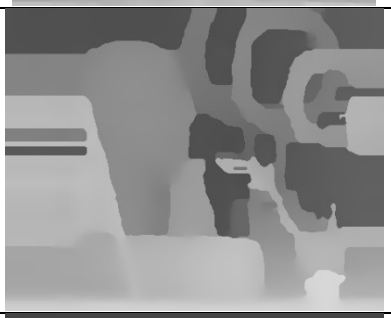
P2.5. mc-cnn Middlebury slow slike dispariteta

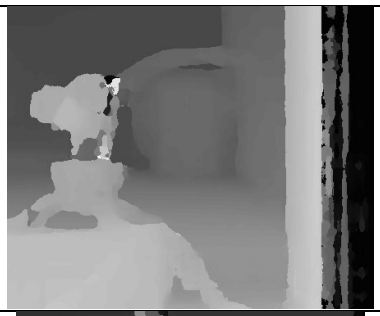
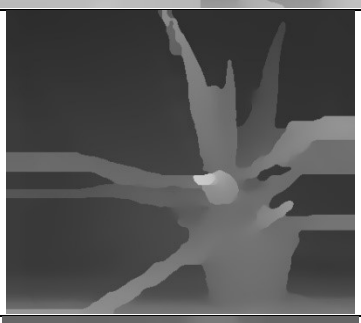
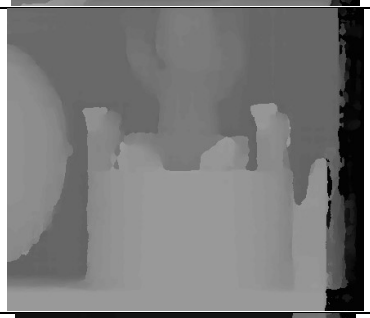
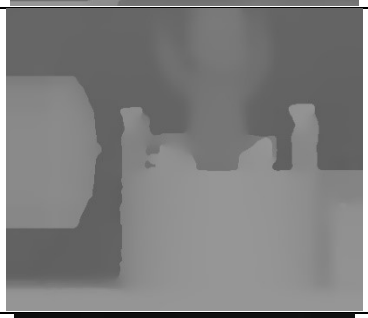
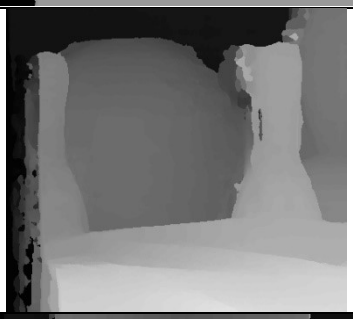
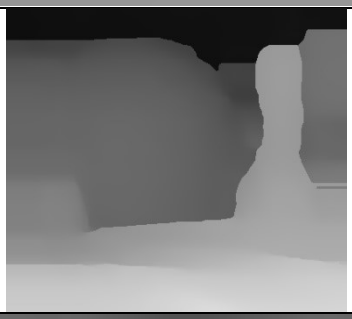
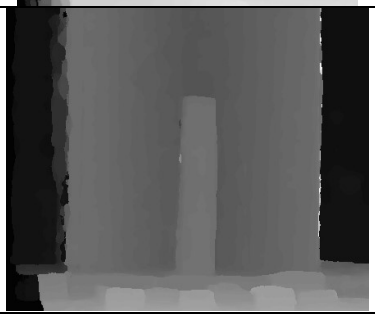
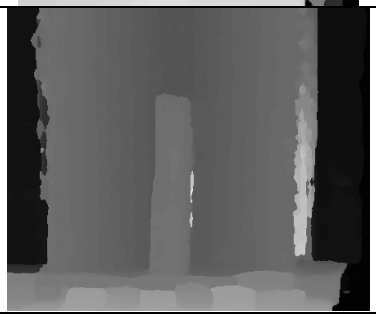
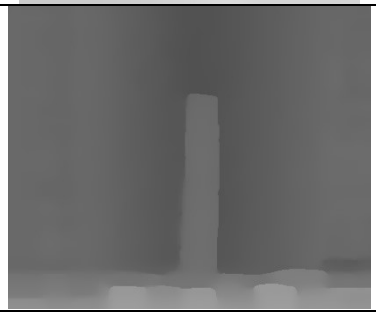
Naziv slike	Lijeva slika	Desna slika	Slika dispariteta
Tsukuba			
Cones			
Teddy			
Art			
Moebius			

Reindeer			
Aloe			
Baby3			
Bowling1			
Monopoly			

Tablica P2.5. Algoritam mc-cnn Middlebury slow – slike dispariteta

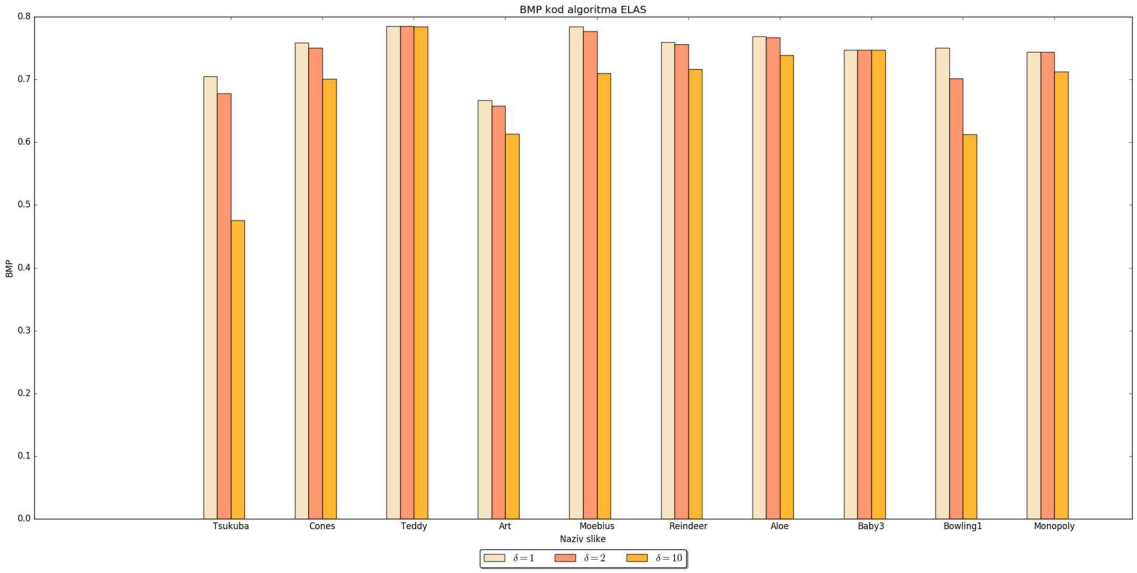
P2.6. mc-cnn KITTI slow slike dispariteta

Naziv slike	Lijeva slika	Desna slika	Slika dispariteta
Tsukuba			
Cones			
Teddy			
Art			
Moebius			

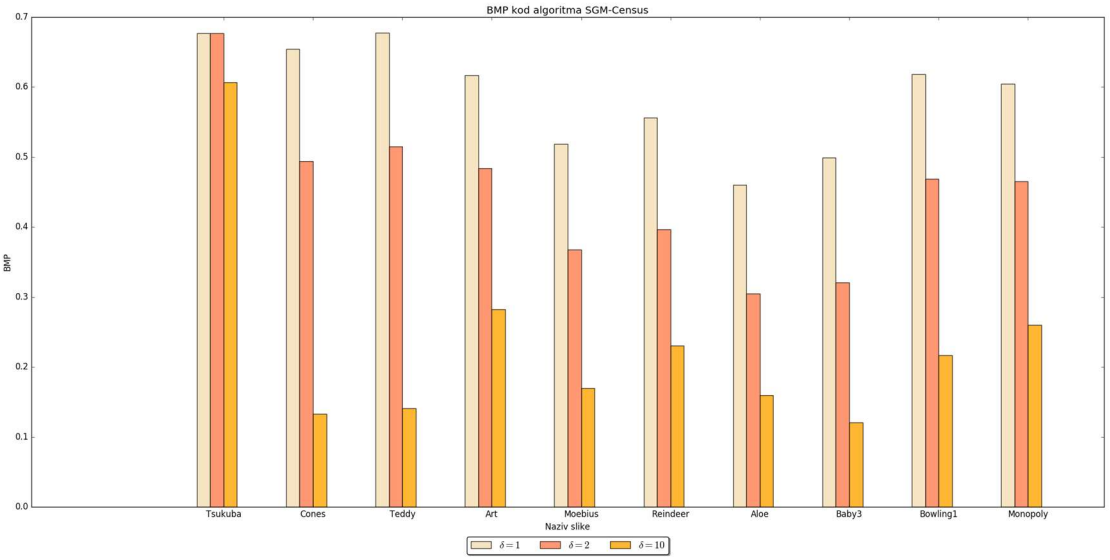
Reindeer			
Aloe			
Baby3			
Bowling1			
Monopoly			

Tablica P2.6. Algoritam mc-cnn KITTI slow– slike dispariteta

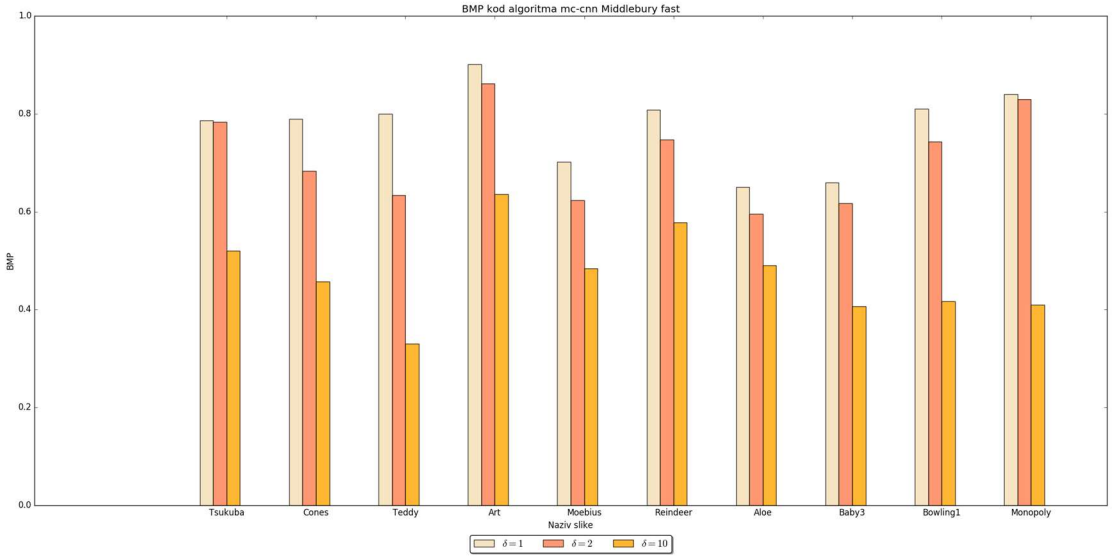
P3. Rezultati BMP metode po algoritmima



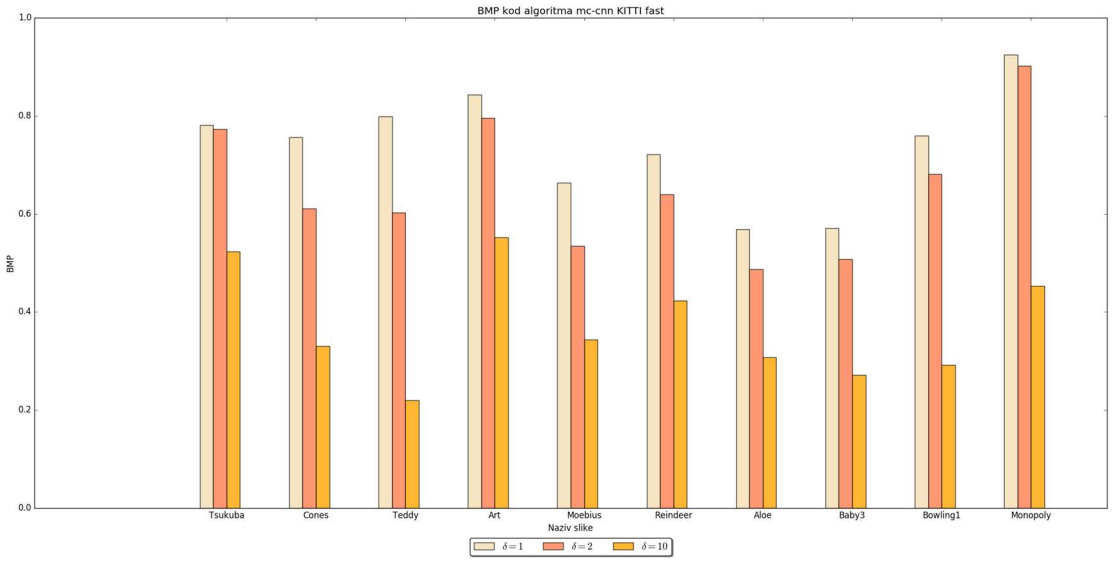
Slika P3.1. Algoritam ELAS – BMP rezultati po slikama



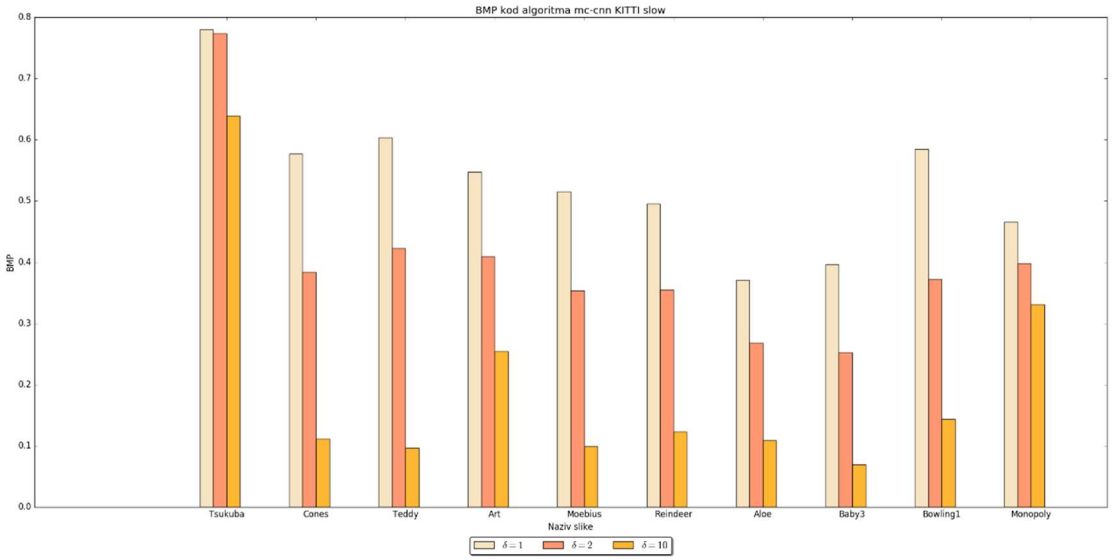
Slika P3.2. Algoritam SGM-Census – BMP rezultati po slikama



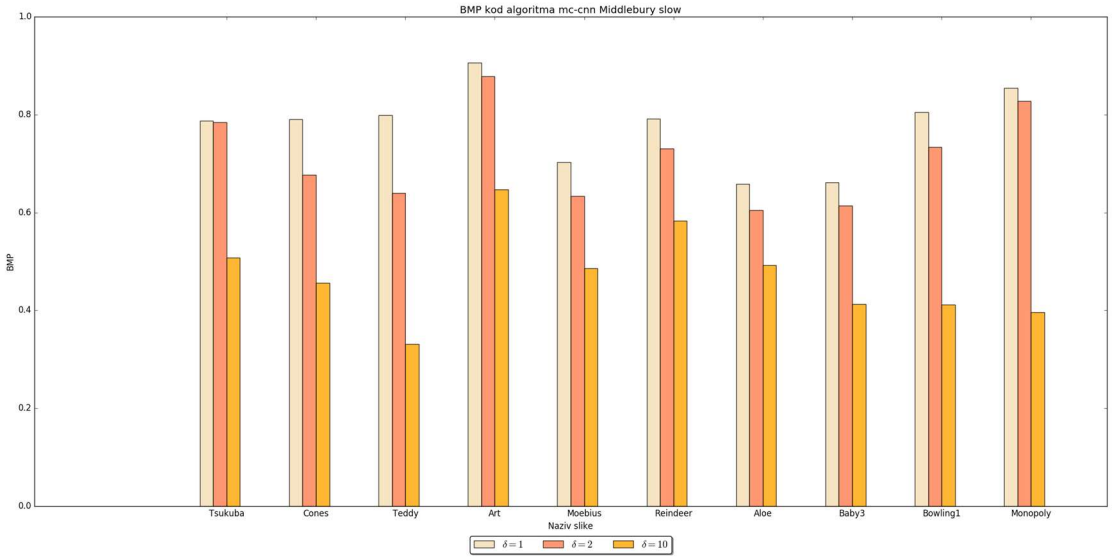
Slika P3.3. Algoritam mc-cnn Middlebury fast – BMP rezultati po slikama



Slika P3.4. Algoritam mc-cnn KITTI fast – BMP rezultati po slikama



Slika P3.5. Algoritam mc-cnn KITTI slow – BMP rezultati po slikama



Slika P3.6. Algoritam mc-cnn Middlebury slow – BMP rezultati po slikama