SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 3576

Eksperimentalno vrednovanje strojno naučenog postupka za pronalaženje istaknutih značajki slike

Kristijan Biščanić

Zagreb, lipanj 2014.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 12. ožujka 2014.

ZAVRŠNI ZADATAK br. 3576

Pristupnik: **Kristijan Biščanić (0036464561)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Eksperimentalno vrednovanje strojno naučenog postupka za pronalaženje istaknutih značajki slike**

Opis zadatka:

Rad razmatra pronalaženje istaknutih značajki slike za potrebe analize kretanja u slikovnoj ravnini. U odabranom pristupu istaknute značajke se pronalaze postupkom čiji se parametri optimiraju strojnim učenjem.
U okviru rada, potrebno je iz literature proučiti različite postupke za detekciju istaknutih značajki slike. Posebnu pažnju obratiti na postupak FAST gdje se parametri postupka optimiraju na slikama iz aplikacijske domene. Razviti komponentu koja implementira postupak FAST izvođenjem zadanog apstraktnog sučelja. Vrednovati kvalitetu praćenja u realističnim eksperimentima, korištenjem postojećih komponenti i slijedova slika s poznatim optičkim tokom. Usporediti dobivene rezultate s postojećim komponentama koje implementiraju druge postupke detekcije istaknutih značajki. Prikazati i ocijeniti ostvarene rezultate.
Radu priložiti izvorni i izvršni kod razvijenih postupaka, ispitne slijedove i rezultate, uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Zadatak uručen pristupniku: 14. ožujka 2014.
Rok za predaju rada: 13. lipnja 2014.

Mentor:

Izv.prof.dr.sc. Siniša Šegvić

Predsjednik odbora za završni rad modula:

Prof.dr.sc. Siniša Srbljić

Djelovođa:

Doc.dr.sc. Tomislav Hrkać

Zahvala mentoru prof. dr. sc. Siniši Šegviću na sugestijama i pomoći pri izradi ovog rada.

SADRŽAJ

1. Uvod

Umjetna se inteligencija, kao jedno od danas najvećih područja istraživanja u računarskoj znanosti, bavi razvojem inteligentnih sustava. Područje umjetne inteligencije je vrlo kompleksno te je ona zbog toga podijeljena na nekoliko podpodručja zasnovanih na ključnim problemima, a među njima se nalazi i problem percepcije stvarnoga svijeta. Kako čovjek svoju percepciju zasniva na osjetilima, tako strojevi percepciju zasnivaju na različitim senzorima poput kamera, mikrofona, senzora pritiska, temperature i sl. Analiza podataka sa svakog je od navedenih senzora zahtjevna i relativno neovisna te se razvijaju grane umjetne inteligencije namijenjene upravo rješavanju takvih problema. Jedno od takvih područja je i područje računalnog vida, koje se bavi analizom, procesiranjem i razumijevanjem slika dobivenih kamerom, a jedan od ciljeva računalnog vida je oponašanje ljudskoga vida i njegovih sposobnosti računalnim sustavom.

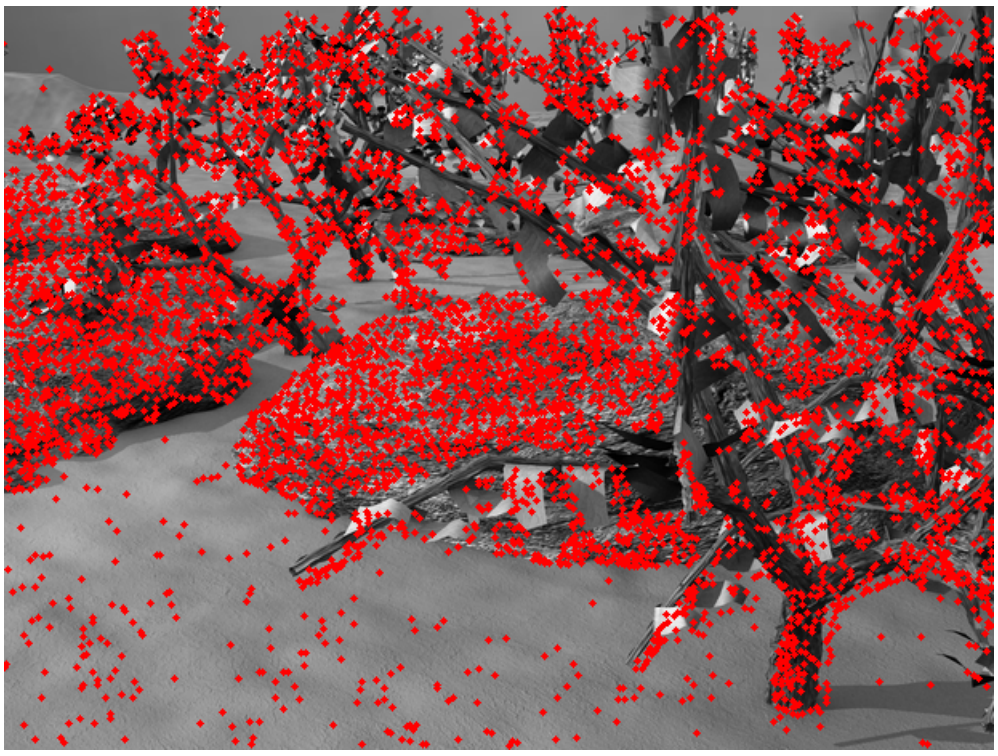
Svaka je digitalna slika sastavljena od velikog broja osnovnih slikovnih elemenata koje nazivamo pikselima. Mnogi od njih se ne razlikuju značajno u odnosu na okolinu te ne postoji uvijek potreba za analizom svakoga pojedinog piksela. Zbog toga su razvijeni postupci čija je namjena izbor dijelova bitnih za daljnju analizu od onih manje bitnih. Takve postupke nazivamo postupcima detekcije značajki, a takve dijelove slike nazivamo značajkama. Značajke, definirane kao zanimljivi dijelovi promatrane slike, mogu biti razne, od točaka, rubova, kutova pa sve do cijelih objekata. Značajke su osnovne čestice u brojnim primjenama računalnog vida kao što su poravnavanje panoramskih snimaka, prepoznavanja objekata i lica, praćenje pomaka te navigacije raznih robota, čak i automobila. Radi navedenoga je važno imati kvalitetne postupke detekcije značajki jer loša detekcija povlači i loše rezultate primjene algoritama koji koriste navedene značajke.

Ovaj se rad bavi eksperimentalnom evaluacijom postupaka detekcije značajki kako bi se utvrdilo koji postupci su prihvatljiviji za korištenje u brojnim drugim i kompleksnijim primjenama računalnoga vida. Konkretnije, promatraju se postupci detektiranja kutova kao točaka od interesa pa će, shodno tome, u nastavku ovoga rada pojam zna-

čajka podrazumijevati kutove, a ne rubove ili objekte.

Glavne karakteristike koje su očekivane od kvalitetnoga postupka detekcije značajki su ponovljivost i preciznost. Ponovljivošću se smatra sposobnost postupka da iste značajke pronalazi na različitim slikama iste scene, a preciznost detektora se odražava u preciznosti naprednijih postupaka praćenja značajki koji koriste navedeni postupak. Shodno tome, eksperimentalni je dio ovog rada podijeljen u 2 skupine eksperimenata na način da prva skupina eksperimenata evaluira ponovljivost detekcije svakog od postupaka, a druga skupina evaluira njihovu preciznost.

U sklopu ovoga rada posebna je pažnja posvećena postupku FAST [7] [8], strojno naučenom postupku detekcije kutova. Razvijena je komponenta koja preuzima funkcionalnost FAST postupka iz biblioteke OpenCV [2] i prilagođuje ju radu sa postojećim sustavom koji koristi praćenje detektiranih značajki za estimaciju pomaka. [4] Usporedno su evaluirane novorazvijena komponenta postupka FAST i postojeća komponenta koja implementira Harrisov postupak za detekciju kutova [3].



Slika 1: Primjer slike sa označenim detektiranim značajkama

2. Korišteni algoritmi i matematičke metode

Kako bismo mogli kvalitetno evaluirati različite postupke, nužno je barem osnvno znanje o principu rada svakog od njih. U ovom poglavlju ukratko će biti pojašnjeni principi rada dvaju evaluiranih postupaka, postupka FAST te Harrisovog detektora kutova. Također, slijedi kratak pregled linearnih geometrijskih transformacija koje se koriste u dijelovima rada koji slijede.

2.1. Postupci detekcije istaknutih značajki slike

Postupci detekcije istaknutih značajki slike koje razmatramo u ovome radu namijenjeni su detekciji kutova (engl. *corner*). Kutove definiramo kao točke u kojima se sjeku bridovi. Svaki od postupaka predpostavlja da radimo sa crno-bijelim slikama u 2 dimenzije.

2.1.1. Harrisov detektor kutova

Osnovna je ideja iza Harrisovog detektora kuteva[3] promatranje promjene intenziteta u ovisnosti o pomicanju prozora. Uzimamo mali dio slike (prozor) te promatramo kako se intenziteti mijenjaju ukoliko taj prozor pomičemo. 3 su moguća slučaja:

1. Bez obzira na smjer pomicanja prozora, nema promjene u intenzitetu ili je promjena vrlo mala - područje koje promatramo je jednoliko te ne sadrži istaknute značajke

2. Pomicanjem prozora u jednom smjeru promjena intenziteta je značajna, a pomicanjem u drugom smjeru promjena je mala ili je nema - područje koje promatramo sadrži rub koji se proteže u smjerovima u kojima nema promjene intenziteta

3. Pomicanjem prozora u bilo kojem smjeru promjena je značajna - područje koje promatramo sadrži kut

Definirano funkciju $I(x, y)$ kao funkciju intenziteta slike u točki (x, y) te funkciju $w(x, y)$ kao funkciju prozora - funkcija vraća 1 ukoliko točka (x, y) pripada prozoru, a u svim ostalim slučajevima 0. Ukoliko se prozor pomiče za $[u, v]$, promjenu intenziteta računamo preko sume kvadrata razlika te ju možemo pisati kao:

$$E(u, v) = \sum_{x,y} w(x, y)[I(x + u, y + v) - I(x, y)]^2 \tag{1}$$

Ukoliko su I_x i I_y parcijalne derivacije od I možemo aproksimirati:

$$I(u + x, y + v) \approx I(x, y) + I_x(x, y)u + I_y(x, y)v \tag{2}$$

Uvrštavanjem u prethodnu formulu dobivamo:

$$E(u, v) = \sum_{x,y} w(x, y)[I_x(x, y)u + I_y(x, y)v)]^2 \tag{3}$$

Što se može zapisati u matričnom obliku kao:

$$E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}, \quad M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_x \\ I_x I_x & I_y^2 \end{bmatrix} \tag{4}$$

Matrica M je Harrisova matrica iz koje izračunavamo svojstvene vektore λ_1 i λ_2 pomoću kojih možemo zaključiti upravo o kakvoj se značajci radi i to:

1. Ako su svojstveni vektori vrlo mali - nema značajki
2. Ako je jedan od svojstvenih vektora velik, a drugi mali - rub
3. Ako su oba svojstvena vektora veliki - kut

Kako je računanje svojstvenih vektora zahtjevna operacija, uvodi se mjera $R = \det(M) - k(\mathrm{trace}(M))$, gdje je k empirijski izračunata konstanta sa uobičajenim vrijednostima $0.04 - 0.06$. Sa tako izračunatim R možemo zaključiti:

1. Ako je $|R|$ mala - nema značajki
2. $R < 0$ - rub
3. $R > 0$ - kut

2.1.2. Postupak FAST

Postupak FAST za svaki pixel p promatra 16 piksela koji ga okružuju (Bresenhamov krug polumjera 3). Ukoliko postoji n od navedenih 16 piksela čiji je intenzitet za vrijednost praga t veći od intenziteta piksela p ili postoji n piksela takvih da je intenzitet svakog od njih manji za vrijednost praga t od intenziteta piksela p, postupak će piksel p proglasiti točkom interesa.

Ukoliko je $n \geq 12$ može se uvesti dodatna optimizacija. Prije promatranja svih 16 piksela, prvo se promatraju 4 određena piksela. Činjenica da barem 12 od 16 piksela mora biti ili svijetlije ili tamnije od p za iznos praga doprinosi brzom odbacivanju piksela koji ne zadovoljavaju taj uvijet. Naime, od 4 odabrana piksela, barem 3 moraju biti ili svi svjetliji ili svi tamniji od piksela p za iznos praga. Prvo se testiraju pikseli 1 i 9, odnoso pikseli koji imaju x koordinatu jednaku kao piksel p. Ako niti jedan od piksela nije niti svjetliji niti tamniji od piksela p za iznos praga, testiranje se zaustavlja i prelazi se na slijedeći piksel. Ukoliko barem jedan je, procedura se ponavlja za piksele 5 i 13, odnosno piksele koji imaju y koordinatu jednaku kao i testirani piksel. Ukoliko 3 od 4 testirana piksela ne zadovoljavaju uvjet, piksel se odbacuje. U protivnome, procedura se izvršava za ostatak piksela te se na temelju svih rezultata dolazi do odluke je li piksel p značajka ili nije.

Dodatno ubrzanje postupka omogućeno je uporabom strojnog učenja kako bi se postupak naučilo na koji način izabirati prva 4 piksela iz skupa od 16. Također, korištenjem potiskivanja nemaksimalnih piksela omogućava se kvalitetniji izbor značajki te se spriječava odabir susjednih točaka kao istaknutih točaka slike.

2.2. Linearne geometrijske transformacije slike u homogenim koordinatama

U računalnoj memoriji slika može biti zapisana na više različitih načina, ali svaki oblik zapisa omogućuje dohvat slikovnih elemenata. Koordinatni sustav slike dogovorno postavljamo tako da gornji lijevi slikovni element ima koordinate $(0, 0)$ te ga smatramo ishodištem. S obzirom da su slike 2D objekti, slikovne elemente možemo prikazati u homogenome prostoru sa 3 koordinate. Svaki će slikovni element sa koordinatama (x, y) u homogenom prostoru biti prikazan kao stupčani vektor: $T = (x, y, 1)$. Razmatranje ćemo ograničiti na linearne geometrijske transformacije[9] nad slikovnim elementima prikazanim homogenim koordinatama. Takve transformacije se mogu opisati matričnim množenjem homogenih koordinata slikovnog elementa matricom transfor-

macije veličine 3×3. Transformacija slike se svodi na provođenje transformacije nad svakim slikovnim elementom koji ta slika sadrži. Postupak svih transformacija je isti, a jedina je razlika u obliku matrice transformacije ($\mathbf{\Omega}$). Koordinate transformirane točke dobiju se matričnim umnoškom matrice transformacija sa početnim koordinatama točke:

$$T' = \mathbf{\Omega} \cdot T \tag{5}$$

Na ovakav način možemo dobiti nekoliko klasa transformacija:

- Izometrične transformacije - čuvaju kutove i udaljenosti: translacija, rotacija
- Sličnosti - čuvaju kutove i omjere duljina: skaliranje
- Afine transformacije - čuvaju paralelnost
- Projekcijske transformacije - čuvaju kolinearnost

2.2.1. Translacija

Ukoliko želimo točku T translatirati za Δ_x po x osi te za Δ_y po y osi, koristit ćemo sljedeću jednadžbu:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta_x \\ 0 & 1 & \Delta_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{6}$$

2.2.2. Rotacija oko ishodišta koordinatnog sustava slike

Ukoliko želimo točku T rotirati za kut α oko ishodišta u smjeru suprotnome od kazaljke na satu, koristit ćemo sljedeću jednadžbu:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{7}$$

2.2.3. Skaliranje

Skaliranje točke faktorima S_x za koordinatu x te S_y za koordinatu y odvija se sljedećom jednadžbom:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{8}$$

Jednadžbom (8) je predočeno nejednoliko skaliranje pri kojemu se svaka os skalira vlastitim faktorom. Jednoliko je skaliranje samo specijalni slučaj pri kojemu je zadovoljen uvjet $S = S_x = S_y$ te za takav slučaj jednadžbu (8) možemo zapisati i kao:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S & 0 & 0 \\ 0 & S & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{9}$$

2.2.4. Rotacija oko središta slike

Primijetimo da nam već gore navedene osnovne geometrijske transformacije nisu dovoljne za izvođenje svih slika potrebnih za eksperimente korištene u ovome radu. Međutim, kako vidimo da sve transformacije imaju oblik kao i jednadžba (5), lagano dolazimo do zaključka kako matrica transformacija ne mora vršiti samo jednu transformaciju, nego se matričnim množenjem osnovnih geometrijskih transformacija mogu dobiti i kompleksnije matrice transformacija. Jednadžbu (5) možemo zapisati i kao:

$$T' = \mathbf{\Omega}_n \cdots \mathbf{\Omega}_2 \cdot \mathbf{\Omega}_1 \cdot T = \mathbf{\Omega}' \cdot T \tag{10}$$

U prethodnoj jednadžbi, matrice $\mathbf{\Omega}_1, \mathbf{\Omega}_2, \ldots, \mathbf{\Omega}_n$ predstavljaju matrice osnovnih geometrijskih transformacija, a matrica $\mathbf{\Omega}'$ je rezultantna matrica afine transformacije nastala matričnim umnoškom spomenutih matrica. S obzirom da matrično množenje nije komutativno, redoslijed operacija je bitan te se u ovakvome umnošku operacije pišu suprotnim redoslijedom od onoga kojim se primjenjuju.

Sada primijetimo da nam jednadžba rotacije (7) nije primjenjiva na rotaciju slike pošto sliku želimo rotirati oko njezinog središta, a ne oko ishodišta koordinatnog sustava što je u slučaju slike predstavljeno gornjim lijevim slikovnim elementom. Koordinate središta slike računaju se na način:

$$x_c = \frac{\text{širina slike}}{2}, \quad y_c = \frac{\text{visina slike}}{2} \tag{11}$$

Rotaciju oko središta slike dobijemo ulančavanjem translacije i rotacije, i to na sljedeći način:

1. Translatiramo točku za $\Delta_x = -x_c$, $\Delta_y = -y_c$ koristeći formulu (6)

2. Rotiramo točku oko ishodišta za željeni kut α koristeći formulu (7)

3. Translatiramo točku za $\Delta_x = x_c$, $\Delta_y = y_c$ koristeći formulu (6)

Prethodni koraci mogu se zapisati i kao matrični umnožak:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_c \\ 0 & 1 & y_c \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_c \\ 0 & 1 & -y_c \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{12}$$

Nakon množenja matrica dolazimo do konačne formule za rotaciju oko središnje točke:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha & (1-\cos\alpha)\cdot x_c + \sin\alpha \cdot y_c \\ \sin\alpha & \cos\alpha & (1-\cos\alpha)\cdot y_c - \sin\alpha \cdot x_c \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{13}$$

2.2.5. Afina transformacija

Afina transformacija u najopćenitijem slučaju je transformacija koja se sastoji od translacije, rotacije, skaliranja te smika. Svaka afina transformacija se može prikazati jednadžbom:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{14}$$

2.2.6. Projekcijska transformacija

Projekcijska transformacija ili homografija je najopćenitija vrsta linearnih geometrijskih transformacija. Kod projeckijske transformacije matrica Ω može biti bilo koja matrica dimenzija 3×3, bez ograničenja na neke elemente kao što je bio slučaj u prijašnjim transformacijama. Međutim, treći član homogenog vektora T' nakon primjene proizvoljne projekcijske transformacije ne mora nužno biti 1 te je stoga potrebno koordinate vratiti u nehomogeni prostor:

$$x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}}, \quad y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}} \tag{15}$$

3. Ispitni skupovi izvornih slika

Svi su ispitni primjeri izvođeni na bazi slika namijenjenoj evaluaciji algoritama računanja optičkog toka koju je objavio Middlebury College [1] [1] te za koju su dostupne točne vrijednosti (engl. *ground-truth*) optičkoga toka za svaku točku slike. Baza slika je napravljena na način da se nepomična scena fotografirala kamerom koja je radila vrlo male pomake (do 5 piksela). Točne vrijednosti pomaka dobivene su tako da je scena premazana slojem UV boje te je kamera u svakoj pozicija radila dvije fotografije: jednu pod normalnim osvjetljenjem koja je postala slika u bazi te drugu pod UV svjetlom za potrebe određivanja točnih vrijednosti pomaka visoke preciznosti. U svim se eksperimentima koriste već dostupne crno-bijele inačice slika.

3.1. Evaluacija ponovljivosti detekcije

Eksperimentalna je evaluacija ponovljivosti detekcije značajki izvođena na jednoj slici (Slika 2a) iz navedene baze nad kojom su provođene transformacije opisane u prethodnom poglavlju i to na sljedeći način (Slika 2):

Rotacija

Početna je slika rotirana koristeći jednadžbu (13) za kut α koji poprima vrijednosti iz intervala $[-90°, +90°]$ isključujući slučaj $\alpha = 0°$. Inkrement iznosi $10°$.

Jednoliko skaliranje

Početna je slika skalirana koristeći jednadžbu (9) faktorom S koji poprima vrijednosti iz intervala $[0.8, 1.3]$, isključujući slučaj $S = 1$. Inkrement iznosi 0.05.

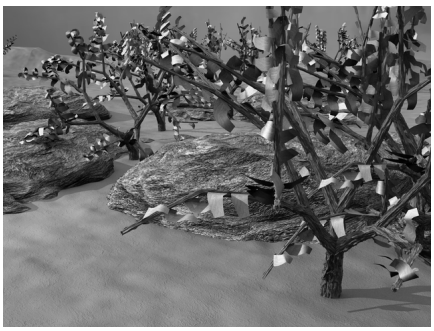
Nejednoliko skaliranje

Početna je slika skalirana koristeći jednadžbu (8) faktorima S_x i S_y. S_x poprima vrijednosti iz intervala $[0.8, 1]$, a S_y poprima vrijednosti iz intervala $[0.5, 1.5]$. Inkrementi za oba faktora iznose 0.1.

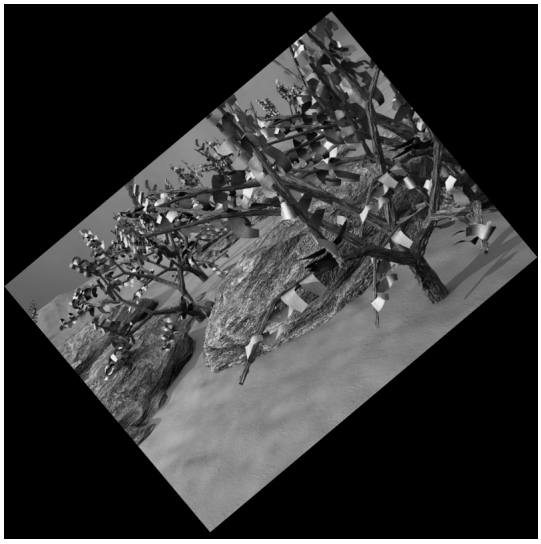
[1] http://vision.middlebury.edu/flow/

Afina transformacija

Na početnu je sliku prvo primijenjena transformacija nejednolikog skaliranja koristeći jednadžbu (8), a zatim je rotirana oko vlastitog središta jednadžbom (13). Za provođenje navedenih transformacija korišteni su parametri S_x, S_y i α. S_x poprima vrijednosti iz intervala $[0.8, 1]$, S_y poprima vrijednosti iz intervala $[0.5, 1.5]$, a α poprima vrijednosti iz intervala $[-90°, +90°]$. Inkrementi za faktore S_x i S_y iznose 0.1, a za faktor α $10°$.



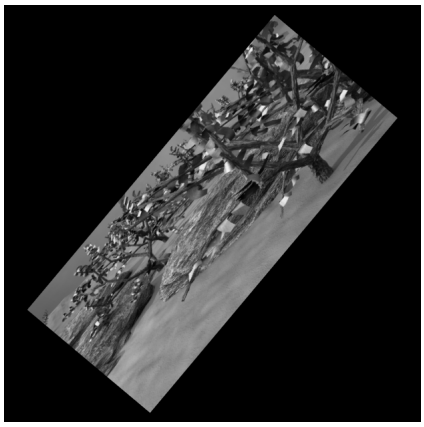
(a) Početna slika



(b) Rotacija

(c) Jednoliko skaliranje

(d) Nejednoliko skaliranje



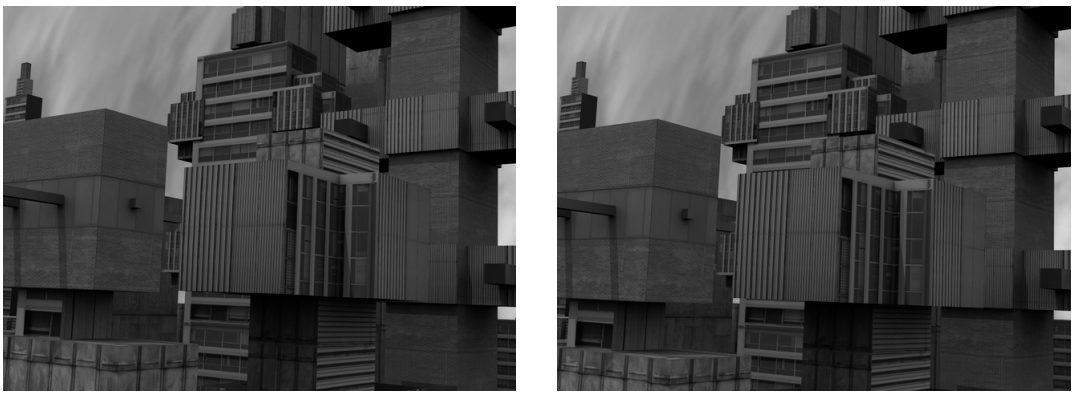
(e) Afina transformacija

Slika 2: Testna slika i primjeri njezinih transformacija

3.2. Evaluacija preciznosti detekcije

Eksperimentalna je evaluacija preciznosti detekcije izvođena na nekoliko parova slika iz navedene baze za koje postoje točne vrijednosti pomaka. Način evaluacije pojašnjen je u poglavlju 5.2. Parovi slika su fotografije iste scene sa vrlo malim pomakom kamere između dvije uzastopne slike te je tako na slikama nastao pomak od nekoliko piksela. Nekoliko primjera takvih parova vidljivo je na slikama: 3, 4 i 5.

Slika 3: Primjer para slika nad kojima je evaluirana preciznost



Slika 4: Primjer para slika nad kojima je evaluirana preciznost

Slika 5: Primjer para slika nad kojima je evaluirana preciznost

4. Programska izvedba i vanjske biblioteke

U sljedećim će poglavljima biti pojašnjena programska izvedba koda implementiranog u sklopu ovoga rada kao i biblioteke koje su korištene pri implementaciji. Sav je programski kod pisan u programskom jeziku C++ zbog ostvarivanja veće brzine te zbog kompatibilnosti sa postojećim modulima koji su korišteni. Cjelokupni se izvorni kod nalazi na optičkom mediju priloženom radu te je prilagođen za kofiguriranje pomoću sustava CMake[1].

4.1. OpenCV

Biblioteka OpenCV[2] (engl. *Open Source Computer Vision Library*) je biblioteka otvorenoga koda koja sadrži preko 2500 optimiziranih algoritama iz područja računalnoga vida i strojnoga učenja[2]. OpenCV je distribuiran pod licencom BSD[3] što omogućuje njegovu slobodnu uporabu u akademskim i komercijalnim programima. Podržana je na svim značajnijim operativnim sustavima: Windows, Linux, Android i Mac OS te postoje inačice biblioteke namijenjene za korištenje u nekoliko programskih jezika: C, C++, Python, Java te Matlab. S obzirom na veličinu, biblioteka je podijeljena na module koji se u razvijeni program uključuju po potrebi. Neki od važnijih modula biblioteke OpenCV su:

- *core* - jezgra biblioteke OpenCV i njezine osnovne funkcionalnosti, strukture i tipovi podataka
- *imgproc* - postupci za obradu slika
- *highui* - podrška za kreiranje korisničkog sučelja, prikazivanje slika na ekranu te učitavanje i zapisivanje slika

[1] http://www.cmake.org/
[2] http://opencv.org/
[3] http://opensource.org/licenses/BSD-3-Clause

– *video* - analiza video zapisa

– *ml* - postupci strojnog učenja

– *gpu* - podrška izvođenju postupaka na grafičkom procesoru

– ...

Programski kod ovog rada koristio je inačicu biblioteke namijenjenu za jezik C++ (inačica 2.4.8) na operacijskom sustavu Windows. Moduli koji su korišteni su: *core*, *highui* te *imgproc*.

4.2. Komponente za detekciju i praćenje značajki

Iz rada [4] su preuzeta sučelja i postojeće komponente za detekciju i praćenje značajki na slikama te je dodana implementacija još jednoga postupka za detekciju značajki koji implementira zadano sučelje te ga se punopravno može koristiti u sustavu [4]. Preuzete su komponente podijeljene u nekoliko modula:

– *core* - pomoćne funkcije i tipovi podataka korišteni u svim ostalim modulima

– *detector* - sučelja i implementacije nekolicine postupaka za detekciju značajki

– *tracker_base* - strukture koje koriste svi postupci za praćenje

– *tracker_mono* - sučelja i implementacije nekolicine postupaka za praćenje značajki na sljedovima slika snimanim jednom kamerom

Cilj je bio implementirati dodatan postupak za detekciju značajki koji nasljeđuje sljedeće sučelje:

```
1 class FeatureDetectorBase
2 {
3 public:
4     virtual ~FeatureDetectorBase() {}
5     virtual void detect(const core::Image& img, std::
      vector<core::Point>& features) = 0;
6 };
```

Sučelje od detektora zahtjeva da implementira funkciju `detect` koja kao argumente prima sliku nad kojom se vrši detekcija značajki te strukturu u koju detektor treba spremiti sve pronađene istaknute točke slike. Sljedeći je korak izvođenje novoga razreda iz navedenoga sučelja:

```
1 class FeatureDetectorFastCV : public FeatureDetectorBase
2 {
3 public:
4   FeatureDetectorFastCV(int threshold = 10, bool
     nonmaxSuppresion = true, cv::Mat mask = cv::Mat(), int
      margin = 0, int hbins = 10, int vbins = 10, int fpb =
      20);
5   virtual void detect(const core::Image& img, std::vector
     <core::Point>& features);
6
7 private:
8    cv::Mat cvimg_, mask_;
9    int margin_;
10   int h_bins_, v_bins_, fpb_;
11   std::shared_ptr<cv::FeatureDetector> detector_;
12   std::vector<cv::KeyPoint> keypoints_;
13 };
```

Argumenti `threshold` i `nonmaxSuppression` su parametri postupka FAST. Argument `mask` predstavlja masku kojom se može ograničiti područje slike na kojoj će se tražiti značajke, argument `margin` predstavlja marginu uz rubove slike unutar koje se pronađene značajke zanemaruju, a argumenti `hbins`, `vbins` i `fpb` služe za ograničavanje broja pronađenih značajki na dijelovima slike. `hbins` i `vbins` određuju broj horizontalnih odnosno vertikalnih pretinaca na slici, a `fpb` maksimalan broj značajki u svakom pretincu. Podatkovni član `detector_` sadrži referencu na implementaciju detektora iz OpenCV biblioteke, a podatkovni član `keypoints_` služi za prihvaćanje pronađenih značajki nakon poziva detekcije. Implementacija ovog razreda ostvarena je na sljedeći način:

```
1   FeatureDetectorFastCV::FeatureDetectorFastCV(int
     threshold, bool nonmaxSuppresion, Mat mask, int margin
     , int hbins, int vbins, int fpb)
2   {
3     detector_ = std::shared_ptr<FeatureDetector>(new
     FastFeatureDetector(threshold, nonmaxSuppresion));
4     margin_ = margin; h_bins_ = hbins; v_bins_ = vbins;
     fpb_ = fpb; mask_ = mask;
```

```
5   }
```

Primijetimo da je razred `FeatureDetectorFastCV` samo omotač koji implementaciju postupka FAST iz biblioteke OpenCV prilagođava za rad sa ostalim komponentama kako bismo nad različitim postupcima mogli provoditi evaluaciju na potpuno jednak način. Metoda `detect` također samu detekciju točaka od interesa prosljeđuje referenciranom detektoru.

4.3. Program za evaluaciju ponovljivosti detekcije

Ovaj je program namijenjen provođenju eksperimenata čiji je cilj evaluacija ponovljivosti detekcije značajki. Za svaki je eksperiment implementirana posebna funkcija čiji izvorni kod slijedi. Ideja je iza svake funkcije ista: u argumentima se prima naziv i lokacija slike koja će za taj eksperiment biti početna. Nad početnom se slikom u svakoj iteraciji primjenjuju linearne geometrijske transformacije čiji parametri ovise o broju trenutne iteracije. Kao rezultat izvršavanja se stvara datoteka sa brojem detektiranih istaknutih točaka slike i trajanjem detekcije za svaku od slika stvorenih na opisan način. Datoteka s rezultatima se u sljedećem poglavlju analizira i komentira.

4.3.1. Evaluacija ponovljivosti na rotiranim slikama

```
1  void evalRotation(string source_folder, vector<string>&
       imglist, FeatureDetectorBase& detector){
2    Mat origimg = imread(source_folder + imglist[0],
       CV_LOAD_IMAGE_GRAYSCALE);
3
4    int diagonal = (int)sqrt(origimg.cols*origimg.cols +
       origimg.rows*origimg.rows);
5    int offsetX = (diagonal - origimg.cols) / 2;
6    int offsetY = (diagonal - origimg.rows) / 2;
7    Mat cvimg(diagonal, diagonal, origimg.type());
8
9    ofstream outfile("rotation.txt");
10
11   for (double i = -90; i <= 90; i += 10){
12     cvimg = Scalar(0, 0, 0);
```

```
13      origimg.copyTo(cvimg.rowRange(offsetY, offsetY +
      origimg.rows).colRange(offsetX, offsetX + origimg.cols
      ));
14      rotate(cvimg, i, cvimg);
15
16      core::Image img;
17      HelperOpencv::MatToImage(cvimg, img);
18
19      std::vector<core::Point> feats;
20      clock_t begin = clock();
21      detector.detect(img, feats);
22      clock_t end = clock();
23
24      outfile << i << ", " << feats.size() << ", " <<
      double(end - begin) / CLOCKS_PER_SEC << endl;
25    }
26    outfile.close();
27 }
```

4.3.2. Evaluacija ponovljivosti na jednoliko skaliranim slikama

```
1 void evalScaling(string source_folder, vector<string>&
     imglist, FeatureDetectorBase& detector){
2   Mat origimg = imread(source_folder + imglist[0],
     CV_LOAD_IMAGE_GRAYSCALE);
3
4   Mat cvimg(origimg.rows * 1.5, origimg.cols * 1.5,
     origimg.type());
5
6   ofstream outfile("uni-scaling.txt");
7
8   for (double i = 0.8; i < 1.34; i += .05){
9     cvimg = Scalar(0, 0, 0);
10    resize(origimg, cvimg, cv::Size(0, 0), i, i);
11
12    core::Image img;
```

```
13      HelperOpencv::MatToImage(cvimg, img);
14
15      std::vector<core::Point> feats;
16      clock_t begin = clock();
17      detector.detect(img, feats);
18      clock_t end = clock();
19
20      outfile << i << ", " << feats.size() << ", " <<
      double(end - begin) / CLOCKS_PER_SEC << endl;
21    }
22    outfile.close();
23 }
```

4.3.3. Evaluacija ponovljivosti na nejednoliko skaliranim slikama

```
1 void evalNUScaling(string source_folder, vector<string>&
     imglist, FeatureDetectorBase& detector){
2   Mat origimg = imread(source_folder + imglist[0],
     CV_LOAD_IMAGE_GRAYSCALE);
3
4   Mat cvimg(origimg.rows * 1.5, origimg.cols * 1.5,
     origimg.type());
5
6   ofstream outfile("nu-scaling.txt");
7   for (double i = 0.8; i < 1.09; i += .1){
8     for (double j = 0.5; j < 1.59; j += .1){
9       cvimg = Scalar(0, 0, 0);
10      resize(origimg, cvimg, cv::Size(0, 0), i, j);
11
12      core::Image img;
13      HelperOpencv::MatToImage(cvimg, img);
14
15      std::vector<core::Point> feats;
16      clock_t begin = clock();
17      detector.detect(img, feats);
18      clock_t end = clock();
```

```
19
20        outfile << "Sx=" << i << " Sy=" <<j << ", " <<
      feats.size() << ", " << double(end - begin) /
      CLOCKS_PER_SEC << endl;
21      }
22    }
23    outfile.close();
24 }
```

4.3.4. Evaluacija ponovljivosti na afino transformiranim slikama

```
 1 void evalAffine(string source_folder, vector<string>&
     imglist, FeatureDetectorBase& detector){
 2   Mat origimg = imread(source_folder + imglist[0],
     CV_LOAD_IMAGE_GRAYSCALE);
 3
 4   Mat cvimg(origimg.rows * 1.5, origimg.cols * 1.5,
     origimg.type());
 5
 6   ofstream outfile("affine.txt");
 7
 8   {
 9     core::Image img;
10     HelperOpencv::MatToImage(origimg, img);
11
12     std::vector<core::Point> feats;
13     clock_t begin = clock();
14     detector.detect(img, feats);
15
16     outfile << "POC:" << feats.size() << endl;
17   }
18
19   for (double i = 0.8; i < 1.09; i += .1){
20     for (double j = 0.5; j < 1.09; j += .1){
21       cvimg = Scalar(0, 0, 0);
22       resize(origimg, cvimg, cv::Size(0, 0), i, j);
```

```
23
24        int diagonal = (int)sqrt(cvimg.cols*cvimg.cols +
      cvimg.rows*cvimg.rows);
25        int offsetX = (diagonal - cvimg.cols) / 2;
26        int offsetY = (diagonal - cvimg.rows) / 2;
27        Mat scimg(diagonal, diagonal, cvimg.type());
28        for (double alpha = -90; alpha < 99; alpha += 10){
29
30          scimg = Scalar(0, 0, 0);
31          cvimg.copyTo(scimg.rowRange(offsetY, offsetY +
      cvimg.rows).colRange(offsetX, offsetX + cvimg.cols));
32          rotate(scimg, alpha, scimg);
33
34          core::Image img;
35          HelperOpencv::MatToImage(scimg, img);
36
37          std::vector<core::Point> feats;
38          clock_t begin = clock();
39          detector.detect(img, feats);
40          clock_t end = clock();
41
42          outfile << feats.size() << ", " << double(end -
      begin) / CLOCKS_PER_SEC << endl;
43        }
44      }
45    }
46    outfile.close();
47 }
```

4.4. Program za evaluaciju preciznosti detekcije

Drugi program koji je razvijen u sklopu ovog rada je program čija je namjena evaluacija preciznosti praćenja značajki na paru slika u ovisnosti o izboru postupka za detekciju. Program učitava slijed slika, pomoću izabranog postupka za detekciju i postojeće komponente za praćenje značajki računa pomake svake od značajki te evaluira podatke dobivene tim putem u odnosu na vrijednosti točnih pomaka.

5. Eksperimentalni rezultati

Eksperimentalna je evaluacija najbolji pokazatelj primjerenosti korištenja pojedinih postupaka u praksi. U ovome će poglavlju biti opisani eksperimenti provedeni nad dva postupka za detekciju točaka interesa (kuteva) na slikama. Isti su eksperimenti usporedno provođeni nad postupkom FAST [7] te Harrisovom postupku za detekciju kuteva [3]. Eksperimentalna je evaluacija podijeljena na dva dijela opisana u nastavku.

5.1. Eksperimentalna evaluacija ponovljivosti detekcije

Prva je skupina eksperimenata namijenjena evaluaciji ponovljivosti detekcije značajki, odnosno ispitivanju konzistentnosti postupka detekcije. Konzistentan postupak trebao bi detektirati jednak broj značajki bez obzira kombinaciju geometrijskih transformacija primijenjenih na početnu sliku. U seriji eksperimenata bit će primjenjivane različite transformacije na početnu sliku te će se pratiti broj detektiranih točaka interesa na svakoj od transformiranoj slika. Kriterij usporedbe preuzet je iz [5] te glasi:

$$CCN = K^{-|N_t - N_0|} \times 100\%, \quad K = 1.001 \tag{16}$$

CCN predstavlja konzistenciju broja kutova (engl. *consistency of corner numbers*), N_0 je broj detektiranih kutova na početnoj slici, a N_t na slici nakon provođenja transformacije. Jedina je preinaka u odnosu na izvorni kriterij konstanta K koja je zbog osjetno većega broja točaka interesa na ispitnome skupu ovoga rada smanjena sa 1.1 na 1.001 kako konzistentnost ne bi prebrzo pala blizu nule. Primijetimo da su koristeći jednadžbu (16) vrijednosti CCN moguće samo iz intervala $[0\%, 100\%]$ što pogoduje laganoj usporedbi. Svaka razlika između brojeva N_t i N_0 rezultira padanjem vrijednosti CCN tako da su ovim kriterijem pokriveni i slučajevi kada su detektirane nepostojeće značajke (engl. *false positive error*) i slučajevi kada je detektiran manji broj značajki nego što je trebao biti. Ispitni su skupovi za ovaj skup eksperimenata napravljeni na način kako je opisano u poglavlju 3.1. Također, uz svaki je eksperiment mjerena i brzina izvođenja optimiziranog strojnog koda za svaki postupak. Ti će

rezultati također biti analizirani imajući na umu da je brzina izvođenja u određenim slučajevima presudna za odabir postupka.

Treba primijetiti da se na ovakav način evaluira samo broj detektiranih značajki, a ne postoji garancija da su na početnoj i transformiranoj slici detektirane upravo iste značajke. Naknadno će biti razvijena unaprijeđena verzija ovoga eksperimenta koja će u obzir uzimati i udio značajki početne slike koje su detektirane na transformiranoj slici i koje uistinu možemo smatrati ponovljenim značajkama.

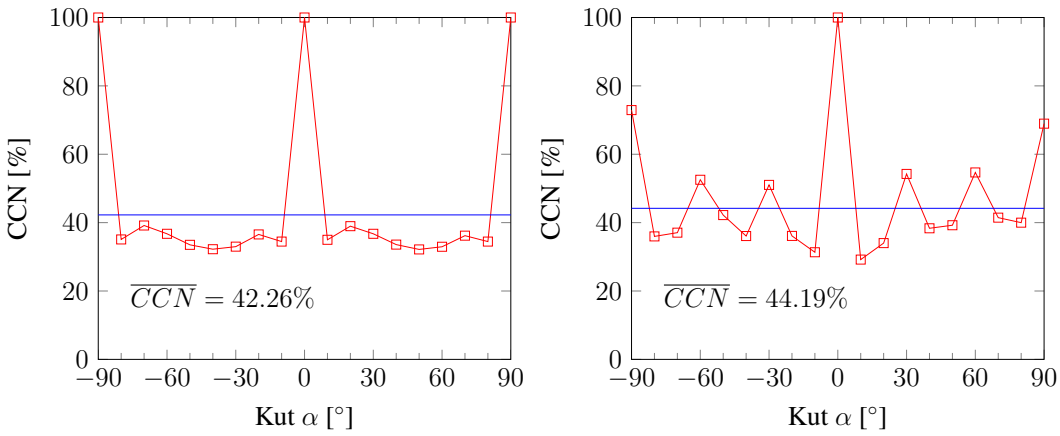
Napomena: Vrijednost CCN za početnu sliku uvijek je 100%, ali ona nije uzimana u obzir prilikom računanja prosječne vrijednosti.

5.1.1. Eksperiment 1

Ispitni skup za ovaj eksperiment nastaje tako da se na početnu sliku primjenjuje rotacija kako je opisano u poglavlju 3.1. Parametar rotacije α poprima vrijednosti iz intervala $[-90°, +90°]$ isključujući slučaj $\alpha = 0°$. Inkrement iznosi $10°$.

(a) $\alpha = 70°$ **(b)** $\alpha = 30°$ **(c)** Početna slika **(d)** $\alpha = -30°$ **(e)** $\alpha = -70°$

Slika 6: Primjer slika ispitnog skupa eksperimenta 1

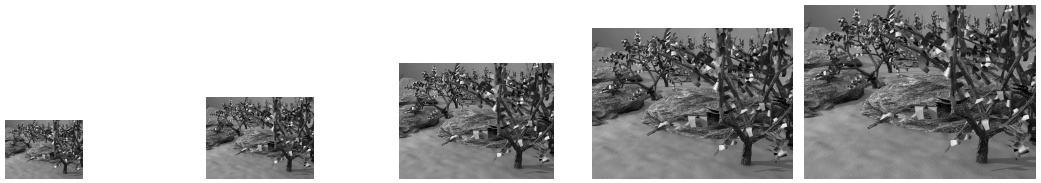


(a) Harrisov detektor kuteva **(b)** FAST

Slika 7: Rezultati izvođenja eksperimenta 1

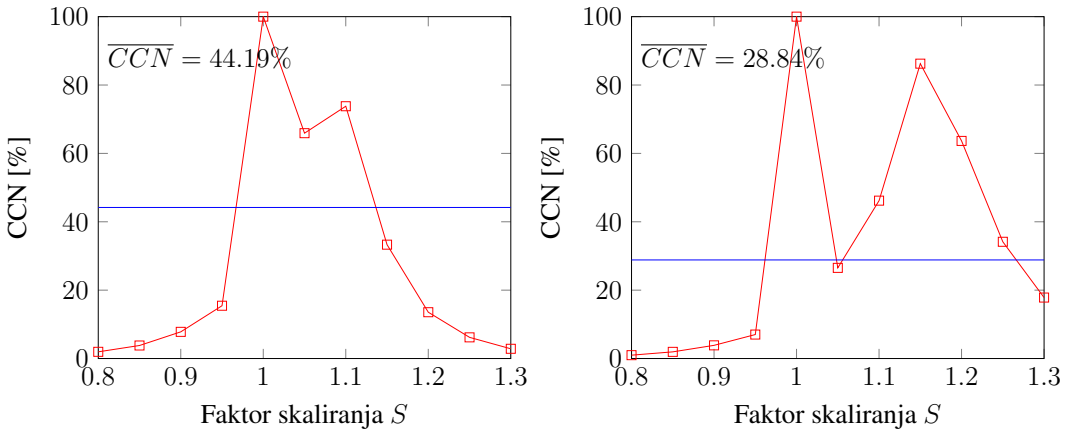
5.1.2. Eksperiment 2

Ispitni skup za ovaj eksperiment nastaje tako da se na početnu sliku primjenjuje jednoliko skaliranje kako je opisano u poglavlju 3.1. Parametar skaliranja S poprima vrijednosti iz intervala $[0.8, 1.3]$, isključujući slučaj $S = 1$. Inkrement iznosi 0.05.



(a) $S = 0.5$ **(b)** $S = 0.7$ **(c)** Početna slika **(d)** $S = 1.3$ **(e)** $S = 1.5$

Slika 8: Primjer slika ispitnog skupa eksperimenta 2



(a) Harrisov detektor kuteva **(b)** FAST

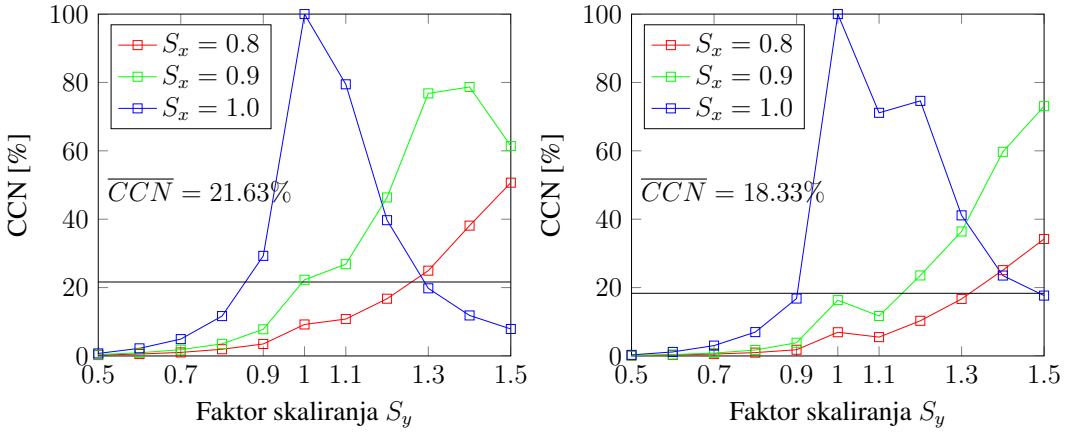
Slika 9: Rezultati izvođenja eksperimenta 2

5.1.3. Eksperiment 3

Ispitni skup za ovaj eksperiment nastaje tako da se na početnu sliku primjenjuje nejednoliko skaliranje kako je opisano u poglavlju 3.1. Parametar S_x poprima vrijednosti iz intervala $[0.8, 1]$, a S_y poprima vrijednosti iz intervala $[0.5, 1.5]$. Inkrementi za oba faktora iznose 0.1.

(a) $S_x = 0.8$, $S_y = 0.5$ **(b)** $S_x = 0.8$, $S_y = 1.5$ **(c)** Početna slika **(d)** $S_x = 1.0$, $S_y = 0.5$ **(e)** $S_x = 1.0$, $S_y = 1.5$

Slika 10: Primjer slika ispitnog skupa eksperimenta 3

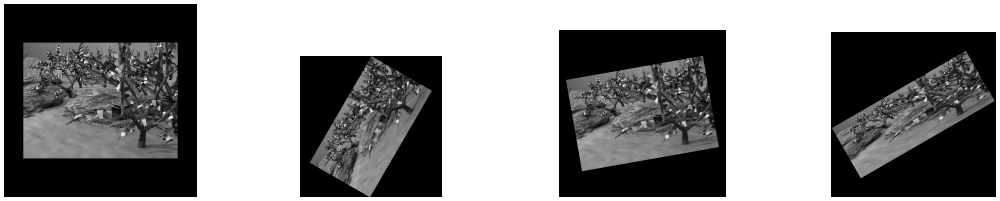


(a) Harrisov detektor kuteva **(b)** FAST

Slika 11: Rezultati izvođenja eksperimenta 3

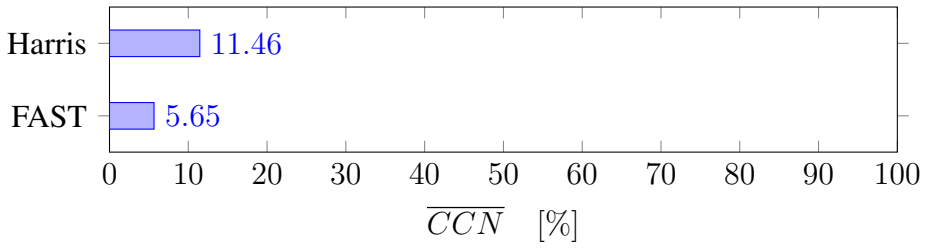
5.1.4. Eksperiment 4

Ispitni skup za ovaj eksperiment nastaje tako da se na početnu sliku primjenjuje nejednoliko skaliranje, a zatim rotacija oko vlastitog središta kako je opisano u poglavlju 3.1. Za provođenje navedenih transformacija korišteni su parametri S_x, S_y i α. S_x poprima vrijednosti iz intervala $[0.8, 1]$, S_y poprima vrijednosti iz intervala $[0.5, 1.5]$, a α poprima vrijednosti iz intervala $[-90°, +90°]$. Inkrementi za faktore S_x i S_y iznose 0.1, a za faktor α $10°$.



(a) Početna slika **(b)** $S_x = 0.8$, $S_y = 0.6$, $\alpha = 60°$ **(c)** $S_x = 0.9$, $S_y = 0.8$, $\alpha = 10°$ **(d)** $S_x = 1.0$, $S_y = 0.5$, $\alpha = 30°$

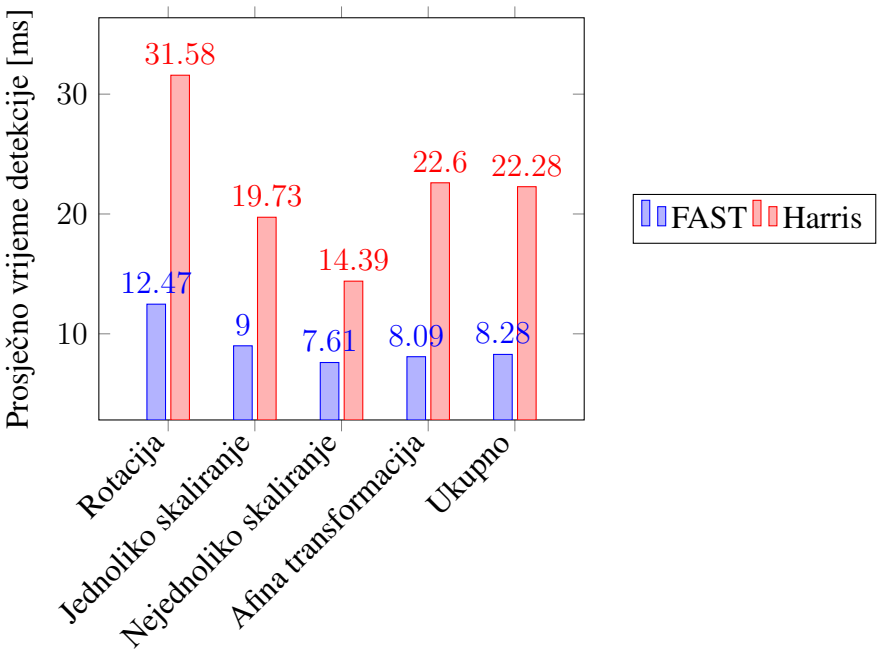
Slika 12: Primjer slika ispitnog skupa eksperimenta 4



Slika 13: Rezultati izvođenja eksperimenta 4

5.1.5. Vremenska usporedba postupaka

Uz praćenje rezultata izvođenja prethodnih eksperimenata, također je bilježeno vrijeme potrebno za detekciju na svakoj slici te su u nastavku prikazana prosječna vremena detekcije za svaki postupak detekcije na svakom eksperimentu zasebno te prosječno vrijeme svih obavljenih detekcija za svaki postupak.



Slika 14: Vremenska usporedba postupaka

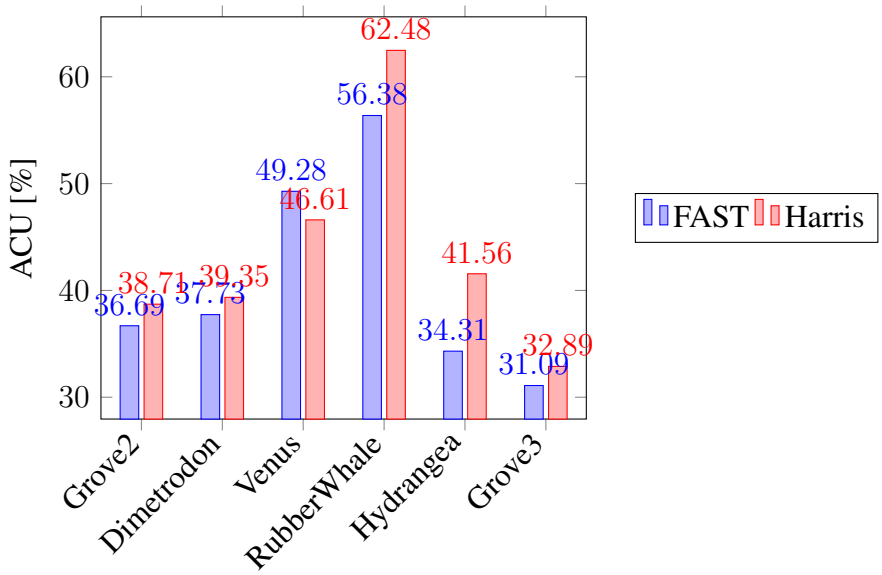
Primjetimo da su prosječna vremena za neke eksperimente značajno niža od ostalih. Razlog tome je što vrijeme detekcije izravno ovisi o veličini slike pa su tako eksperimenti koji skaliraju sliku na manje vrijednosti od početne ujedno i vremenski manje zahtjevni.

5.2. Eksperimentalna evaluacija preciznosti detekcije

Druga je skupina eksperimenata namijenjena je evaluaciji preciznosti detekcije značajki. Postupci za detekciju značajki evaluirani su na način da se pomoću njih izvodio postupak za praćenje značajki na testnim parovima slika. Na početnoj (lijevoj) slici detektirane su istaknute značajke. Zatim se na završnoj (desnoj) slici pokušavaju pronaći značajke istovjetne onima sa lijeve slike. Ukoliko se značajki sa lijeve slike pronađe odgovarajuća značajka na desnoj slici, praćenje se proglašava uspješnim, a pomak između njih se evaluira u odnosu na točne vrijednosti pomaka svakoga slikovnog elementa.. Isti eksperimenti, na istim parovima te sa istim postupkom praćenja, su izvršeni 2 puta, prvi put sa postupkom FAST kao postupkom za pronalazak praćenih točaka, a drugi put sa Harrisovim detektorom kutova u istoj ulozi. Kriterij usporedbe preuzet je iz [5] te ima oblik:

$$ACU = 100 \times \frac{\frac{N_a}{N_0} + \frac{N_a}{N_g}}{2} \tag{17}$$

U gornjoj jednadžbi ACU predstavlja ocjenu preciznosti postupka detekcije, N_0 je broj putanja izračunatih postupkom praćenja, N_g je broj točaka za koje postoji točna vrijednost pomaka i veća je od granične, a N_a je broj točaka za koji se izračunati i stvarni pomak podudaraju. Vrijednost ACU uvijek će se kretati iz intervala $[0\%, 100\%]$. Ovime su kriterijem pokriveni i slučajevi u kojima postupak računa krive putanje ($\frac{N_a}{N_0}$ pada pa i ACU pada) te slučajevi kada se računaju putanje za premali broj točaka ($\frac{N_a}{N_g}$ pada pa i ACU pada). Idealan bi detektor imao vrijednost $ACU = 100\%$. Postupak je ponovljen na 6 parova slika te su rezultati prikazani u nastavku.



Slika 15: Rezultati evaluacije preciznosti detekcije

Treba uvidjeti da je ova skupina eksperimenata također osjetljiva i na pogreške u praćenju značajki te vrijednost ACU ne ovisi isključivo o izabranom postupku detekcije istaknutih točaka. Međutim, ukoliko se na istom ispitnom skupu koristi isti postupak praćenja, rezultati usporedbe postupaka detekcije međusobno su usporedivi.

5.3. Analiza eksperimentalnih rezultata

Na temelju provedenih eksperimenata možemo izračunati prosječne vrijednosti te standardne devijacije vrijednosti CCN te ACU za oba detektora.

	Harris	FAST
Rotacija	42.26%	44.19%
Jendoliko skaliranje	22.56%	14.34%
Nejednoliko skaliranje	21.63%	18.33%
Afina transformacija	11.45%	5.65%
Prosječan CCN	24.45%	20.63%
Standardna devijacija	11.15%	14.35%

Tablica 1: Srednje vrijednosti i devijacije konzistencije detektora značajki

	Harris	FAST
Prosječan ACU	43.60%	40.91%
Standardna devijacija	9.37%	8.92%

Tablica 2: Srednje vrijednosti i devijacije konzistencije detektora značajki

Iz navedenih je vrijednosti vidljivo da niti jedan od postupaka osjetno ne odskače od onog drugog te da razlike ovise o konkretnim eksperimentima i testnim primjerima. Harrisov detektor kutova pokazao se marginalno boljim na oba seta eksperimenata. Međutim, što se brzine izvođenja tiče, postupak FAST je u svakom eksperimentu posao obavljao osjetno brže.

6. Zaključak

Ovaj rad je prvenstveno razmatrao eksperimentalnu evaluaciju postupaka za detekciju istaknutih značajki slike. Iako su u samome radu evaluirana dva konkretna postupka detekcije, implementacija je napravljena na način da je omogućena evaluacija proizvoljnog postupka detekcije, uz uvjet da komponenta koja se evaluira nasljeđuje zadano apstraktno sučelje ili da postoji komponenta koja omata testirani postupak kako bi ga se moglo koristiti uz postojeće komponente.

Evaluacijom je postupaka FAST i Harrisovog detektora kutova zaključeno da su oba postupka imala vrlo bliske rezultate, u određenim ispitnim slučajevima boljim se pokazao FAST, a u ostalima Harrisov detektor kutova. U prosječnom je slučaju Harrisov detektor kutova za 3.82% konzistentniji te za 2.69% precizniji od postupka FAST. Međutim, postupak FAST se pokazao vremenski efikasnijim na svakoj od ispitnih slika te je u prosječnom slučaju brži za 269%.

U budućem bi se radu uspješnost evaluacije mogla poboljšati dodavanjem realnih ispitnih slučajeva iz stvarnog svijeta, a ne samo korištenjem umjetno transformiranih slika. Također, na predloženi načine evaluacije i koristeći implementaciju razvijenu u sklopu ovoga rada, moglo bi se evaluirati još nekolicina detektora te na taj način uočiti postupak još prikladniji za neku od brojnih primjena pronalaska značajki.

LITERATURA

[1] Simon Baker, Daniel Scharstein, J.P. Lewis, Stefan Roth, Michael J. Black, i Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011. ISSN 0920-5691. doi: 10.1007/s11263-010-0390-2. URL `http://dx.doi.org/10.1007/s11263-010-0390-2`.

[2] G. Bradski. Opencv library. *Dr. Dobb's Journal of Software Tools*, 2000.

[3] Chris Harris i Mike Stephens. A combined corner and edge detector. U *Alvey vision conference*, svezak 15, stranica 50. Manchester, UK, 1988.

[4] Ivan Krešo. Napredno estimiranje strukture i gibanja kalibriranim parom kamera. Diplomski rad, Sveučilište u Zagrebu, Fakultet Elektrotehnike i Računarstva, 2013.

[5] Farzin Mokhtarian i Farahnaz Mohanna. Performance evaluation of corner detectors using consistency and accuracy measures. *Computer Vision and Image Understanding*, 102(1):81–94, 2006.

[6] V Rodehorst i A Koschan. Comparison and evaluation of feature point detectors. U *Proc. 5th International Symposium Turkish-German Joint Geodetic Days" Geodesy and Geoinformation in the Service of our Daily Life", Berlin, Germany*, 2006.

[7] Edward Rosten i Tom Drummond. Fusing points and lines for high performance tracking. U *IEEE International Conference on Computer Vision*, svezak 2, stranice 1508–1511, October 2005. doi: 10.1109/ICCV.2005.104. URL `http://edwardrosten.com/work/rosten_2005_tracking.pdf`.

[8] Edward Rosten i Tom Drummond. Machine learning for high-speed corner detection. U *European Conference on Computer Vision*, svezak 1, stranice 430–443,

May 2006. doi: 10.1007/11744023_34. URL http://edwardrosten.com/work/rosten_2006_machine.pdf.

[9] Richard Szeliski. *Computer vision: algorithms and applications*. Springer, 2010.

Eksperimentalno vrednovanje strojno naučenog postupka za pronalaženje istaknutih značajki slike

Sažetak

Rad razmatra i evaluira postupke čija je zadaća pronalaženje istaknutih značajki slike. Usporedno su evaluirani postupci FAST, strojno naučeni postupak za pronalaženje istaknutih značajki, te Harrisov detektor kutova. U okviru rada je razvijena komponenta koja omata FAST postupak za pronalaženje istaknutih značajki iz biblioteke OpenCV te implementira zadano apstraktno sučelje kako bi se mogla koristiti umjesto postojećih komponenata za pronalaženje značajki. Navedena se komponenta eksperimentalno vrednuje koristeći postojeće komponente za praćenje značajki na sljedovima slika s poznatim optičkim tokom te se ocjenjuje odnos rezultata dobivenih uporabom novorazvijene komponente u odnosu na rezultate dobivene uporabom postojeće komponente za pronalaženje značajki. U usporedbi se rezultata razmatraju ponovljivost detekcije značajki i konzistentnost postupaka detekcije, preciznost detekcije značajki putem evaluacije preciznosti praćenja značajki te vrijeme potrebno za pronalazak značajki na pojedinim slikama. Za provođenje se eksperimenata koristi postojeći ispitni skup sljedova slika namijenjen za evaluaciju postupaka praćenja optičkog toka koji je objavio Middlebury College.

Ključne riječi: pronalaženje značajki; praćenje značajki; postupak FAST; eksperimentalno vrednovanje postupaka; konzistentnost postupaka detekcije; preciznost postupaka detekcije

Experimental evaluation of a machine learned algorithm for detecting distinctive image features

Abstract

This paper examines and evaluates algorithms whose task is detecting distinctive image features. FAST, machine learned algorithm for feature detection, and Harris corner detector are being evaluated simultaneously. As a part of this paper, there was developed an adapter component wrapping OpenCV implementation of FAST algorithm and enabling its use instead of existing feature detectors. This component is experimentally evaluated using the existing feature tracking system on image sequences with known optical flow values and results are compared with those using Harris corner detector instead. The comparison results focus on repeatability and consistency values of feature detectors, the accuracy tracking using said detectors and on analysing time required for detection on each image. All experiments are conducted on image database consisting of image sequences made for evaluation of optical flow calculation algorithms provided by Middlebury College.

Keywords: feature detection; feature tracking; FAST algorithm; experimental evaluation of algorithms; feature detection consistency; feature detection accuracy