Detekcija vrhova slike

Pejković, Ana

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij elektrotehnike

DETEKCIJA VRHOVA SLIKE

Završni rad

Ana Pejković

Osijek, 2018.

SADRŽAJ

1. UVOD

Učinkovito otkrivanje zanimljivih značajki ključan je korak za različite zadatke u računalnom vidu kao što su praćenje, simultano lociranje i mapiranje, lokalizacija, podudaranje slike i prepoznavanje [1]. Kutovi su periferne značajke jer, za razliku od značajki temeljenih na boji ili rubovima, su ograničeni u dvije dimenzije te postoje brži algoritmi za njihovo otkrivanje koji imaju veliku točnost.

Ta je potreba pokrenula razvoj velikoga broja kutnih detektora [2]. Unatoč masivnom povećanju računalne snage od osnutka kutnoga detektora još uvijek stoji da kada se obrađuje videozapis uživo pri punoj brzini frekvencija okvira postojeći detektori ostavljaju malo,ako i ikako,vremena za daljnju obradu usprkos posljedicama *Mooreovoga* zakona.

S obzirom na kutove kao križanje (presjek, raskrižje) dva ruba (oboda), ove značajke nemaju prostorni nastavak (produžetak) i, stoga, nema podudarnosti (dvosmislenosti) u njihovoj lokaciji [1].

Na početku ćemo definirati značenje riječi kut [1]. To je mjesto gdje se sijeku dva pravca, također, kut je točka za koju postoje dva dominantna i drugi rubni pravci u lokalnom susjedstvu točaka. Kutovi se obično detektiraju i usklađuju u bazi podataka stoga je važno da se istovjetne točke u stvarnom vremenu više puta otkriju iz više prikaza.

Nekoliko različitih algoritama otkrivanja kuta poznati su u literaturi. Svi pokušavaju pronaći rješenje za učinkovito, precizno i pouzdano otkrivanje kuta [1]. Algoritam koji će biti korišten u ovom radu je *Features from Accelerated Segment Test* (FAST).

1.1. Zadatak rada

Zadatak ovog rada je pojasniti način rada algoritama FAST koji pripada bibliotekama *OpenCV-a*. Pokazati i objasniti kako on radi te priložiti i pojasniti rezultate eksperimenta. Izložiti prednosti i nedostatke ovakvog načina identifikacije ovog algoritma.

2. OPĆENITO O ZNAČAJKAMA

Nema univerzalne ili točne definicije što predstavlja značajku, a točna definicija često ovisi o problemu ili vrsti aplikacije.

Budući da se značajke upotrebljavaju kao početna točka i glavne osnove za naknadne algoritme, ukupni algoritam će često biti jednako dobar kao i njegov detektor značajki.
Prema tome, poželjno svojstvo detektora značajki je ponovljivost: hoće li se istovjetna značajka detektirati u dvije ili više različitih slika iste scene.

2.1. Detekcija značajki

Govoreći o računalnom vidu i obradi slika, detekcija značajki slike uključuje metode za dobivanje informacija iz slike i donošenje odluka o tome postoji li određena značajka u nekoj točki slike ili ne [3]. Dobivene značajke bit će podskupovi domene slike, često u obliku izoliranih točaka, kontinuiranih krivulja ili povezanih regija.

U ovom poglavlju ćemo prikazati usporedbu performansi detektora značajki i metoda deskriptora, pošto se njihove različite kombinacije koriste za podudaranje slike [4]. Podudaranje slika ima širok raspon primjena u stvarnom svijetu poput prepoznavanja objekta i lica. Pored toga, podudaranje slike glavna je operacija za dobivanje semantičkih informacija. S druge strane, podudaranje slika i dalje je izazovni problem za aplikacije u stvarnom vremenu zbog količine podataka koji se obrađuju.

2.1.1. Detekcija rubova

Rubovi su točke gdje postoji granica (ili rub) između dvije regije slike. Općenito, rub može biti gotovo proizvoljnog oblika i može uključivati spojeve [3].
U praksi se rubovi obično definiraju kao skup točaka na slici koji imaju jak gradijent veličine. Nadalje, neki uobičajeni algoritmi će zatim povezati visoku gradijentnu točku kako bi oblikovali potpuniji opis ruba. Ti algoritmi obično postavljaju neka ograničenja na svojstva ruba, kao što su oblik, glatkoća i vrijednost gradijenta.
Na netrivijalnim slikama rubovi su često pogrešno detektirani, odnosno rubne krivulje nisu povezane jer nedostaju rubni segmenti. [5]

2.1.2. Detekcija kutova

Pojmovi kutovi i točke interesa koriste se pomalo izmjenjivo i odnose se na točke slične osobine na slici, koje imaju lokalnu dvodimenzionalnu strukturu [3].

Naziv "Kut" nastao je od ranih algoritama koji su najprije obavili detekciju ruba, a zatim analizirali rubove kako bi pronašli brze promjene u smjeru (uglovima).

Uočeno je da su ,takozvani, kutovi također otkriveni na dijelovima slike koji nisu tradicionalno u uglovima (primjerice može se otkriti mala svijetla točka na tamnoj podlozi). Ove točke su često poznate kao interesne točke, ali izraz "kut" koristi se tradicija.

Jedna od stavki koja određuje kvalitetu detektora kuta je njegova sposobnost otkrivanja istog kuta na više sličnih slika koje se razlikuju u svjetlini, rotaciji kuta i sl. Neki od poznatih detektora kutova su *Harris* detektor [14] i SUSAN [14], *Moravec* algoritam [15] i dr.

2.1.3. Detekcija *blobova*

Blokovi pružaju komplementarni opis strukture slike u smislu regija, za razliku od kutova koji su više točkasti [3]. Ipak, *blob* deskriptori često mogu sadržavati željenu točku (lokalni maksimum odziva operatora ili centar gravitacije), što znači da se mnogi *blob* detektori mogu ,također, smatrati operatorima interesa. Postoje dvije glavne metode detekcije *bloba* s obzirom na područje koje će se ispitivati, a to su: diferencijalne metode koje se temelje na izvodima funkcije s obzirom na poziciju i metode koje se temelje na pronalaženju lokalnih maksimuma i minimuma funkcije [6]. Treba uzeti u obzir smanjenje slike te onda vršiti detekciju kutova. Detektor će reagirati na točke koje su oštre na smanjenoj slici, ali mogu biti glatke na izvornoj slici. U ovom trenutku razlika između kutnog detektora i *blob* detektora postaje pomalo nejasna. U velikoj mjeri, ova se razlika može otkloniti uključivanjem odgovarajućeg mjerila [3].

2.1.4. *Ridge* detekcija

Za izdužene predmete, pojam grebena je prirodni alat. Deskriptor ridge izračunat iz slike sive razine može se promatrati kao generalizacija središnje osi ,tj. može promatrati kao dopuna interesnim točkama ili lokalnim ekstremnim točkama [8]. Praktično gledajući, greben se može smatrati jednodimenzionalnom krivuljom koja predstavlja os simetrije, a također može imati i atribut lokalne grebenske širine povezane sa svakom točkom grebena. Nažalost, ipak je algoritamski teže izvući značajke grebena iz općih klasa sivih slika nego rubnih, kutnih ili *blobnih* značajki [3]. Nažalost, ipak je algoritamski teže izvući značajke grebena iz općih klasa sivih slika nego rubnih, kutnih ili *blobnih* značajki. Ipak, deskriptori grebena često se koriste za vađenje ceste u zračnim snimkama i za vađenje krvnih žila u medicinskim slikama.

1.1 Deskriptori značajki

Nakon otkrivanja značajki, slijedi povezivanje. Odrediti koje značajke dolaze iz odgovarajućih lokacija na različitim slikama [3]. U mnogim će se slučajevima lokalna pojava značajki mijenjati u orijentaciji, skali, pa čak i *Affinom* okviru između slika. Postoji više deskriptora od kojih su najznačajniji: MOPS, SIFT, PCA-SIFT, GLOH i SURF.

1.1.1 Scale Invariant Feature Transform (SIFT)

Da bi se postiglo pouzdano prepoznavanje, važno je da se značajke izvučene iz slike za testiranje mogu otkriti čak i kod promjene slike, buke i osvjetljenja. Takve točke obično leže na visoko kontrastnim područjima slike, kao što su rubovi objekta [9]. Slično tome, značajke koje se nalaze u artikuliranim ili fleksibilnim objektima obično ne rade ako se dogodi bilo kakva promjena u njihovoj unutarnjoj geometriji između dvije slike u skupu koji se obrađuju. Međutim, u praksi SIFT otkriva i koristi mnogo veći broj značajki sa slike, što smanjuje doprinos pogrešaka uzrokovanih tim lokalnim varijacijama. SIFT može robusno identificirati objekte čak i između nereda (eng. *clutter*) i djelomično sakrivenih objekata (eng. *partialocclusion*), jer je SIFT deskriptor značajki nepromjenjiv prema uniformnom skaliranju, orijentaciji, promjenama osvjetljenja i djelomično nepromjenjivim afinitetnim izobličenjima [10]. Za podudaranje i prepoznavanje slike SIFT značajke prvo se izdvajaju iz skupnih referentnih slika i pohranjuju u bazi podataka [9]. Nova slika uspoređuje se pojedinačno usporedbom svake značajke s nove slike u ovu prethodnu bazu podataka i utvrđivanjem karakteristika podudaranja kandidata na temelju euklidske udaljenosti njihovih značajki vektora. Od cijelog niza podudaranja

identificirani su podskupovi ključnih točaka koji se slažu oko objekta i njegovoga položaja,skale i orijentacije na novoj slici [10]. Određivanje dosljednih grupa se brzo provodi korištenjem učinkovite implementacije tablice generalizirane *Houghove* transformacije. Svaki skup od tri ili više značajki koje se slažu s objektom i njegovim položajem podliježu daljnjoj detaljnoj verifikaciji modela, a naknadno se odbacuju *outlieri*. Vjerojatnost da određeni skup značajki ukazuje na prisutnost objekta se računa uzimajući u obzir broj točnih podudaranja kao broj mogućih lažnih podudaranja [9]. Podudarni objekti koji prolaze sve te testove mogu se prepoznati kao točni s visokim povjerenjem.

Ovo su glavne faza računanja koja se koriste za generiranje skupa značajki slike koristeći *Scale Invariant Feature Transform* (SIFT) algoritam:

1. Detekcija ekstremnog mjerenja prostora: prva faza računanja traži sva mjerila i lokacije slika. Ona se učinkovito primjenjuje pomoću razlike u Gaussovoj funkciji kako bi se identificirale potencijalne interesne točke koje su nepromjenjive po skali i orijentaciji.
2. Lokalizacija ključnih točaka: na svakom mjestu kandidata, detaljan je model za određivanje lokacije i ljestvice. Ključne točke su odabrane na temelju mjera njihove stabilnosti

Određivanje orijentacije: jedna ili vise orijentacija se dodjeljuje svakoj lokaciji ključne točke temeljene na smjeru lokalnog gradijenta slike. Sve buduće operacije se obavljaju na slikovnim podacima koji su transformirani u odnosu na dodijeljenu orijentaciju, skalu i mjesto za svaku značajku, čime se osigurava invariancija tih transformacija.

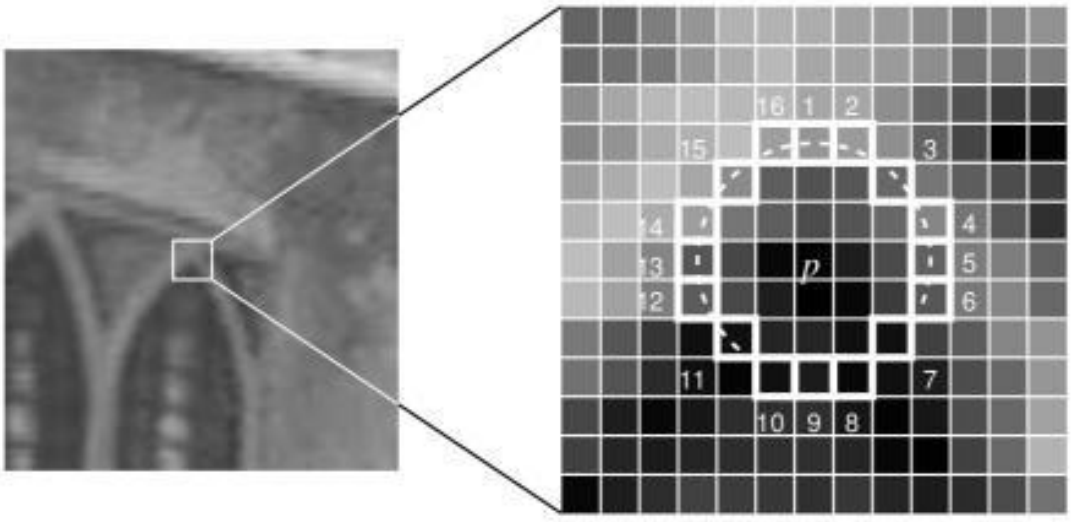
3. Deskriptor ključnih točaka: lokalni gradijenti slike se mjere na odabranoj skali u regiji oko svake ključne točke. One se pretvaraju u prikaz koji omogućuje značajne razine lokalnog oblikovanja i promjene u osvjetljenju.

Važan aspekt ovog pristupa je da generira velik broj značajki koje gusto prikazuju sliku preko cijelog raspona vage i lokacija [9]. Tipična slika veličine 500x500 piksela dovest će do oko 2000 stabilnih značajki (iako taj broj ovisi o slikovnom sadržaju i izboru različitih parametara). Količina značajki osobito je važna za prepoznavanje objekta, pri čemu sposobnost detekcije malih objekata u pretrpanim pozadinama zahtijeva da se najmanje tri značajke ispravno uspoređuju sa svakog objekta radi pouzdanog identifikacije podataka.

3. FEATURES FROM ACCELERATED SEGMENT TEST (FAST)

FAST, algoritam za detekciju kutova računalno je veoma učinkovit te je mnogo brži od ostalih algoritama kao što su SIFT,SUSAN i *Harris* algoritmi [11].

Kriterij ispitnog segmenta djeluje s obzirom na krug od šesnaest piksela oko kutu kandidat *p* [12]. Izvorni detektor klasificira p kao kut ako postoji skup od *n* susjednih piksela u krugu koji su svi svjetliji od intenziteta piksela kandidata I_p plus praga *t*, ili svi tamnije od Ip minus praga *t*, kao što je prikazano na slici 3.1. U početku je odlučeno da n bude dvanaest jer omogućuje testove velikih brzina koji se mogu upotrijebiti za isključivanje velikog broja ne-kutova. Testovi velikih brzina ispituju piksele 1 i 9. Ako su oba piksela unutar praga t od intenziteta I_p , tada piksel *p* ne može biti kut. Ako *p* još uvijek može biti kut, pregledavaju se pikseli 5 i 13. Ako je *p* kut, tada barem tri od njih moraju biti svjetliji od I_{p+t} ili tamnija od I_{p-t} [12]. Ako niti jedan od njih nije slučaj, onda *p* ne može biti kut. Cijeli kriterij ispitnog segmenta može primijeniti na preostale kandidate pregledavajući sve piksele u krugu.



Slika 3.1. *12 istaknutih kutova u susjedstvu testiranog segmenta slike* [12]

Ovaj detektor je pokazao velike preformanse, ali ima i nekoliko nedostataka:

1) Testovi velikih brzina ne odbacuju mnogo kandidata (mogućih kutova) za $n < 12$ jer točka može biti kut ako su samo dva od četiri piksela značajno svjetlija ili oba značajno tamnija od p (uz pretpostavku da su pikseli susjedni) [11]. Potrebno je i dodatno ispitivanje ako se cijeli test treba izvesti za svijetli prsten ili tamni prsten.
2) Efektivnost detektora ovisit će o redoslijedu zahtjeva i raspodjeli kutnih pojava. Malo je vjerojatno da je ovaj izbor piksela optimalan.
3) Višestruke značajke otkrivene su jedna uz drugu.

3.1. Poboljšanje cjelokupnoga strojnoga učenja i njegove brzine

Proces se odvija u dvije faze. Prvo, za izgradnju detektora kuta za određeni n, svih 16 piksela prstena se izdvajaju iz skupa slika (po mogućnosti iz ciljne domene aplikacije) [11]. One su označene koristeći jednostavnu provedbu kriterija za ispitivanje segmenta za n i prikladan prag.

Za svako mjesto u krugu $x \in \{1 \dots 16\}$, piksel u tom položaju u odnosu na p označen je po $p \rightarrow x$, može imati jednu od tri stanja:

$$S_{p \rightarrow x} = \begin{cases} d, & I_{p \rightarrow x} \leq I_p - t \quad \text{(tamnije)} \\ s, & I_p - t < I_{p \rightarrow x} < I_p + t \quad \text{(istovjetan)} \\ b, & I_p + t \leq I_{p \rightarrow x} \quad \text{(tamniji)} \end{cases} \tag{3-1}$$

Neka P bude skup svih piksela u svim treninzima. Odabiremo x particija P na tri podskupine, P_d, P_s i P_b, gdje:

$$P_b = \{p \in P : S_{p \rightarrow x} = b\}, \tag{3-2}$$

P_d i P_s su slično određeni. Drugim riječima, dani izbor x se koristi za particioniranje podataka u tri skupine [11]. Set P_d sadrži sve točke u kojima je piksel x tamniji od središta piksela za prag t,

P_b sadrži točke svjetlije od piksela središta *t*, a sadrži P_s preostale točke gdje je piksel *x* sličan pikselu središta. Neka K_p bude booleovska varijabla koja je istinita ako je *p* kut ,a inače laž.

Faza 2

koristi algoritam koji se koristi u ID3 i počinje odabirom *x* koji daje najviše informacije o tome je li piksel kandidat kut, mjeren entropijom K_p [12].

Ukupna entropija *K* za proizvoljan skup uglova, Q, je:

$$H(Q) = (c + \bar{c}\,)\, log_2(c + \bar{c}\,) - c\, log_2\, c - \bar{c}\, log_2\, \bar{c} \qquad (3\text{-}3)$$

gdje je $c = \left| \{ i \in Q : K_i \text{ je istina} \} \right|$ (broj kutova)

i $\bar{c} = \left| \{ i \in Q : K_i \text{ je laž} \} \right|$ (broj bez kutova)

Izbor x tada daje dobitak informacija *(Hg):*

$$Hg = H(P) - H(P_d) - H(P_s) - H(P_b) \qquad (3\text{-}4)$$

Nakon što odaberete *x* koji daje najviše informacija, proces se rekurzivno primjenjuje na sva tri podskupa, tj. xb je odabran za podjelu P_bu $P_{b,d}$, $P_{b.s}$,$P_{b,b}$, X_S je odabran za podjelu P_s u$P_{s,d}$, $P_{s,s}$, $P_{s,b}$i tako dalje, gdje je svaki *x* izabran da daje maksimalne informacije o skup na koji se primjenjuje [12]. Rekurzivni proces završava kada je entropija podskupa nula. To znači da svi *p* u ovom podskupu imaju istu vrijednost K_p-a, tj. oni su svi kutovi ili su svi ne-kutovi. To se može zajamčiti jer *K* je točna funkcija podataka. Ukratko, ovaj postupak stvara stablo odluke koje može ispravno klasificirati sve kutove vidljive na setu podataka za treniranje kutova i stoga (u bliskoj aproksimaciji) ispravno utjelovljuje pravila odabranog FAST detektora kuta [12]. U nekim slučajevima, dva od tri podstabla mogu biti ista

U ovom slučaju, boolean test koji ih odvaja se uklanja. Ovo drvo odluke zatim se pretvara u C kod, stvarajući dugi niz ugniježđena if-else izjava koje se sastavljaju i koriste kao kutni detektor. Za daljnju optimizaciju, mi forsiramo x_b,x_d *i* $x_s$da budu jednaki. U ovom slučaju drugi testirani piksel je uvijek isti. Budući da je to slučaj, test na prvom i drugom pikselu može biti izvedena u seriji. To omogućuje da se prva dva testa provode paralelno za traku piksela korištenjem vektorskih uputa prisutnih na mnogim mikroprocesorima visokih performansi. Budući da je većina točaka odbijena nakon dva testa, to dovodi do značajnog povećanja brzine. Budući da

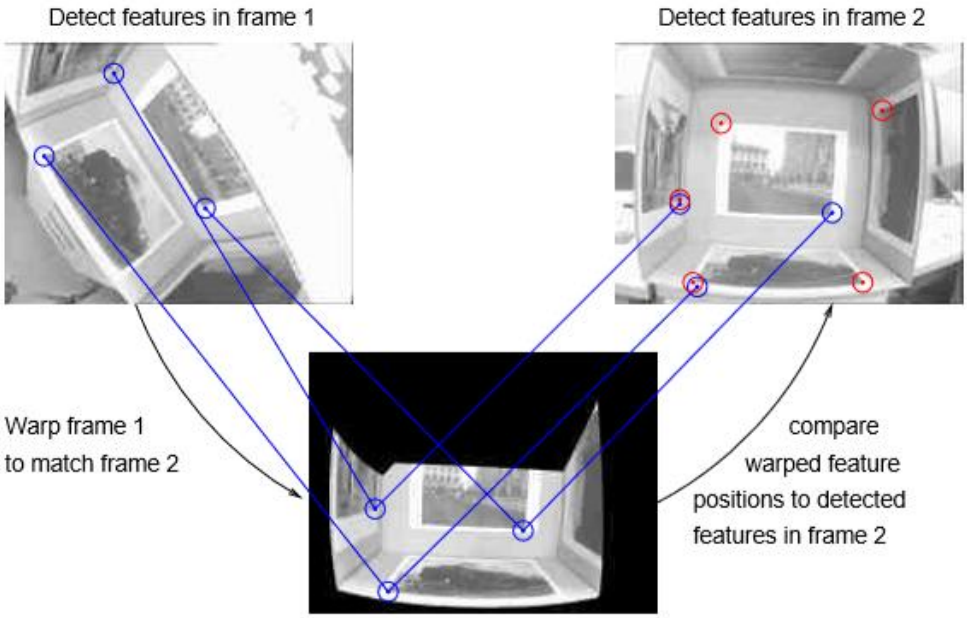
podaci sadrže nepotpunu pokrivenost svih mogućih kutova,treba uzeti u obzir da naučeni detektor nije baš isti kao i detektor testova segmenta.
U slučaju FAST detektora, jednostavno je uključiti pojavu svake moguće kombinacije piksela (postoji 316 = 43,046,21 kombinacija) s niskom težinom kako bi se osiguralo da detektor točno izračunava segment testiranja.

3.2. Ne-maksimalno suzbijanje

Budući da test segmenta ne izračunava funkciju kutnog odziva, ne-maksimalno suzbijanje (eng. *non-maximal suppersesions*) ne može se izravno primijeniti na rezultirajuće značajke. Za određeni *n*, kada povećavamo *t,* broj otkrivenih kutova će se smanjiti. Budući da *n* = 9 proizvodi najbolje rezultate ponovljivosti, varijacije u *n* neće se razmatrati.
Snaga kuta (eng. *the corner strength*) je stoga definirana kao maksimalna vrijednost *t* za koju se točka prepoznaje kao kut. Razvrstavanje stabla odluke može učinkovito odrediti klasu piksela za određenu vrijednost *t*. Klasa piksela (na primjer, 1 za kut, 0 za ne-kutu) je monotono smanjenje funkcije *t* [12]. Stoga, možemo upotrijebiti bisekciju kako bismo učinkovito pronašli točku u kojoj se funkcija mijenja od 1 do 0. Ova točka daje najveću vrijednost *t* za koju se točka prepoznaje kao kut. Budući da je *t* diskretan, to je binarni algoritam pretraživanja. Alternativno, može se koristiti shema iteracije. Piksel na prstenu 'prolazi' test segmenta ako nije unutar *t* centra. Ako dovoljan broj piksela ne uspije, točka neće biti klasificirana kao kut.
Detektor se pokreće, a od svih piksela koji prolaze test, nalazi se količina kojom prolaze [12]. Prag se potom poveća za najmanji od tih iznosa, a detektor se ponavlja. To povećava prag dovoljno da bi se osiguralo da se drugačiji put provede kroz stablo. Taj se proces ponavlja sve dok detekcija ne uspije. Budući da brzina ovisi jako o naučenom stablu i specifičnoj procesorskoj arhitekturi, niti jedna tehnika nema prednost konačnih brzina nad drugom tehnikom.

3.3. Mjerenje ponovljivosti detektora

Za par slika, značajka je "korisna" ako se ekstrahira na jednoj slici i može se potencijalno pojaviti na drugoj (tj. nije zatvorena). "Ponavljanje" je ,također, ako se otkrije u blizini iste *stvarne točke svijeta* (eng. *World point*) u drugoj. U svrhu mjerenja ponovljivosti ovo omogućava vise značajki na prvoj slici (na prvom prikazu) koje se podudaraju s jednom značajkom na drugoj slici (na drugom prikazu)



Sl. 3.2. *Provjeravanje ponovljivosti otkivanjem iste značajke u stvarnom svijetu u različitim pogledima* [12]

Ponovljivost, *R*, je definirana kao:

$$R = \frac{N_{ponavljanja}}{N_{korisnih}} \tag{3-5}$$

gdje se $N_{ponavljaja}$ i $N_{korisnih}$ zbrajaju preko svih parova slike u slijedu slika. To je ekvivalentno opterećenom težinskom prosjeku ponovljivosti za svaki par slika, pri čemu se opterećenost težine broji kao broj korisnih značajki [12]. U ovom smo radu općenito izračunali ponovljivost za određeni broj značajki po okviru, koji variraju između 0 i 2000 značajki (za sliku veličine 640 × 480).

To nam također omogućuje izračunavanje područja ispod krivulje ponovljivosti, *A*, kao ukupni rezultat. Mjerenje ponovljivosti zahtijeva mjesto i vidljivost svakog piksela na prvoj slici da bude poznat na drugoj slici. Da bismo ovo izračunali, upotrebljavamo model 3D površine prizora za izračunavanje ako i gdje se otkrivene značajke trebaju pojaviti u drugim pogledima.

Ovo je prikazano na slici 3.2.

Sl. 3.3. *Kutija podataka* [12]

To postaje vjerojatnije s promjenom gledišta i stoga promjene oblika kuta postanu velike. [12] Umjesto korištenja pouzdanih znakova, 3D model usklađen je s prizorom ručno, a zatim se optimizira pomoću mješavine simuliranog žarenja (eng. Simulated annealing) i sloja gradijenta kako bi se smanjio zbroj kvadratnih razlika (eng. Sum of squared difference) između svih parova okvira i ponavljanja. Za izračun SSD-a između okvira *i* i ponovnog izbora okvira *j*, položaj svih točaka u okviru *j* nalazi se u okviru *i*. [12] Slike su zatim pojašnjene. Visoke frekvencije se uklanjaju radi smanjenja buke, dok se niske frekvencije uklanjaju kako bi se smanjio utjecaj promjena rasvjete.

Da bi se poboljšala brzina sustava, SSD se izračunava samo pomoću 1000 slučajnih mjesta. Navedeni skupovi podataka prikazani su na slici 3 i na slici 4. S ovim skupovima podataka pokušali smo uhvatiti široki raspon geometrijskih i teksturnih tipova kuta.

3.4. Fast-er: povećanje ispitivanja

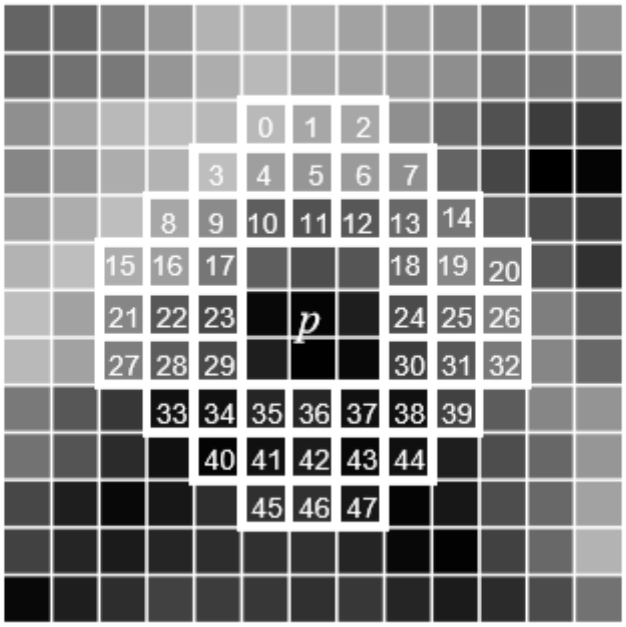
Budući da se detektor testova segmenta može prikazati kao trostruko stablo odluke s određenom ponovljivosti, detektor se može generalizirati određivanjem detektora značajki kao trostruko stablo odluke koja detektira točke s visokom ponovljivosti. [12] Ponovljivost takvog detektora je nekonveksna funkcija konfiguracije stabla, tako da optimaliziramo stablo pomoću simuliranog žarenja. To rezultira višenamjenskom optimizacijom. Ako je svaka točka otkrivena kao značajka, tada je ponovljivost trivijalno savršena. Također, ako je bez granica dopušten rast kompleksnosti stabla, tada je optimizacija sposobna pronaći jednu jedinu značajku u svakoj slici u setu podataka za testirane koja se ponavlja.

Slika 3.4. *Skup podataka o labirintu* [12]

Niti jedan od njih nije koristan rezultat. Kako bi se to utvrdilo, funkcija troška za stablo je definirana kao:

$$k = (1 + (\frac{w_r}{r})^2) (1 + \frac{1}{N} \sum_{i=1}^{N} \left(\frac{d_i}{w_n} \right)^2) (1 + (\frac{s}{w_s})^2) \qquad (3\text{-}6)$$

gdje *r* je ponovljivost (kako je definirano u (2 - 5)), di je broj otkrivenih kutova u okviru *i*, *N* je broj okvira, a *s* veličina (broj čvorova) stabla odluke [13]. Učinak ovih troškova kontrolira se pomoću w_r, w_n *i* w_s. Treba imati na umu da je za učinkovitost ponovljivost izračunata pri fiksnom pragu, za razliku od određenog broja značajki po okviru. Kutni detektor trebao bi biti nepromijenjen za rotaciju, osvjetljenje i intenzitet inverzije slike. Kako bi se spriječilo prekomjerno opterećenje algoritma za optimizaciju, svaki put kad se stablo procjenjuje, primjenjuje se šesnaest puta: pri četiri rotacije, u razmaku od 90°, sa svim kombinacijama refleksija i inverzija intenziteta. Rezultat je logička ILI aplikacija detektora: kut se otkriva ako bilo koja od šesnaest aplikacija stabla klasificira točku kao kut.



Slika 3.5. *Položaji pomaka u ubrzanom* detektoru [12]

Svaki čvor stabla ima pomak u odnosu na središnji piksel, *x*, s $x \in \{0 \ldots 47\}$ kako je prikazano na slici 5. Stoga $x = 0$ odnosi se na pomak (-1,4) [12]. Svaki list ima klasu *K*, s
0 za ne-kutove i 1 za kutove. Osim čvora korijena, svaki je čvor na *b*, *d* ili *s* ogranku svog roditelja, ovisno o ishodu testa koji vodi do te grane. Stablo je ograničeno, tako da list na *s* granama svog izravnog roditelja ima $K = 0$. To osigurava da se broj kutova općenito smanjuje dok se prag povećava. Simulirani optimizator žarenja pravi slučajne modifikacije stabla tako što prvo odabire čvor slučajno odabiraom, a zatim ga mutira. Ako je odabrani čvor

- list, a zatim s jednakom vjerojatnošću:
    1. Zamijenite čvor sa slučajnim podstablom dubine 1
    2. *Flip* klasifikacija čvora. Ovaj izbor nije dostupan ako je klasa listova ograničena.
       čvor, a zatim s jednakom vjerojatnošću, odaberite bilo koji od njih

- čvor,a zatim s jednakom vjerojatnošću,odabirete bilo koji od njih
    1. Zamijenite pomak sa slučajnom vrijednosti u rasponu 0 ... 47
    2. Čvor se zamijeni listom slučajnog razreda (ovisno o ograničenju) Uklonite slučajno odabranu granu čvora i zamijenite je kopijom druge slučajno odabrane grane tog čvora. Na primjer, *b* grana može biti zamijenjena kopijom grane *s*.

Nasumično uzgojeno podstablo sastoji se od jednog čvora za odlučivanje (s nasumičnim pomakom u 0 ... 47), i tri čvora lista [12]. S izuzetkom ograničenog lista, lišće ove slučajne podvrsta ima slučajne klase. Ove modifikacije stabla omogućuju rast, mutaciju i smanjenje stabla, respektivno.

Posljednja modifikacija stabla motivirana je našim opažanjima FAST-9 detektora. U FAST-9, velik broj čvorova ima svojstvo da su dva od tri podstabla identična. [12] Budući da FAST-9 pokazuje visoku ponovljivost, uključili smo ovu modifikaciju kako bismo FAST-ER lako mogli dozvoliti sličnu strukturu. Modifikacije se prihvaćaju prema kriteriju prihvaćanja *Boltzmanna*, gdje je vjerojatnost *P* prihvaćanja promjene na iteraciji *I*:

$$P = e^{\frac{\hat{k}_{I-1}-k_I}{T}} \tag{3-7}$$

gdje je $\hat{k}$ trošak nakon primjene kriterija prihvaćanja i *T* je temperatura. Temperatura slijedi eksponencijalni raspored:

$$T = \beta e^{-\alpha \frac{I}{I_{max}}} \tag{3-8}$$

gdje je I_{max} broj iteracija. Algoritam je inicijaliziran slučajno uzgojenim stablom dubine 1, a algoritam koristi fiksni prag, *t* [12]. Umjesto izvođenja jedne optimizacije, optimizator se ponavlja mnogo puta pomoću različitih slučajnih sjemena. Budući da detektor mora biti primijenjen na slike svake iteracije, svako kandidat stablo u svih šesnaest transformacija kompajlira se u strojni kod u memoriji i izvršava izravno. Budući da se primjenjuje sa šesnaest transformacija, rezultirajući detektor nije posebno učinkovit. Dakle, za učinkovitost, detektor se koristi za generiranje podataka o obuci kako bi se moglo generirati jedno stablo. Rezultirajuće stablo sadrži oko 30.000 čvorova koji nisu listovi.

4. EKSPERIMENTALNI DIO

4.1. OpenCV

OpenCV (*Open Source Computer Vision*) je biblioteka otvorenog koda koja se koristi za rad računanim vidom, a odnosi se na aplikacije koje rade s podacima u stvarnom vremenu [13]. Biblioteka sadrži preko 2500 optimiziranih algoritama koji obuhvaćaju gotovo sva područja računalne vizije i strojnog učenja. *OpenCV* je originalno pisan u programskom jeziku C budući da obavljanje velikog broja operacija i algoritama u stvarnom vremenu zahtjeva optimiziran kod koji je pisan na nižoj razini. Također, razvijena su sučelja i prema drugom jezicima više razine kao što su *Java, Python, Ruby* itd.

4.2. Eksperiment

Na pet slika je implementirano pet različitih algoritama: FAST, SIFT, SURF, ORB i BRISK. Cilj algoritma je bio detektirati vrhove. U nastavku će biti prikazani rezultati.

(a) FAST

(b) SIFT

(c) SURF

(d) ORB

(e) BRISK

(f) Originalna slika

Sl. 4.1. *Rezultat detekcije značajki na slici 1 iz seta slika. Slika prikazuje rezultate metoda (a-e) te originalnu sliku (f)*

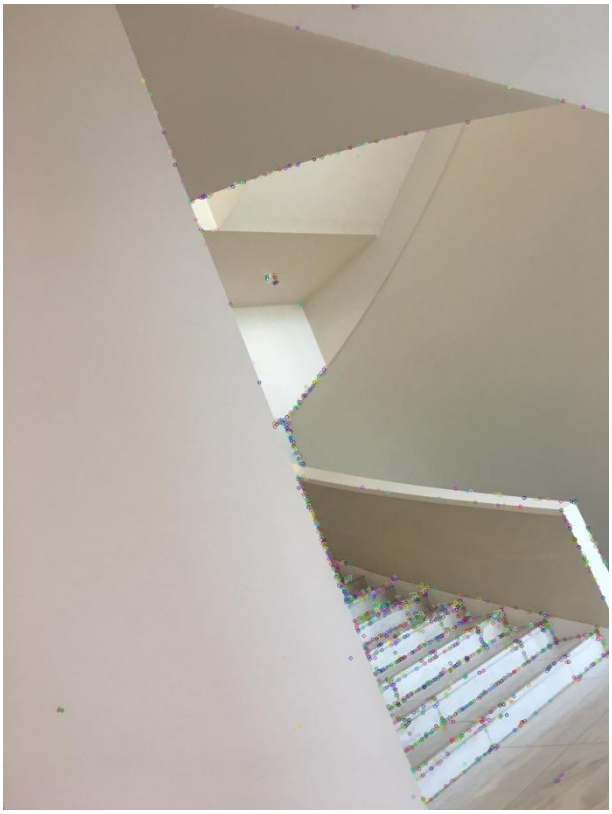
FAST pravi mali broj grešaka, a detektira veliki broj točaka koje označavaju rubove, odnosno vrhove. (Slika 4.1.a.)

Algoritmom SIFT se poneke detektirane točke nalaze u blizini rubova ili vrhova, odnosno nisu sve precizno detektirane. (Slika 4.1.b.)

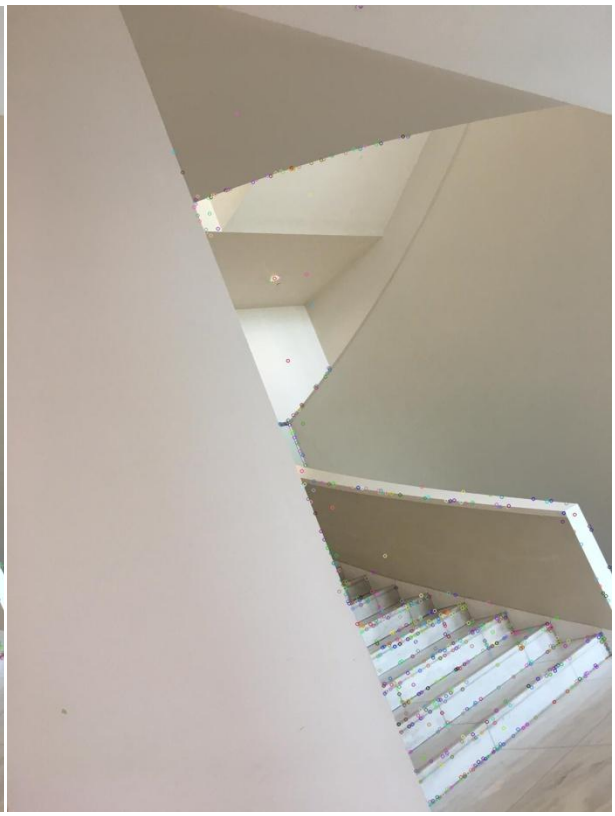
Na slici se vidi kako je algoritam SURF detektirano puno više vrhova u odnosu na BRISK, odnosno točke koje su detektirane su puno gušće, ali isto postoje pogrešne detekcije, budući da su detektirane i čisti dijelovi površine bez rubova. (Slika 4.1.c.)

Metodom ORB je detektiran dosta mali broj točaka, no detektirane točke su uglavnom isključivo vrhovi, a ne i rubovi. Pogrešnih detekcija, odnosno detekcija čiste površine gotovo da i nema. (Slika 4.1.d.)

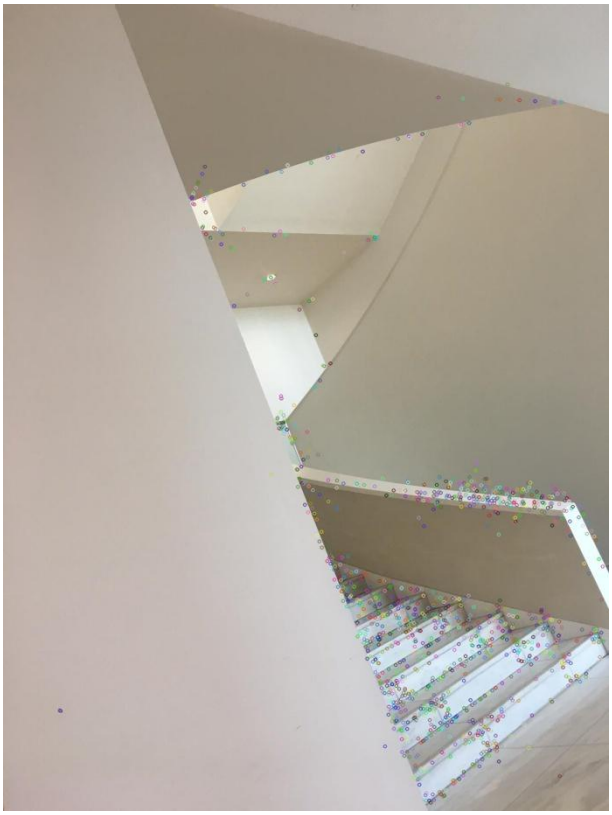
Može se vidjeti kako algoritam BRISK ne detektira mnogo vrhova, odnosno rubova, te su poneki detektirani krivo, odnosno detektira čistu površinu na kojoj nema niti vrhova, niti rubova. (Slika 4.1.e.)



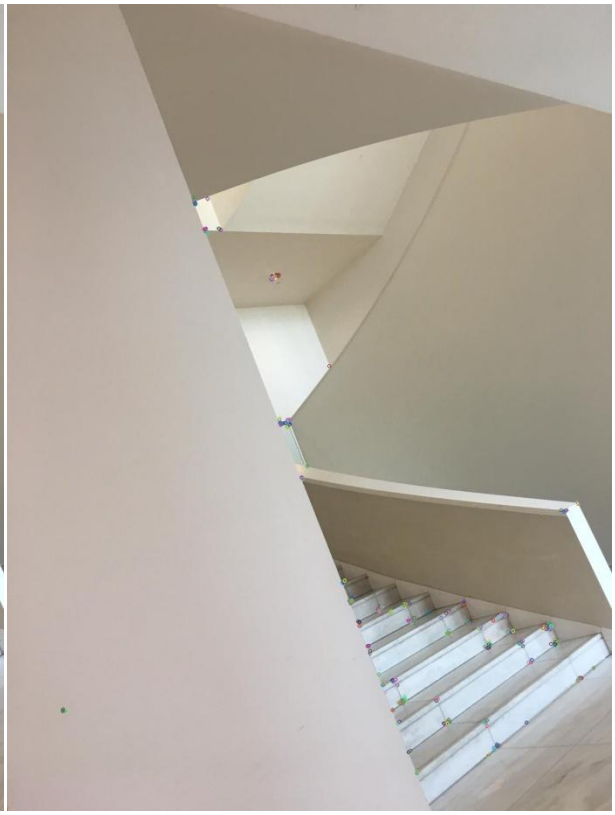
(a) FAST



(b) SIFT



(c) SURF

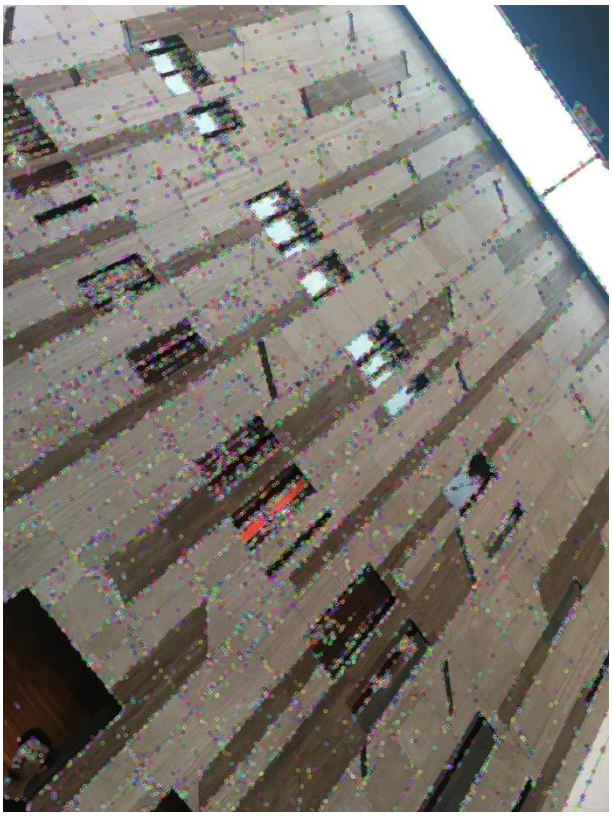


(d) ORB

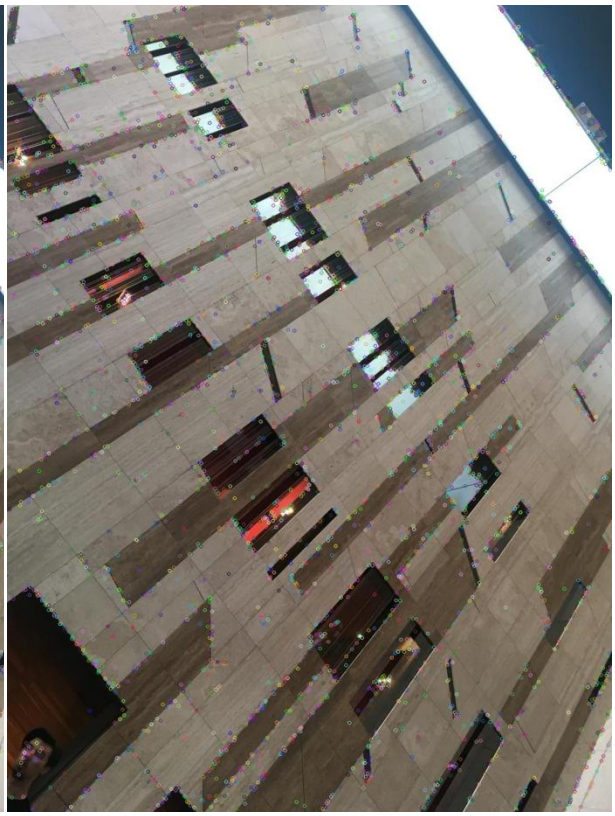
(e) BRISK (f) Originalna slika

Sl. 4.2*. Rezultat detekcije značajki na slici 2 iz seta slika. Slika prikazuje rezultate metoda (a-e) te originalnu sliku (f)*

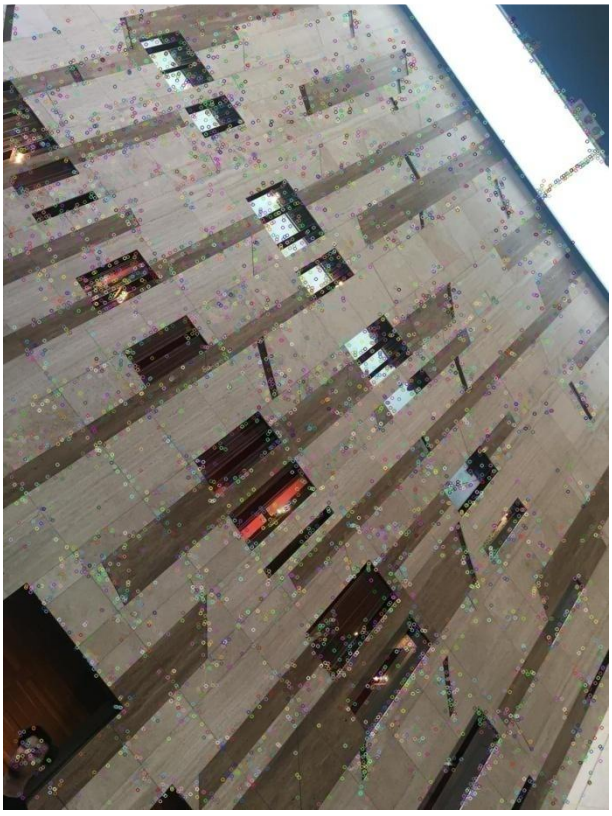
Kao i u prethodnom primjeru FAST detektira rubove ,tj. vrhove s vrlo velikom točnosti. (Slika 4.2.a.) SIFT detektira s malo većim stupnjem pogreške u odnosu na FAST ,ali i dalje je vrlo točan. (Slika 4.2.b.) Za razliku od FAST-a i SIFT-a, SURF detektira puno više vrhova pa tako raste i stupanj pogreške. (Slika 4.2.c.) Algoritam ORB je detektirao vrlo mali broj točaka te su to sve gotovo kutovi. (Slika 4.2.d.) BRISK je detektirao mali broj točaka ,ali su one uglavnom točno detektirane. (Slika 4.2.e.)



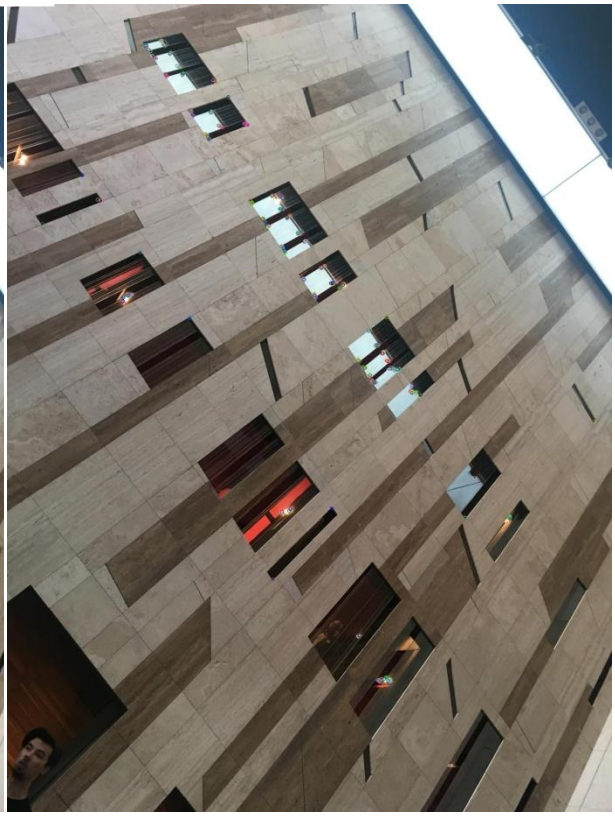
(a) FAST



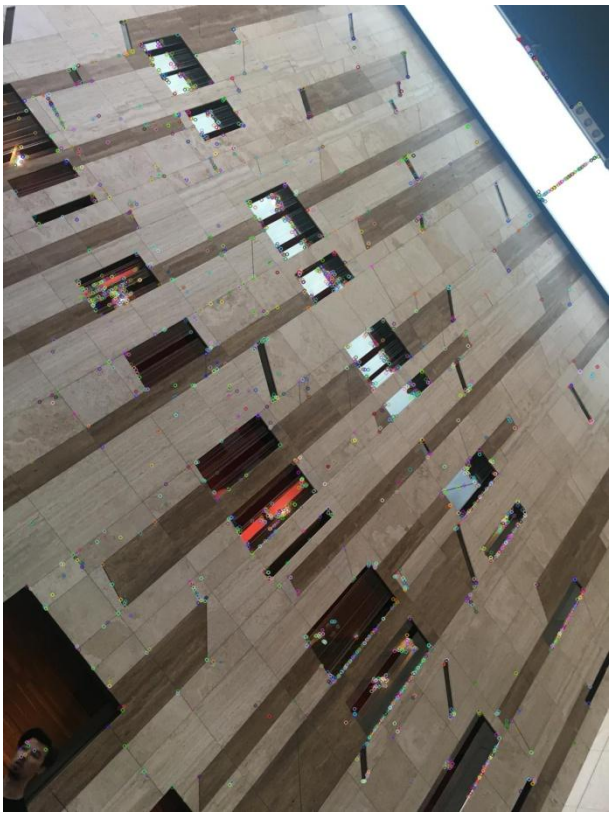
(b) SIFT



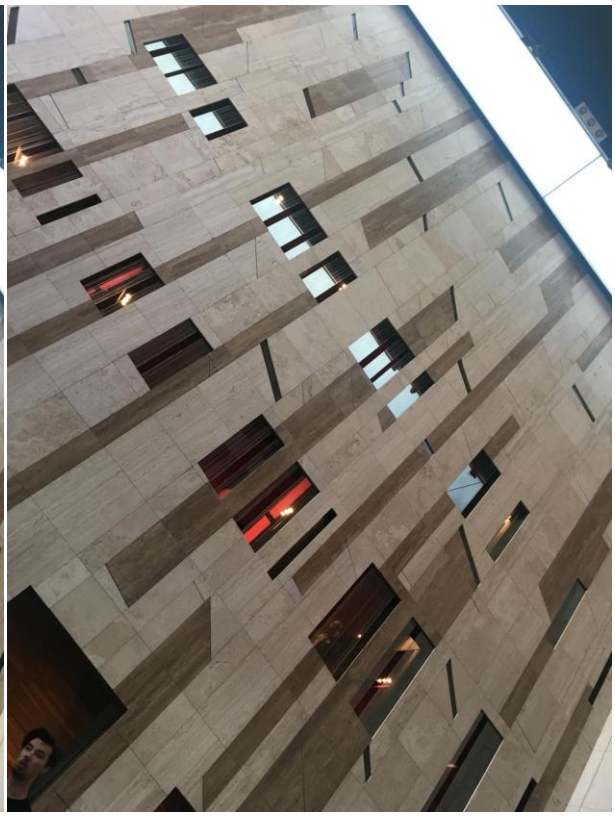
(c) SURF



(d) ORB



(e) BRISK



(f) Originalna slika

Sl. 4.3*. Rezultat detekcije značajki na slici 2 iz seta slika. Slika prikazuje rezultate metoda (a-e) te originalnu sliku (f)*

Kao i u prethodna dva primjera FAST se i ovdje pokazao iznimno točnim i preciznim.

(Slika 4.3.a.) SIFT ,također, detektira rubove i vrhove ,ali s malo manjom preciznošću u odnosu na FAST algoritam. (Slika 4.3.b.) SURF je detektirao i rubove i vrhove ,ali isto tako i čiste površine. (Slika 4.3.c.) Algoritam ORB je detektirao samo par točaka ,ali te točke su isključivo vrhovi. (Slika 4.3.d.) RISK je detektirao mali broj točaka te se tu nalaze vrhovi,rubovi te čiste površine. (Slika 4.3.e.)

5. ZAKLJUČAK

Računalni vid ,u smislu inženjerstva, pokušava automatizirati zadatke koje ljudski vizualni sustav ne može,metodama stjecanja,obrade,analize i razumijevanja digitalnih slika te ekstrakcija visokodimenzijskih podataka iz stvarnoga svijeta kako bi se dobile numeričke ili simboličke informacije. Početni ,a ujedno i ključni korak je ispravna detekcija značajki. Ovaj rad je obrađivao više vrsta algoritama ,ali se posebno fokusirao na FAST algoritam.

U eksperimentalnom dijelu ovoga rada nalaze se slike na kojima je testirano pet različitih algoritama:FAST, SIFT, SURF,ORB i BRISK. Pomoću njih su uočene prednosti i mane pojedinoga algoritma, tj. njegova točnost,brzina i preciznost. Poneki algoritmi nam nude brzinu, ali ne i preciznost dok su drugi vremenski sporiji, ali precizniji.

Na temelju provedenih eksperimenata zaključeno je kako u većini slučajeva FAST detektira veći broj ključnih točaka od ostalih testiranih algoritama. Također,t točke detektirane pomoću FASTA-a postavljene su u regijama slike s visokom varijabilnosti. Zbog toga je pretpostavka kako će rezultati obrade ključnih točaka bolje raditi korištenjem FAST-a nego ostalih algoritama,jer detektira veći broj jedinstveno određenih točaka.

6. LITERATURA

[1] Adaptive and Generic Corner Detection Based on the Accelerated Segment Test,

https://link.springer.com/chapter/10.1007%2F978-3-642-15552-9_14,

pristupljeno: 13.9.2018

[2] E. Rosten;T. Drummond;R.Porter;Faster and better: a machine learning approach to corner detection; 14.listopad 2008

file:///C:/Users/tanja/Desktop/zavrsni%20rad/fast-method.pdf, pristupljeno: 14.9.2018

[3] Feature detection (computer vision),

https://en.wikipedia.org/wiki/Feature_detection_(computer_vision),

pristupljeno 13.9.2018

[4] Analysis of Feature Detector and Descriptor Combinations with a Localization Experiment for Various Performance Metrics ;Mechanical Engineering Department; Istanbul Technical University,

[5] Edge detection, https://en.wikipedia.org/wiki/Edge_detection , pristupljeno: 14.9.2018.

[6] Corner detection, https://en.wikipedia.org/wiki/Corner_detection, pristupljeno: 15.9.2018.

[7] Blob detection, https://en.wikipedia.org/wiki/Blob_detection, pristupljeno: 14.9.2018.

[8] Ridge detection, https://en.wikipedia.org/wiki/Ridge_detection, pristupljeno: 13.9.2018.

[9] Scale-invariant feature transform,

https://en.wikipedia.org/wiki/Scale-invariant_feature_transform, pristupljeno: 13.9.2018

[10] file:///C:/Users/tanja/Desktop/zavrsni%20rad/sift.pdf

[11] E. Rosten; T. Drummond; Machine learning for high-speed corner detection; University of Cambridg;

[12] FAST Algorithm for Corner Detection
https://docs.opencv.org/3.0-
beta/doc/py_tutorials/py_feature2d/py_fast/py_fast.html, pristupljeno:
17.9.2018

[13] OpenCV, https://opencv.org/about.html, pristupljeno: 17.9.2018

[14] Harris Corner Detection, https://en.wikipedia.org/wiki/Harris_Corner_Detector,
pristupljeno 12.9.201

[15] Corner detection, https://en.wikipedia.org/wiki/Corner_detection, pristupljeno: 12.9.2018

7. SAŽETAK

Računalni vid i detekcije značajki za obradu slike obuhvaćaju metode za računanje apstrakcija slikovnih informacija i donošenje lokalnih odluka na svakoj točki slike, u tom trenutku postoji li neka vrsta slike određenog tipa ili ne. Dobivene značajke bit će podskupovi domene slike, često u obliku izoliranih točaka, kontinuiranih krivulja ili povezanih regija.

U ovom radu je predstavljen algoritam FAST, jedan od mnogih, za detekciju i opis značajki s nekim poboljšanim svojstvima u odnosu na druge algoritme. Paralelno su provedeni eksperimenti i na ostala četiri algoritma: SIFT, SURF, ORB, BRISK radi usporedbe rezultata.

Na osnovnu provedenih eksperimenata i dobivenih rezultata zaključeno je kako su značajke detektirane pomoću FAST-a kvalitetno slične ili malo bolje od ostalih metoda , ali je sam algoritam značajno brži od ostalih testiranih algoritama.

Ključne riječi: računalni vid, značajke, FAST

8. ABSTRACT

Computer vision and image detection features include methods for computing abstraction of image information and making local decisions at each image point, at which point there is a certain type of image of a particular type or not. The resulting features will be subdomains of the image domain, often in the form of isolated points, continuous curves, or related regions.

This paper presents a FAST algorithm, one of many, for detecting and describing features with some improved features compared to other algorithms. Parallel experiments were performed on the other four algorithms: SIFT, SURF, ORB, BRISK for comparison of results.

Based on the baseline experiments and the results obtained it was concluded that the features detected by FAST are very similar or slightly better than the other methods, but the algorithm itself is significantly faster than the other algorithms tested.

Keywords: Computer Vision, Features, FAST

9. ŽIVOTOPIS

Ana Pejković rođena je 26.11.1996 u Osijeku. Osnovnu je školu završila u Višnjevcu. Opću gimnaziju je završila u Osijeku 2015. godine. Nakon završetka srednje škole upisuje preddiplomski studij elektrotehnike na Fakultetu elektrotehnike,računarstva i informacijskih tehnologija u Osijeku,te se opredjeljuje za smjer Komunikacije i informatika.