

**Sveučilište u Zagrebu
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
Zavod za elektroniku, mikroelektroniku, računalne i inteligentne sustave**

**Projekt iz predmeta Raspoznavanje uzorka
Ak. god. 2015./16.**

**Podudaranje slika uporabom bilateralne
lokalne simetrije**

Studenti:

Domagoj Boroš
Matea Cerovac
Andrea Gradečak
Ivan Hrastinski
Ivan Jurin
Josip Knežević

siječanj, 2016.

Sadržaj

1. Projektni zadatak.....	4
1.1 Opis projektnog zadatka.....	4
1.2 Pregled i opis srodnih rješenja	6
1.2.1 Daniel Cabrini Hauagge, Noah Snavely: <i>Image Matching using Local Symmetry Features</i>	6
1.2.2 Eli Shechtman, Michal Irani: <i>Matching Local Self-Similarities across Images and Videos</i>	7
1.2.3 Zhenhua Wang, Bin Fan, Fuchao Wu: <i>Local Intensity Order Pattern for Feature Description</i>	7
1.2.4 Scale Invariant Feature Transform (SIFT)	8
1.3 Konceptualno rješenje zadatka	9
2. Postupak rješavanja zadatka.....	12
2.1 Predobrada slike.....	12
2.1.1 Gaussovo zamućivanje.....	12
2.1.3 Računanje gradijenata	15
2.1.4 Uklanjanje nepotrebnih dijelova slike	17
2.2 Stvaranje značajki i opisnika	19
2.2.1 Izračun rezultata simetrije	19
2.2.2 Filtriranje kandidata za značajke.....	19
2.2.3 Odabir značajki	20
2.2.4 Stvaranje opisnika.....	20
2.3 K najблиžih susjeda.....	21
2.3.1 Metoda K najблиžih susjeda	21
2.4 Klasifikacija.....	22
2.4.1 Klasifikacija	22
3. Ispitivanje rješenja	23
3.1 Ispitna baza	23
3.2 Rezultati učenja i ispitivanja	24
3.2.1 Slike urbane arhitekture	24
3.2.2 Slučajevi podudaranja testnih slika urbane arhitekture	25
3.2.3 Slike lica.....	27
3.3.1 Usporedba rezultata sa SIFT značajkama.....	29
3.3 Ostali testirani pokušaji konstrukcije opisnika za postupak podudaranja korištenjem lokalnih simetrija	31
3.3.1 Prvi pokušaj - Konstrukcija 128-bitnog vektora	31
3.3.2 Drugi pokušaj – korištenje Harrisovog detektora.....	32
3.3.3 Konačan odabir izgradnje opisnika	33

4.	Opis programske implementacije rješenja.....	35
4.1	Korištenje implementacije.....	35
4.2	Opis programskog rješenja.....	36
4.2.1	MagentoClassifier	37
4.2.2	ImageLoader.....	37
4.2.3	Feature	37
4.2.4	Building	37
4.2.5	Image	37
4.2.6	ORBIImage	38
4.2.7	OpenImage	38
4.2.8	PyramidImage.....	38
4.2.9	PyramidFaceImage.....	38
4.2.10	PyramidLevel	38
5.	Zaključak	39
6.	Literatura	40

1. Projektni zadatak

1.1 Opis projektnog zadatka

Zadatak ovog projekta je implementacija postupka koji pomoću izračuna lokalne simetrije pronalazi točke koje se podudaraju na različitim slikama koje prikazuju isti objekt. Takav je postupak potrebno testirati na slikama arhitekture i kasnije licima ljudi. Inicijalno se postupak napravio za testiranje na slikama urbane arhitekture ali se kasnije pokazao dosta dobrim i za podudaranje slika ljudskih lica.

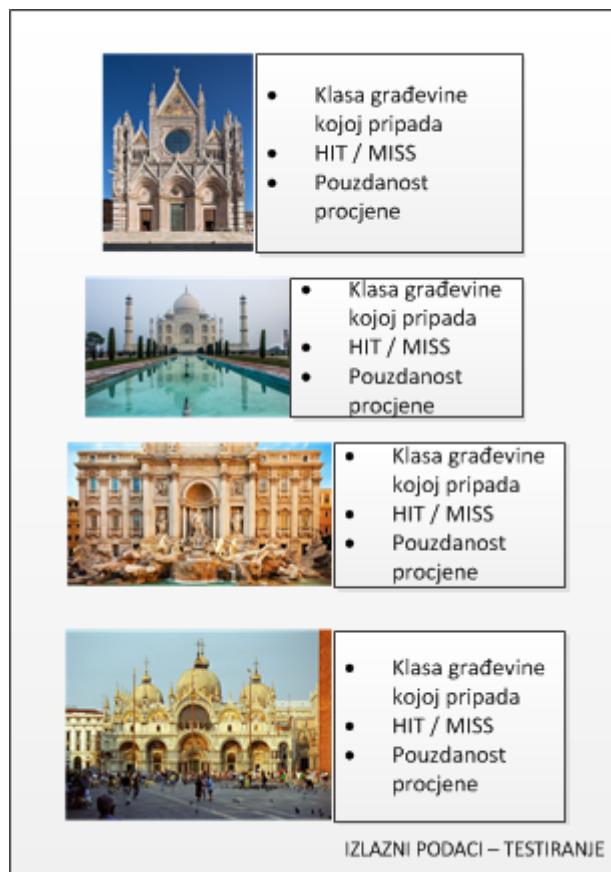
S obzirom na složenost problema, potrebno je osmisliti postupak koji će pronalaziti lokalne simetrije u slikama, spremati pronađene točke i značajke koje simboliziraju lokalnu simetriju nad tim točkama u opisnike i koristiti klasifikator koji će ispravno klasificirati pojedine testne slike s obzirom na njihovu pripadnost nekoj klasi koju predstavljaju.

Ulagni podaci dakle predstavljaju slike arhitekture i ljudska lica. Kod ulaznih podataka potrebno je imati na raspolaganju slike više različitih zgrada i ljudi. Takve se slike podijele u zasebne klase u ovisnosti o tome koju građevinu ili čije ljudsko lice predstavljaju. Primjerice, slika 1 prikazuje unutarnju organizaciju ulaznih podataka; kod slika urbane arhitekture, tj slika građevina, svaka je građevina zasebna klasa.



Slika 1 Unutarnja organizacija skupa slika za učenje i testiranje

Izlazni podaci su podaci o pripadnosti testnih slika arhitektura ili ljudskih lica navedenim klasama. Ti podaci, koji su pregledno prikazani na slici 2, sadrže vjerojatnost pripadanja pojedinoj klasi svake testne slike, klasu koju je klasifikator odredio temeljem postupka i podatak je li klasa pogodena ili promašena. Također, na temelju korištenja različitih klasa slika i varirajući njihov broj trebalo bi testirati rad postupka pronalaženja podudaranja klase slika urbane arhitekture i ljudskih lica koristeći značajke dobivene pronalaženjem lokalnih simetrija na slikama.



Slika 2 Pregledni prikaz izlaznih podataka dobivenih nakon provođenja postupka

U sklopu pronalaženja lokalnih simetrija u slikama, zbog bržeg, učinkovitijeg i boljeg izlučivanja značajki i opisnika lokalnih simetrija, trebalo se iskoristiti proces preprocesiranja slika za učenje i testiranje.

Nakon što se obavi preprocesiranje, potrebno je gledati u smjeru pronalaženja postupka koji bi sadržavao jednostavan algoritam za brzo traženje lokalnih simetrija u slikama i stvaranje jednostavnih značajki ili opisnika koji će te lokalne simetrije opisati

na jednostavan način, kako bi ih klasifikator mogao dalje iskoristiti za učenje i predikciju rezultata.

S obzirom da na bi na raspolaganju mogli imati mnogo primjera za učenje, korisno je razmišljati u smjeru klasifikatora ili postupka koji bi učio klasifikaciju na primjerima za učenje, koji u svojoj biti predstavljaju naše opisnike lokalnih simetrija u slikama dobivene prethodnim korakom. Tu zapravo dobivamo odluku o odabiru između parametarskih i neparametarskih modela, i probabilističkih i neprobabilističkih modela. U dostupnoj literaturi [5], navedeni su neki od algoritama koji bi se mogli iskoristiti za probleme slične našima.

Jedan od jednostavnih, već implementiranih i dostupnih algoritama je algoritam K najbližih susjeda koji pronalazi za svaki dani primjer, K njegovih najbližih susjeda u euklidskom prostoru koristeći euklidsku normu. Zbog primjera u literaturi [5], odlučili smo se za algoritam K-najbližih susjeda kao klasifikator.

1.2 Pregled i opis srodnih rješenja

U nastavku slijedi opis radova srodnih tema koji su bili korisni tijekom izrade ovog rada.

1.2.1 Daniel Cabrini Hauagge, Noah Snavely: *Image Matching using Local Symmetry Features*

U ovom radu predstavljena je tehnika izdvajanja lokalnih značajki na slikama urbane arhitekture korištenjem lokalne simetrije. Lokalne simetrije su osnovna karakteristika mnogih zgrada urbane arhitekture. Također, ova tehnika je uzeta zbog toga što su takve značajke nepromjenjive, odnosno, manje su osjetljive na vanjske promjene poput osvjetljenja na slici, nego neki poznati algoritmi poput SIFT-a (engl. Scale invariant feature transform), koji ipak daju lošije rezultate u takvim uvjetima.

Značajke se zasnivaju na jednostavnim mjerama lokalne bilateralne i rotacijske simetrije koje su izračunate koristeći lokalne operacije nad slikom. Ove mjere su korištene i za detekciju značajki, kao i za računanje opisnika.

Uz rad priložen je i skup slika [2] nad kojim je učena i testirana ova metoda. Jedna klasa se sastoji od dvije slike koje su različite po nekoj karakteristici: osvjetljenje, starost građevine, prikaz slike. Takve različitosti su uvedene u slike upravo zbog toga što se želi dokazati da one ne utječu na lokalne simetrije.

Unutar rada, osim s algoritmom SIFT napravljena je usporedba s algoritmom MSER te se pokazalo da su lošiji od algoritma predstavljenog u ovome radu.

Ovaj rad poslužio je kao kostur našega projekta. Također, jedan od načina evaluacije našeg projekta je isti kao u ovom radu, odnosno uzete su po dvije slike te se traže značajke s jedne slike na drugoj.

1.2.2 Eli Shechtman, Michal Irani: *Matching Local Self-Similarities across Images and Videos*

U ovom radu prikazan je pristup za mjerjenje sličnosti između vizualnih entiteta (slika ili videa) koji se zasniva na poklapanju internih samosličnosti.

Ove interne samosličnosti su efikasno nađene sa kompaktnim lokalnim samosličnim opisnikom. Ovaj način omogućava podudaranje kompleksnih vizualnih podataka, što uključuje detekciju objekata u pretrpanim (engl. *cluttered*) slikama koristeći samo ručne skice, upravljanje teksturiranim objektima koji nemaju jasne granice te detekciju kompleksnih akcija u pretrpanim videima bez prethodnog učenja nekog klasifikatora.

U radu je provedena usporedba između predložene mjere i nekih poznatih mjera sličnosti te se demonstrira primjenjivost na detekciju objekata te detekciju akcija.

1.2.3 Zhenhua Wang, Bin Fan, Fuchao Wu: *Local Intensity Order Pattern for Feature Description*

Ovaj radpredstavlja metodu za opis značajki koja se zasniva na redu intenziteta. LIOP (engl. *Local Intensity Order Pattern*) šifrira lokalnu rednu informaciju za svaki slikovni element i ukupna redna informacija je korištena za podjelu lokalnih dijelova slike u podregije koje su korištene za akumuliranje LIOP-a.

Dakle, i lokalna i ukupna informacija redna informacija o lokalnom dijelu će biti nađena LIOP opisnikom te je zbog toga opisnik jako diskriminativan. Pokazano je da ovaj opisnik nije samo nepromjenjiv na monotone promjene intenziteta i rotaciju slike, nego i na brojne geometrijske i fotometrijske transformacije kao što su promjena gledišta, zamućivanje slike te kompresija JPEG formata.

Opisnik je evaluiran nad Oxfordovim skupom slika [12] te na četiri dodatna para slika s kompleksnim promjenama osvjetljenja.

Rezultati su pokazali da je opisnik postigao puno bolje rezultate u usporedbi sa *state-of-the-art* opisnicima.

1.2.4 Scale Invariant Feature Transform (SIFT)

SIFT (engl. Scale-Invariant Feature Transform), [6] je jedan od algoritama koji se upotrebljava za neke od zadaća računalnog vida. SIFT, naime, služi za pronalazak i opisivanje lokalnih značajki neke slike. Ono što je pritom posebno važno jest njegova invarijantnost s obzirom na rotaciju i uvećanje odnosno smanjenje slike.

Također je i relativno otporan na promjene u osvjetljenju na slici i promjene točke gledišta te zaklonjenost objekata. Ovaj algoritam ima velike mogućnosti uporabe kod uparivanja odnosno prepoznavanja te kod 3 - dimenzionalnog rekonstruiranja scene. Omogućava uspoređivanje sa relativno opsežnim bazama podataka u približno realnom vremenu (ovisno o računalnom sustavu).

Provedba algoritma sastoji se od 4 koraka:

1. Izgradnja prostora mjerila
2. Detekcija i lokalizacija značajki

-
3. Dodjela orientacije
 4. Izgradnja opisnika

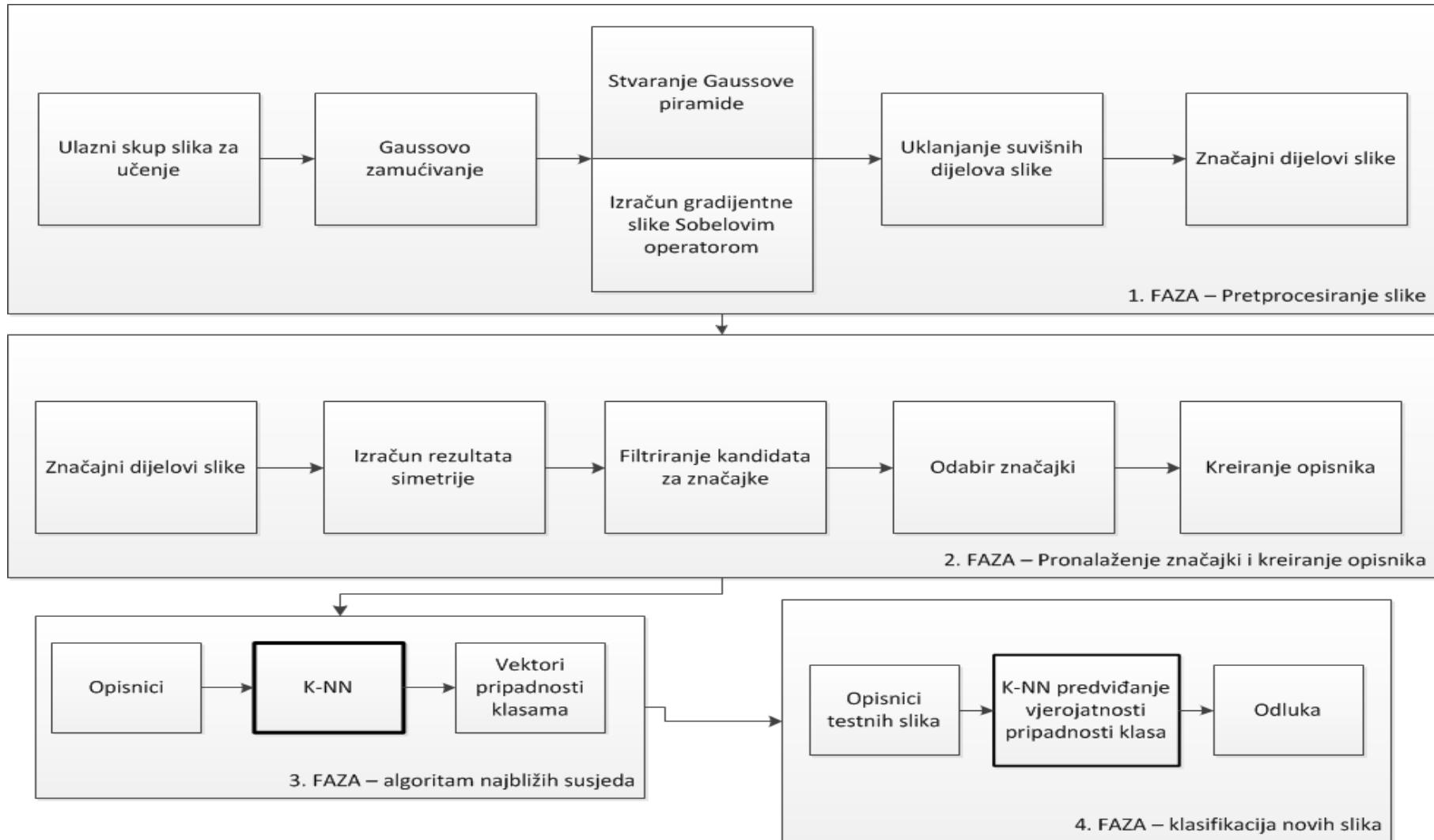
1.3 Konceptualno rješenje zadatka

U rješavanju zadatka opisanog u poglavljiju 1.1 korišten je sljedeći niz algoritama:

- **Gaussovo zamućivanje** (engl. *Gaussian blur*) [13] koje stvara mutniju sliku s manje detalja, uz očuvanje njezinih glavnih obilježja. Primijenjeno je u početnoj obradi slike i prilikom stvaranja Gaussove piramide.
- **Gaussova piramida** [14], se sastoji od više jednakih slika pri čemu svaka sljedeća prolazi kroz niskopropusni filter (Gaussovu jezgru) što uzrokuje smanjivanje njezinih dimenzija i zamućivanje. Za svaku sliku koja se koristi stvara se spomenuta piramida s pripadnim razinama.
- **Računanje gradijenata** korištenjem Sobelovog operatora [8][9][10] računa gradijent najprije za vertikalnu i horizontalnu os slike, a potom pomoću dobivenih rezultata i njegovu konačnu aproksimaciju. U ovom zadatku upotrijebljen je Sobelov operator dimenzija 5x5. Svaka slika, odnosno svaka razina Gaussove piramide definirana je svojstvima gradijenata izračunatih primjenom Sobelovog operatora.
- **Uklanjanje nepotrebnih dijelova slike** se postiže izračunom značajnosti svakog slikovnog elementa na slici. Težina određenog slikovnog elementa ovisi o vrijednosti njegovog gradijenta i temeljem toga se uklanjaju suvišni dijelovi slike kao što su pozadina ili monotoni dijelovi građevina.
- **Izračun rezultata simetrije** temelji se na rezultatima prethodne analize iz kojih su dobiveni dijelovi slike koji su se pokazali kao značajni. Prolaskom kroz te dijelove promatra se okolina svakog slikovnog elementa definirana prozorom i uspoređuju se vrijednosti gradijenata svakog slikovnog elementa zahvaćenog tim prozorom. Iz rezultata ove analize svakom slikovnom elementu se dodjeljuje težina s obzirom na mogućnost da upravo on bude odabran kao značajka.

-
- **Filtriranje kandidata za značajke** prolazi kroz rezultate simetrije za svaki slikovni element i odbacuje okolne slikovne elemente koji imaju veću vrijednost od onog kojeg promatramo. Ovakvim postupkom ne postoji mogućnost da od grupe slikovnih elemenata na istom području i s vrlo sličnim rezultatima simetrije odaberemo cijelu grupu i pritom dobijemo nakupinu sličnih značajki, već odabiremo samo lokalne minimume.
 - **Značajke** na svakoj slici **odabiru** se postupkom pronađenja 100 slikovnih elemenata za koje je izračun simetrije dao najmanju vrijednost.
 - **Stvaranje opisnika** je postupak izračuna vektora koji predstavlja značajku. Pomoću opisnika metoda K najbližih susjeda računa udaljenost opisnika od ostalih opisnika značajki. Prozor svakog slikovnog elementa koji je odabran kao značajka se podijeli na 4 jednakih dijela. Za svaki od tih dijelova je stvoren histogram od 8 mogućih vrijednosti koje su definirane intervalima kuteva. Za svaki od 4 dijelova se vrijednosti kuteva u tom dijelu svrstavaju u 8 intervala na koje je podijeljen. Histogram nam daje broj kuteva u svakom od 8 intervala (8 brojeva). Rezultati za sva 4 dijela prozora zajedno tvore 32-dimenzijski vektor koji predstavlja opisnik za tu značajku.
 - **K najbližih susjeda** je metoda korištena iz biblioteke sklearn [15]. Pomoću metode K najbližih susjeda za svaki se opisnik na slici koja se ispituje kreira vektor vjerojatnosti pripadnosti svakoj klasi.
 - **Klasifikacija** nove slike se sastoji od izračuna opisnika prema gore navedenim koracima, poziva metodu K najbližih susjeda i analizom vjerojatnosti koju nam ona vraća. U ovom radu koristi se metoda koja zbraja vjerojatnosti opisnika za svaku klasu te se slika klasificira u onu klasu čiji je zbroj vjerojatnosti najveći.

Na slici 3, vidljiv je pregledan prikaz dijagrama toka osmišljenog postupka:



Slika 3 Dijagram toka osmišljenog postupka

2. Postupak rješavanja zadatka

2.1 Predobrada slike

Prvu fazu našeg postupaka čini predobrada svake slike iz skupa za učenje.

Predobrada slike ima više koraka:

1. Prebacivanje slike u rezoluciju 640x480 slikovnih elemenata
2. Prebacivanje slike u boji u sivu sliku
3. Gaussovo zamućivanje i stvaranje Gaussove piramide
4. Računanje gradijenata pomoću Sobelovog operatora
5. Uklanjanje nepotrebnih dijelova slike

2.1.1 Gaussovo zamućivanje

Ulas: stvarna slika

Gaussovo zamućivanje [13] je proces zamućivanja slike korištenjem Gaussove funkcije u jednoj ili više dimenzija, u cilju smanjivanja slikovnog šuma i ujednačavanja slike. Vizualni efekt koji se dobiva procesom Gaussovog zamućivanja je sličan fotografijama koje kao da su dobivene snimkom detalja kroz zamućeno staklo.

Vjerojatnosna gustoća Gaussove funkcije, za jednu dimenziju se matematički može zapisati ovako:

$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

gdje su σ^2 varijanca i μ srednja vrijednost funkcije.

Za više dimenzija, matematički zapis gustoće vjerojatnosti Gaussove funkcije je ovakvog tipa:

$$f_x(x_1, \dots, x_n) = \frac{1}{\sqrt{(2\pi)^k |\Sigma|}} \exp(-0.5(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})),$$

gdje su $\boldsymbol{\Sigma}$, $\boldsymbol{\mu}$ i \mathbf{x} kovarijacijska matrica, vektor srednjih vrijednosti i realni vektor ulaznih vrijednosti, respektivno.

Matematički, provođenje spomenutog postupka jednako je konvoluciji slike s Gaussovom funkcijom. Gaussova funkcija se izračunava prethodno i sprema se u matricu (engl. *kernel*) nakon čega vršimo konvoluciju te matrice s matricom koja predstavlja sliku. Matematički postupak možemo zapisati ovako:

$$G = \begin{bmatrix} 1 & 2 & 1 \\ -1 & 1 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\text{Zamućena slika} = G * \text{Početna slika}$$

gdje je G Gaussova funkcija, a $*$ operator konvolucije.

Gaussovo zamućivanje ima učinak smanjivanja šuma na slici tako da smanjuje visoke frekvencije, pa ga možemo zvati i niskopropusnim filtrom.

Ulagni podaci za ovu fazu postupka su skalirane sive slike promijenjene rezolucije. Na slici 4 vidi se cijelokupni postupak ove faze; nad početnom slikom u boji sprovedeno je prebacivanje u sivu sliku, te je nad takvom slikom napravljeno Gaussovo zamućivanje.



Slika 4 Gaussovsko zamućenje

Izlaz: slika s primjenjenim Gaussovim zamućenjem

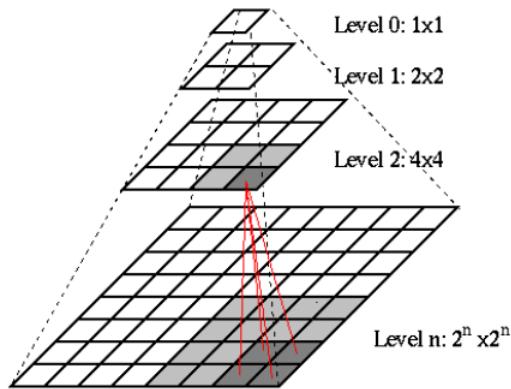
2.1.2 Stvaranje Gaussove piramide

Ulagni: slika s primjenjenim Gaussovim zamućenjem

U svrhu traženja simetrije na različitim veličinama prozora, iskoristili smo postupak stvaranja tzv. Gaussovinih piramida, [14]. Taj se postupak radi na sljedeći način: Iz početnog uzorka E , propuštanjem kroz niskopropusni filter koji zapravo predstavlja Gaussovo zamućivanje, dobivaju se slike manjih dimenzija. Zamućenost slika svakim

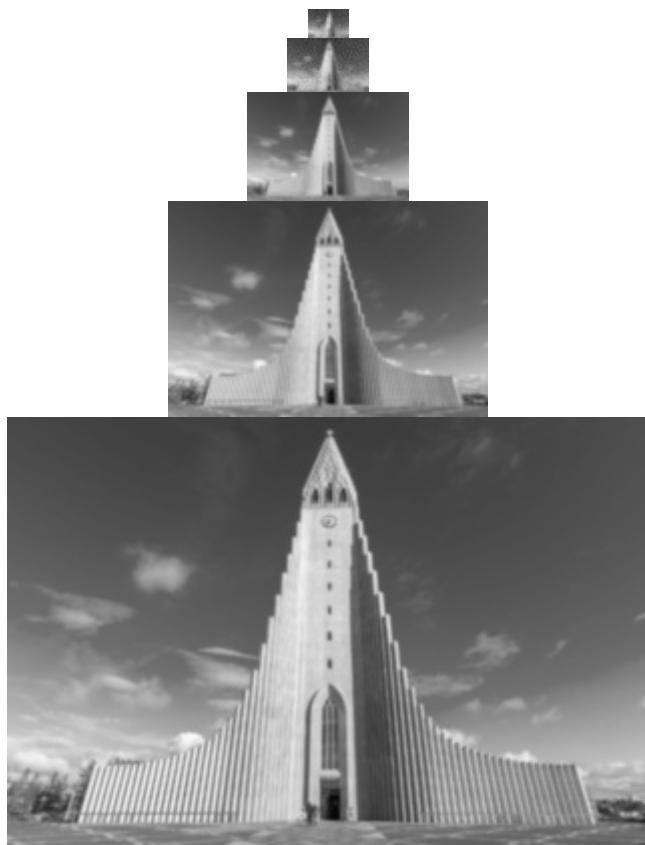
korakom sve više raste, ali se glavni detalji slike koji će se iskoristiti i dalje preslikavaju, kao što je vidljivo na slici 5.

Prednost i cilj ovakvog postupka je iskoristiti algoritam koji radi s fiksnom veličinom prozora koji bi ocjenjivao simetrije na različitim veličinama, od detaljnijih (lokalnijih) do onih globalnijih. Alternativni pristup bi bio povećanje prozora kojim bi se utvrđivala simetrija, što bi značilo veću složenost i imali bi neusporedive značajke s obzirom na veličinu prozora. Korištenjem Gaussova piramida dobivamo još jedno bitno poboljšanje, a to je da možemo uspoređivati slike građevina i ljudskih obraza koje su dobivene različitim udaljenostima kamere od objekta.



Slika 5 Primjer Gaussove piramide

Na slici 6 je vidljiv primjer ulaznog podatka u fazu računanja Gaussova piramida i dobivanja izlaza:



Slika 6 Dobivena Gaussova piramida

Izlaz: Gaussova piramida

2.1.3 Računanje gradijenata

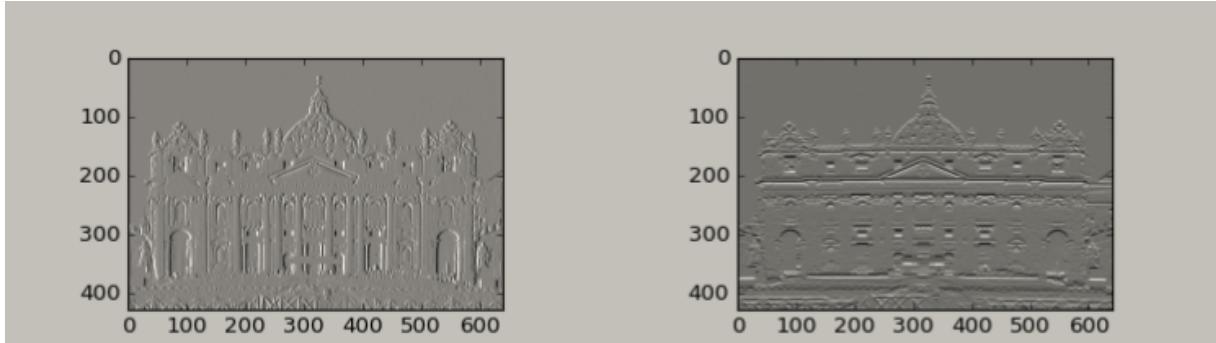
Uaz: Gaussova piramida

Za računanje gradijentne slike iskoristili smo Sobelov operator[8][9][10] koji zapravo računa aproksimaciju slikovnih intenziteta funkcije. U svakoj točki slike, općeniti rezultat Sobelovog operatora je ili odgovarajući vektor gradijenata ili norma tog vektora. Sobelov operator je zapravo konvolucija početne slike s malim filterom po horizontalnoj i vertikalnoj osi .

Tako se dobije gradijent u svakoj točki u sustavu (X,Y) što se u svrhu daljnje analize transformira u sustav (R, ϕ).

Sobelov operator je kompozicija dva filtera, Gaussovog zamućivanja i filtera koji radi derivaciju same slike. Tako dijelovi slike koji imaju "najbržu" promjenu vrijednosti slikovnih elemenata imaju veću vrijednost od slikovnih elemenata koji su gotovo

jednakih vrijednosti svojim susjednim slikovnim elementima. Na slici 7 vidljiva je upotreba Sobelovog operatora po apscisi i ordinati.



Slika 7 Sobelov operator nad vertikalnom i horizontalnom osi

Sobelov operator koji djeluje po horizontalnoj osi se matematički može opisati ovako:

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * M$$

gdje je $*$ operator konvolucije, a M matrica koja predstavlja sliku nad kojom primjenjujemo Sobelov operator. Na analogni način dobivamo i rezultat primjene Sobelovog operatorka nad vertikalnom osi:

$$Gy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * M$$

Sobelov operator koristimo za izlučivanje značajnih dijelova slika nad kojima želimo pronaći simetrije. Stoga zbog jednostavnijeg računanja značajki i kasnije samih opisnika, pretvaramo gradijente slikovnih elemenata dobivene Sobelovim operatorom u bolji zapis, prelaskom u kompleksnu ravninu, na ovaj način:

$$|G| = \sqrt{G_x^2 + G_y^2}$$

gdje je $|G|$ norma vektora gradijenta, i

$$\varphi = \tan^{-1} \frac{Gy}{Gx}$$

gdje je φ argument, a G_x i G_y dobiveni vertikalni i horizontalni gradijent u prvom koraku.

Nedostatak korištenja ovog pristupa su računalno skupe operacije korjenovanja zbroja horizontalnog i vertikalnog gradijenta, i operacija pronalaženja argumenta kompleksne ravnine, zbog korištenja trigonometrijske funkcije arkus tangens.

Izlaz: vrijednosti gradijenata za svaki slikovni element

2.1.4 Uklanjanje nepotrebnih dijelova slike

Ulaz: vrijednosti gradijenata za svaki slikovni element

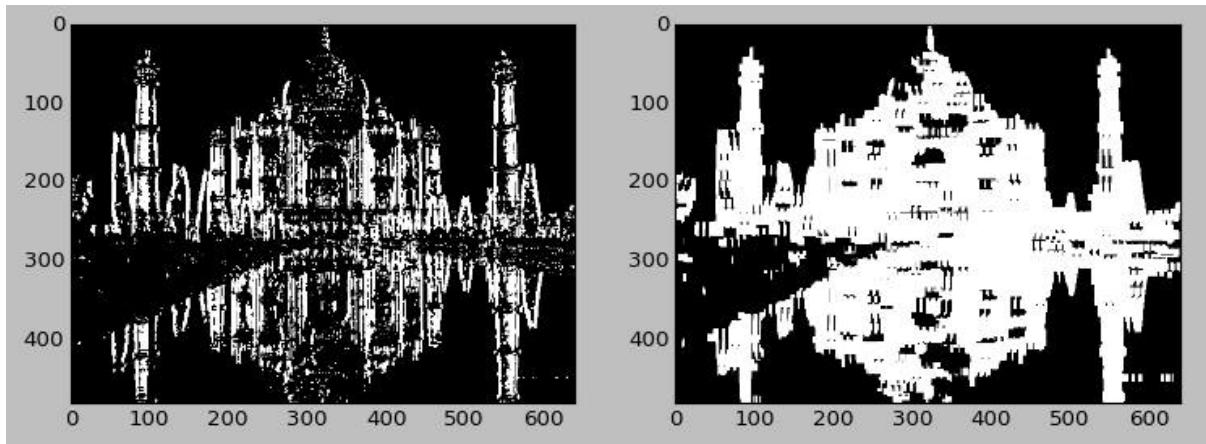
Kako bi se smanjilo područje slike na kojem se vrši pretraga za kvalitetnim horizontalno simetričnim značajkama, postavlja se uvjet nad svakim prozorom pretrage da mora sadržavati barem jedan gradijent iznad određene norme $R_{threshold}$ koji zatvara kut s x osi manji od zadanih $\varphi_{threshold}$. Oba parametra su hiperparametri sustava empirijski određeni na 500 i 30 stupnjeva respektivno. Norme gradijenata slike se transformiraju pomoću prvog hiperparametra funkcijom:

$$nove\ norme = \begin{cases} \frac{norme}{2 * R_{threshold}}, & \text{za } norme < R_{threshold} \\ 1, & \text{inače} \end{cases}$$

Tako se norme mapiraju u interval $[0,1]$ što će se dalje tretirati kao težina (važnost) pojedinog gradijenta.

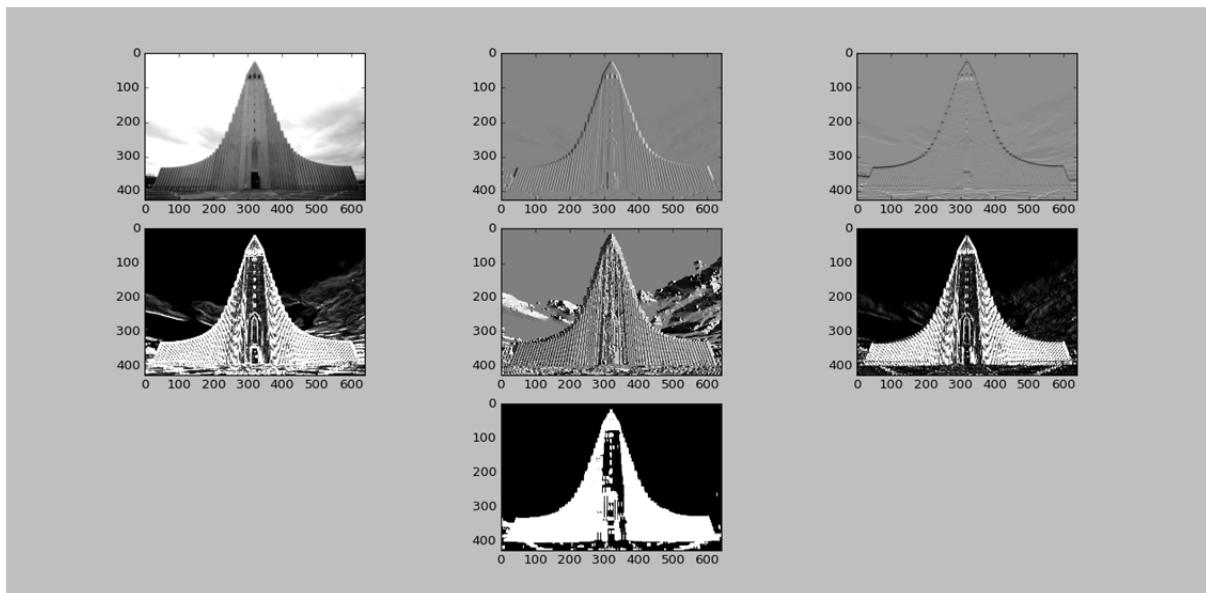
Izračunom matrice područja pretraživanja, normalizacijom gradijenata na opisan način te transformacijom kuteva na način da odgovaraju odmaku od y osi u smjeru kazaljke na satu završava navedena faza.

Izlaz: normalizirani gradijenti u (r,φ) sustavu i područje pretraživanja



Slika 8 Proširen prostor pretrage

Na slici 8 prikazana je dilatacija prostora pretraživanja. Od početnog prostora pretraživanja označenog bijelom bojom i dobivenog izračunavanjem gradijenata Sobelovim operatorom, manipulacijom norma i argumenata, dobivamo sliku desno, gdje imamo prošireni prostor pretraživanja lokalnih simetrija.



Slika 9 Detaljan prikaz koraka

Na slici 9, vidljiv je detaljan prikaz koraka smanjivanja prostora pretraživanja. Prva slika u prvom redu predstavlja ulaznu sivu sliku. Dvije slike desno od nje predstavljaju rezultat primjene vertikalnog i horizontalnog Sobelovog operatora, respektivno. U drugom redu, slijede slike, s lijeva na desno, koje predstavljaju prikaz izračunatih normi dobivenih Sobelovim operatorima, argumenata izračunatih Sobelovim

operatorima te naposljetu težine koje se dalje koriste u izračunu, respektivno. Zadnja slika predstavlja dozvoljenu površinu koja je označena bijelom bojom i nad kojom će se izračunavaju značajke.

2.2 *Stvaranje značajki i opisnika*

Značajke i pripadajući opisnici su nam potrebni za izvršavanje klasifikacije. Njihov izračun i stvaranje se odvija redom u sljedećim koracima:

1. Izračun rezultata simetrije
2. Filtriranje kandidata za značajke
3. Odabir značajki
4. Stvaranje opisnika

2.2.1 **Izračun rezultata simetrije**

Ulag: normalizirani gradijenti u (r,ϕ) sustavu i područje pretraživanja

Svaki slikovni element iz područja pretraživanja mapira se u njegovu ocjenu simetrije te se dobiva matrica ocjena. Uzima se prozor oko slikovnog elementa gore navedenih dimenzija te se unutar njega računa razlika kuteva gradijenata između slikovnih elemenata koje su jednako udaljeni od centra simetrije. Svaka razlika se transformira faktorom težine koji je jednak umnošku normaliziranih gradijenata na tim slikovnim elementima. Na kraju se uzima težinski prosjek tako dobivenih razlika. Tako dobivena ocjena je takva da odgovara prosječnoj apsolutnoj razlici u kutevima s lijeve i desne strane horizontalne simetrije. Kako bi preferirali one značajke koje imaju više podudarnih gradijenata ocjeni pridodajemo omjer suma težina u tom prozoru i veličine tog prozora skaliran za empirijski određeni faktor. Taj faktor je još jedan hiperparametar sustava te je empirijski određena njegova vrijednost na 10. Tako dobivenu matricu ocjena predajemo u idući korak algoritma.

Izlaz: matrica ocjene

2.2.2 **Filtriranje kandidata za značajke**

Ulag: matrica ocjene

Matrica ocjene koju smo dobili u prethodnom koraku sadrži vrijednosti za svaki slikovni element koja definira njegovu mogućnost da bude odabrana kao značajka. Pri tome manja vrijednost označava veću mogućnost odabira. Iz te matrice potrebno je izvući one slikovne elemente koji imaju najmanje vrijednosti. To se postiže metodom pronalaženja lokalnih minimuma.

Kroz matricu ocjene se prolazi prozorom veličine 3x3. U tom prozoru se traži najmanja vrijednost i samo te točke opstaju. Problem koji se javlja u ovakvom pristupu je mogućnost da su dvije vrijednosti jednake te se obje uzimaju kao potencijalni ili čak pravi kandidati za značajke.

Takav problem se može riješiti korištenjem adaptivne ne-ekstremalne supresije [16] (engl. *adaptive non-maximal suppression*), ali je u našem primjeru dala tek neznatno bolje rezultate uz osjetno duže vrijeme izvođenja.

Izlaz: binarna matrica s kandidatima za značajke

2.2.3 Odabir značajki

Ulaz: binarna matrica s kandidatima za značajke

U binarnoj matrici se odabere do 100 značajki koje imaju najmanju vrijednost. Koraci 2.2.1, 2.2.2 te odabir do 100 značajki se odvijaju nad slikom svake razine u Gaussovoj piramidi. Skup svih značajki jedne slike čine značajke slike sa svih razina Gaussove piramide.

Izlaz: skup značajki

2.2.4 Stvaranje opisnika

Ulaz: značajka

Informacije o okolini značajke, odnosno pripadajućeg slikovnog elementa, se dobivaju iz prozora. Prozor sadrži informacije o gradijentima njegovih slikovnih elemenata, odnosno njihove kuteve i norme. Za izračun deskriptora se u obzir uzimaju samo kutevi. Razlog tomu je što postoji preveliki utjecaj karakteristika slike (na primjer, kontrast ili svjetlina) na norme, a time i na vrijednosti opisnika.

Prozor značajke, veličine $(2 \cdot \text{širina}) \times (2 \cdot \text{visina} + 1)$, odnosno 16×9 , se podijeli na 4 jednakih dijela po širini. Za svaki od 4 dijela prozora se stvara histogram. Histogram definiraju 8 mogućih vrijednosti, odnosno "kutija", koje predstavljaju određeni interval kuteva na kružnici.

Potom se za sve kuteve u tom dijelu prozora svrstavaju vrijednosti kuteva gradijenta u pripadajući interval, odnosno "kutiju". Dakle, histogram pojedinog dijela nam daje informaciju koliko kuteva pripada svakom od 8 intervala. Rezultat svakog od 4 dijela je vektor koji se sastoji od tih 8 brojeva. Sva 4 dijela zajedno tvore vektor s 32 komponente, odnosno dimenzije i on čini opisnik te značajke.

Izlaz: opisnik (32-dimenzijski vektor)

2.3 *K najbližih susjeda*

2.3.1 Metoda K najbližih susjeda

Ulaz: opisnici

Metoda K najbližih susjeda daje rezultate na temelju sličnosti opisnika ispitne slike i svih drugih opisnika [15]. Korištena je implementacija K najbližih susjeda iz biblioteke sklearn, gdje je parametar K postavljen na vrijednost 10. Korištena metoda usporedbe za opisnike se definira u parametru 'weights' i postavljena je na vrijednost 'distance'.

Tako postavljeni parametar računa sličnost opisnika prema udaljenosti. Uzima se 10 najbližih opisnika i računa se udio njihove značajnosti obrnuto proporcionalan udaljenosti. Bliži susjedi tada imaju veći udio u odluci nego susjedi koji su udaljeniji. Zbrojeni udjeli za svaku klasu i potom skaliranje na vrijednosti u intervalu od 0 do 1 daju matricu u kojoj retci odgovaraju opisnicima ispitne slike, a stupci klasama. Svaki element matrice predstavlja vjerojatnost da određeni deskriptor pripada određenoj klasi.

Izlaz: Matrica vjerojatnosti pripadnosti opisnika svakoj klasi

2.4 Klasifikacija

2.4.1 Klasifikacija

Ulaz: Matrica vjerojatnosti pripadnosti opisnika svakoj klasi

Analiza matrice vjerojatnosti nam daje oznaku klase kojoj slika pripada. Isprobana su četiri načina analize matrice:

1. Za svaki opisnik se uzima klasa čija je vjerojatnost najveća. Klasa koja se najčešće pojavljuje se uzima kao klasa kojoj slika pripada.
2. Za svaki opisnik se uzima klasa čija je vjerojatnost najveća, ali samo ako je ta vjerojatnost veća od 0.5. Ako nije, taj opisnik ne sudjeluje u konačnoj klasifikaciji. Od onih koji ulaze u konačnu klasifikaciju se odabire najčešća klasa.
3. Za svaki opisnik se uzima klasa čija je vjerojatnost najveća, ali samo ako je ta vjerojatnost 2 ili više puta veća od bilo koje druge vjerojatnosti klase. Ako nije, taj opisnik ne sudjeluje u konačnoj klasifikaciji. Od onih koji ulaze u konačnu klasifikaciju se odabire najčešća klasa.
4. Za svaku klasu se zbrajaju sve vjerojatnosti po svim opisnicima i klasa koja ima najveću vrijednost se uzima kao konačna klasa.

Kao glavna metoda je korištena posljednja jer je empirijski utvrđeno da daje puno bolje rezultate od ostalih metoda. Jedan od razloga je što pojava jednakih vjerojatnosti za različite klase kod pojedinog opisnika ne utječu na njezin konačni rezultat. Na još problema se nailazi kod druge metode koja ne radi dobro za veći broj klasa jer su tada sve vjerojatnosti za klase male. Metoda zbog toga odbacuje mnogo opisnika te ih vrlo malo sudjeluje u konačnoj odluci, što dovodi do lošijih rezultata.

Izlaz: Oznaka klase

3. Ispitivanje rješenja

3.1 Ispitna baza

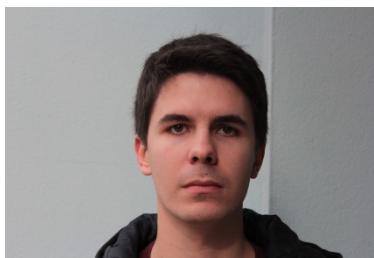
Prilikom testiranja korištene su dva skupa slika na kojima je ispitan postupak podudaranja. Prvi skup se sastoji od slika koje sadrže 34 različite građevine koje prikazuju svjetske znamenitosti kao što su Eiffelov toranj, Brandenburška vrata i sl. Taj je skup dobiven pretraživanjem Interneta.

Pregledno je to prikazano na slici 10.



Slika 10 prikaz dijela ispitnog skupa za građevine

Drugi skup se sastoji od slika koje sadrže 30 različitih osoba kao što je pregledno prikazano na slici 11, koje su nastale kao skup slika na projektu kolegija Računalni vid. Varijacija među slikama iste osobe je malena, jer su slike nastale u istim uvjetima osvjetljenja.



Slika 11 prikaz dijela ispitnog skupa za lica

Prilikom testiranja određen broj slika se upotrijebio za učenje klasifikatora, a preostali broj za testiranje. Testiranje se provodilo nad istim parametrima u više iteracija,

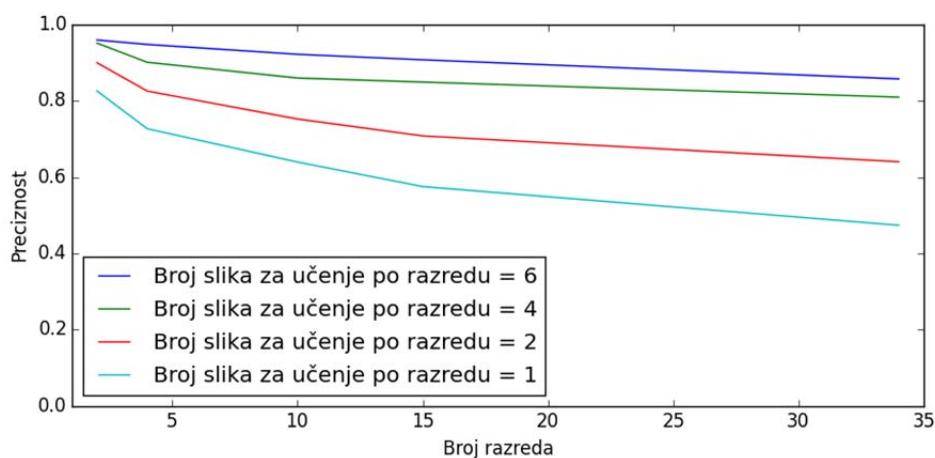
prilikom čega su svaki puta odabранe različite slike za učenje i testiranje kako bi se dobila prosječna uspješnost nad bazom. Za klasifikaciju je korišten klasifikator K-najbližih susjeda.

3.2 Rezultati učenja i ispitivanja

3.2.1 Slike urbane arhitekture

Radi usporedbe, na slici 12 prikazan je rad osmišljenog postupka našeg algoritma, i njegova preciznost, nad testnim slikama uz mijenjane parametre broja slika za učenje.

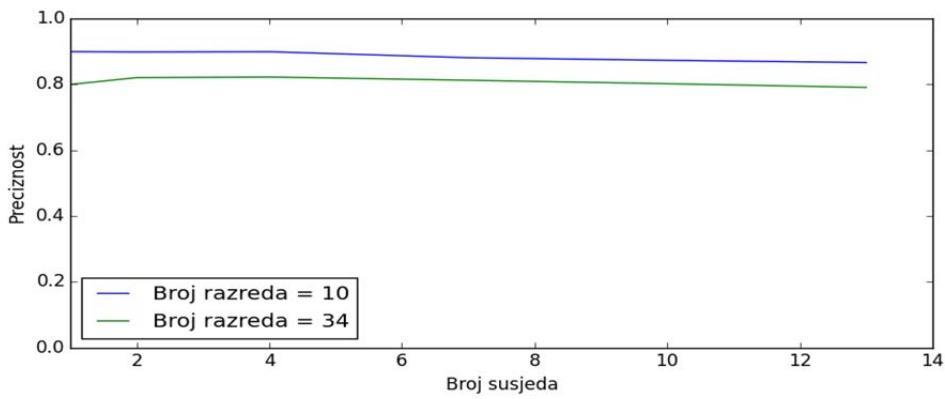
Slike prikazuju kako se ponaša preciznost kod osmišljenog postupka prilikom odabira različite količine slika za učenje po razredu. Za ispitivanje su odbrane sve slike koje nisu odabrane za učenje.



Slika 12 preciznost osmišljenog algoritma nad ispitnim skupom slika građevina

Na slici 12, x-os označava broj različitih razreda koji su nasumično odabrani, a y-os označava preciznost po provedenom testiranju. Testiranje je napravljeno u više iteracija, po principu K-fold, u više od 50 iteracija. Postoje četiri krivulje u ovisnosti o količini slika koje su odabrane za učenje; broj vezan uz krivulju označava broj slika odabranih za učenje.

Također, pokušali smo vidjeti hoće li naš algoritam biti robustan u ovisnosti o broju odabranih k-susjeda u K-NN algoritmu. To je prikazano na slici 13:



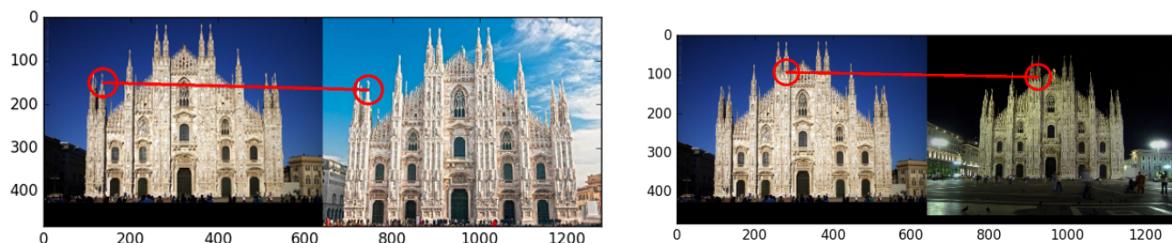
Slika 13 preciznost osmišljenog algoritma nad ispitnim skupom slika građevina

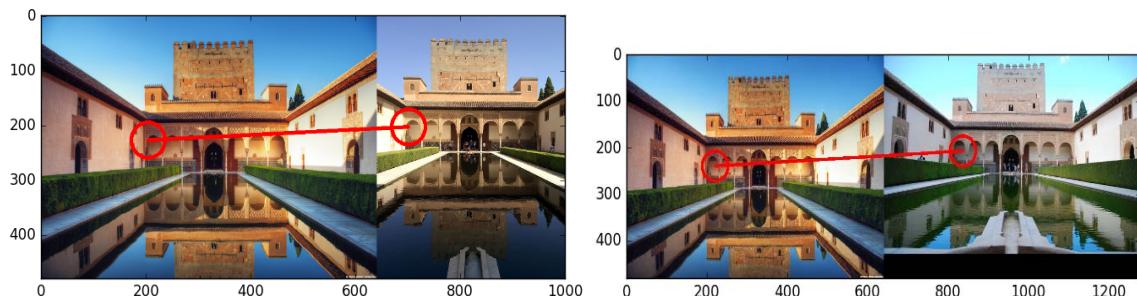
Os x prikazuje broj susjeda kao parametar K-NN-a, dok y os prikazuje preciznost za pojedini ispitni primjer. Kao i u prethodnom ispitivanju, odabrano je nekoliko iteracija, odnosno testirano je nekoliko puta s istim parametrima, ali različitim razredima i slikama za učenje, po principu K-fold, te je izračunata srednja vrijednost. Ispitano je za slučaj 10 i 34 razreda. Za treniranje su odabrane 4 slike, a na ostatku slika se provodilo ispitivanje.

3.2.2 Slučajevi podudaranja testnih slika urbane arhitekture

Sljedeće slike prikazuju rezultate podudaranja značajki prilikom ispitivanja nad slikama građevina.

1. Slučaj – dobro podudaranje:

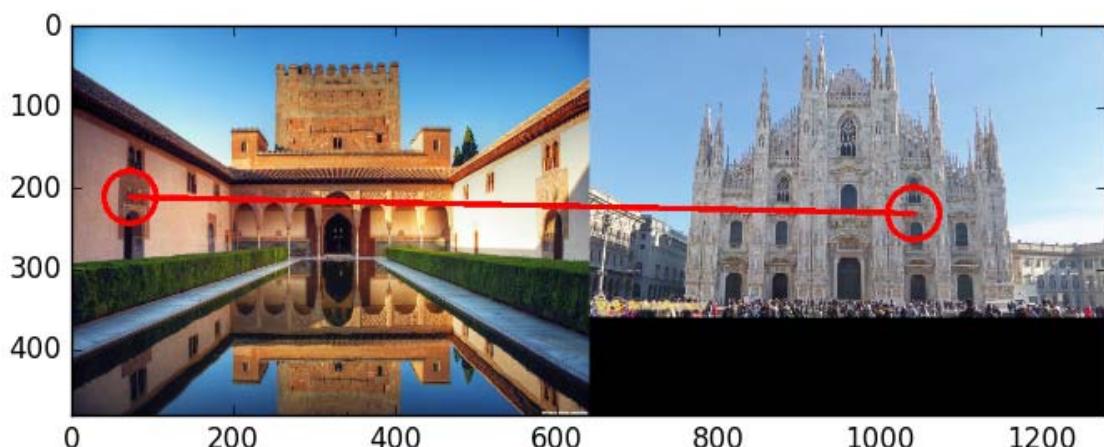




Slika 14 Dobro podudaranje

Na slici 14 vidljiva su 4 slučaja uspješnog podudaranja značajki i razreda građevina. Za sve slučajeve, algoritam je pronašao podudarne značajke.

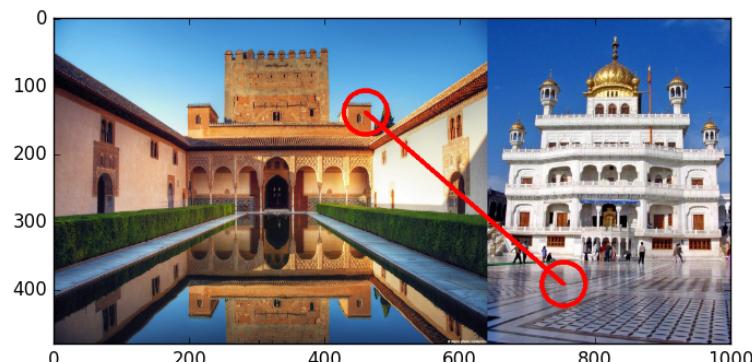
2. Slučaj, algoritam nije našao iste klase građevina, ali je razlika u opisnicima slika klasa za učenje i testne slike veoma mala:



Slika 15 Loše podudaranje, slični opisnici

Kao što je vidljivo na slici 15, algoritam je zbog malih razlika u opisnicima odlučio da su slike podudarne.

3. Slučaj, podudaranje slika je loše te nema primjetne sličnosti na slici (slika 16)

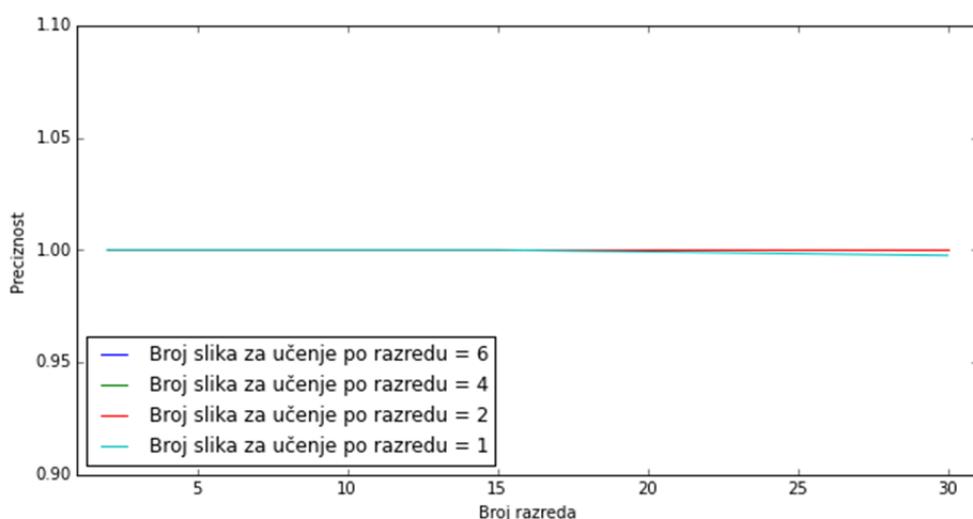


Slika 16 Loše podudaranje slika

3.2.3 Slike lica

Kao i nad građevinama, nad licima je provedeno jednako ispitivanje. Razlika između klasifikacije građevina i klasifikacije lica, sastoji se u koraku koji pomoću kaskadnog klasifikatora koji koristi Haar značajke pronalazi površinu lica (Viola – Jones algoritam – [17]), te nju koristi kao sliku za obradu. Time izoliramo samo lice te ne gledamo pozadinu koja može prouzročiti dodatni šum. Naučeni klasifikator je preuzet iz OpenCV biblioteke[18].

Ispitivanje je potvrdilo slabu varijabilnost slika u bazi, ali i dobar odabir značajki koje se mogu primjeniti na sličnu bazu, odnosno za klasifikaciju osoba, kao što i prikazuje grafikon na slici 17.

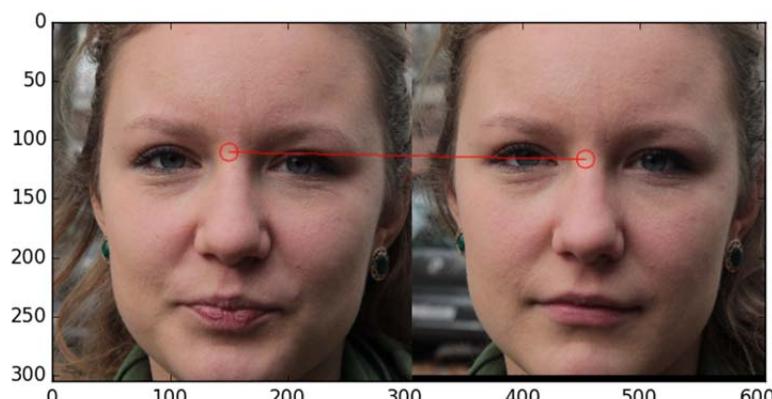


Slika 17 preciznost osmišljenog algoritma nad ispitnim skupom slika lica

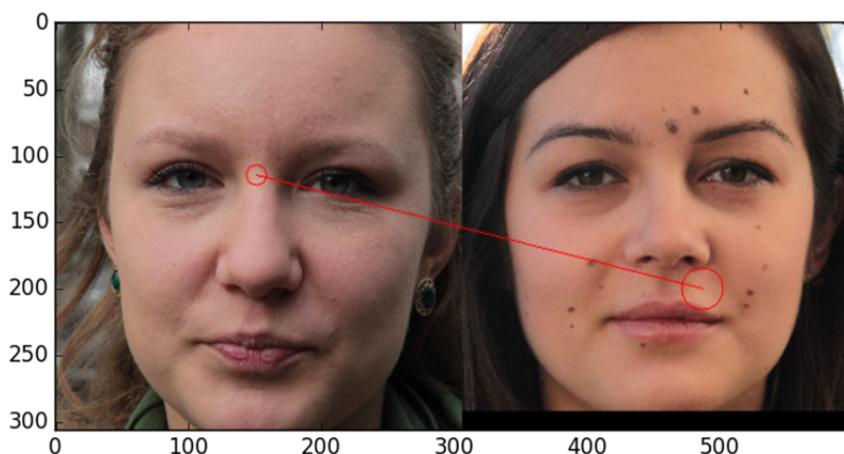
U nastavku su prikazane slike koje prikazuju neke značajke korištene pri klasifikaciji. Slike 18 i 19 prikazuju značajke koje odabiru dijelove slike koje su prema metodi simetrične. Također, na slici 20 je prikazana greška koja dvije različite značajke smatra sličnima.



Slika 18 Dobro podudaranje



Slika 19 Dobro podudaranje



Slika 20 Loše podudaranje

3.3 Analiza rezultata

3.3.1 Usporedba rezultata sa SIFT značajkama

Metodu smo usporedili sa postojećom metodom koja koristi SIFT (Scale-invariant feature transform) značajke. SIFT smo testirali na obje baze na jednak način, te usporedili rezultate.

Rezultati prikazuju da SIFT ima bolje rezultate. U svim mjeranjima rezultati SIFT-a su bili bolji, no razlika nije velika.

U sljedećoj tablici, tablici 1, pregledno su prikazani rezultati testiranja postupka podudaranja za slike urbane arhitekture:

broj klase	broj slika za učenje	broj iteracija	preciznost
34	6	10	0.86
15	6	100	0.91
10	6	100	0.92
4	6	100	0.95
34	4	10	0.81
15	4	5	0.85
10	4	10	0.86
4	4	64	0.90
2	4	50	0.95
34	2	32	0.64
15	2	50	0.71
10	2	50	0.75
4	2	50	0.83
2	2	50	0.90
35	1	19	0.47
15	1	79	0.58
10	1	100	0.64
4	1	100	0.73
2	1	100	0.83

Tablica 1 Histogram gradijenata + lokalne simetrije + K-NN - građevine

U tablici 2, prikazani su rezultati dobiveni korištenjem algoritma SIFT:

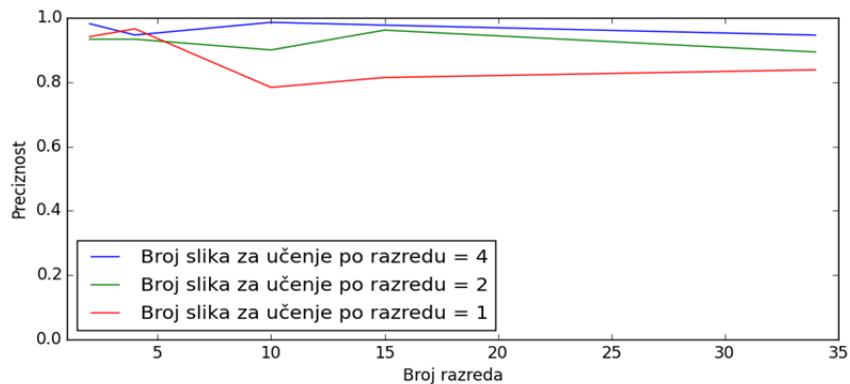
broj klase	broj slika za učenje	broj iteracija	preciznost
34	4	7	0.95
15	4	8	0.98
10	4	1	0.99
4	4	5	0.95
2	4	5	0.98

34	2	10	0.89
15	2	1	0.96
10	2	1	0.90
4	2	10	0.93
2	2	10	0.93
34	1	10	0.84
15	1	1	0.81
10	1	2	0.79
4	1	10	0.97
2	1	10	0.94

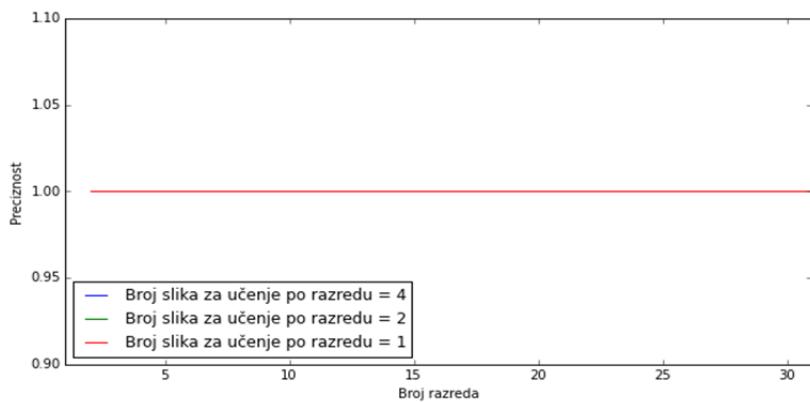
Tablica 2 SIFT - građevine

Iz rezultata je vidljivo da je podudaranje dobiveno algoritmom SIFT u više slučajeva bolje od korištenja lokalne simetrije za izračun opisnika koji se kasnije koriste u algoritmu K-NN, što je i očekivano s obzirom da postoji manji broj lokalnih simetrija u slikama koji se može iskoristiti za traženje podudarnih točaka.

Pregledno, grafikon algoritma SIFT provedenog nad slikama urbane arhitekture može se vidjeti na slici 21, i analogno, grafikon algoritma SIFT sprovedenog nad slikama ljudskih lica prikazan je na slici 22



Slika 21 SIFT algoritam - građevine



Slika 22 SIFT algoritam - lica

3.3 Ostali testirani pokušaji konstrukcije opisnika za postupak podudaranja korištenjem lokalnih simetrija

Prethodno opisani postupak je nastao iscrpnim proučavanjem literature[1][2][5][6], ali i nekoliko prethodnih pokušaja u kojima su razvijene slične metode i uočene neke korisne pravilnosti koje su kasnije bile iskorištene u našoj konačnoj verziji rada.

Svim pokušanim algoritmima je zajedničko to što obojanu sliku transformiraju u sivu sliku, smanje ju na veličinu 640x480 slikovnih elemenata te iskoriste Gaussovo zamućenje s parametrom $\sigma = 0.8$. Pomoću slike grade matricu ocjena simetrije za svaki slikovni element iz koje se kasnije različitim metodama izlučuju slikovni elementi koji će predstavljati značajke.

Ocjene se grade pomoću simetričnog prozora oko slikovnog elementa. Isprobani su razni pokušaji izrade opisnika značajki, a najboljom se pokazao ranije opisan postupak korišten u finalnom postupku.

3.3.1 Prvi pokušaj - Konstrukcija 128-bitnog vektora

Inicijalno se razmatrala metoda koja bi koristila intenzitete slikovnih elemenata u sivoj slici. Za svaku širinu prozora ($2 \cdot w$) kao potenciju broja 2 (u intervalu od $w=8$ do $w=128$) kreirala se matrica ocjena simetričnosti za svaki pojedini slikovni element.

Ocjena simetričnosti vršila se tako što bi se za svaki slikovni element u prozoru s lijeve strane izračunala apsolutna razlika od slikovnog elementa s desne strane na istoj udaljenosti od središta (dakle, bilateralno preslikanog).

Dobiveni vektor apsolutnih razlika (dimenzije w) transformira se u opisnik tako što se prvo izračuna prosječna vrijednost vektora te se sve vrijednosti manje od prosječne vrijednosti postave na nulu, a ostale na jedan. Tako je dobiven binarni vektor koji još valja podesiti da bude invarijantan na širinu prozora (w).

Po uzoru na SIFT opisnik [6] koristi se 128 bitni deskriptor koji se kreira skaliranjem vektora širine w za faktor $128/w$, pri čemu je ostatak vektora popunjeno interpolacijom. Pokušane su različite metode interpolacije (interpolacija najbližim susjedom, linearna interpolacija).

Daljnji postupci klasificiranja ne razlikuju se previše od razvijene metode, pri čemu treba izdvojiti da se ovdje pokušalo različitim vektorima težina usmjeriti zaključivanje metode najbližih susjeda prema određenim indeksima deskriptora. Neke od ideja su bile sljedeće:

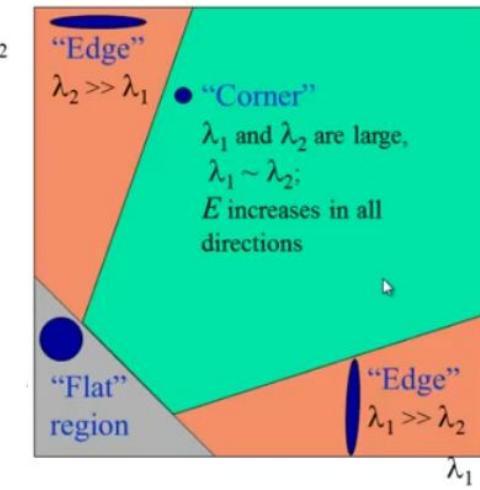
- Iskoristiti diskretizirani pozitivni dio Gaussove krivulje kako bi veću važnost predali točkama u prozoru bližima središtu simetrije. Pretpostavka je bila da su bliže točke relevantnije te da imaju manje šuma. Problem kod takvog pristupa jest što daljnje točke nemaju previše utjecaja na odluku klasifikatora
- Kako bi se spriječio prethodni problem pokušan je i obrnuti postupak koji je veće težine pridavao rubnim točkama, ali takav pristup je imao još veću pogrešku
- Idealnim se pokazao postupak u kojem točke na srednjoj udaljenosti od središta imaju najveću težinu, što je zaključeno i u radu [1].

Rezultati ovakvog sustava bili su nama nezadovoljavajući (uz 16 slika i 4 klase algoritam je polučivao prosječni uspjeh od **svega 30% pogodaka** (dok nasumični odabir klase bi imao 25% pogodaka).

3.3.2 Drugi pokušaj – korištenje Harrisovog detektora

Nezadovoljni uspjehom inicijalne ideje konstruirana je nova metoda. Harrisovim detektorom kuteva [10][11] odredi se matrica faktora koja za svaki slikovni element dodjeli faktor koji je veći što su veće šanse da se na tom slikovnom elementu nalazi vrh nekog kuta na slici.

Faktor za svaki slikovni element se računa izračunom svojstvenih vektora kovarijacijske matrice u okolini te točke. Ako su oba svojstvena vektora razmjerno velika takva točka se smatra kutem na slici, a ako je samo jedan onda se radi o rubu, kao što pokazuje slika 23.



Slika 23 proces odlučivanja kod Harrisovog detektora kutova

Prednost ovakve metode osim male složenosti je i robustnost, naime lokacije koje Harrisov detektor kuteva prepoznaće na jednoj slici u velikoj mjeri se pojavljuju lokacijama koje prepoznaće na drugoj slici.

Dalje je odabранo najboljih 10 000 potencijalnih kuteva na slici te od njih je uzeto onih 1 000 koji imaju najbolju ocjenu simetrije po postupku jednakom kronološki prethodnoj gornjoj metodi. Na jednak način se kreiraju i opisnici te se utvrđuje rezultat. Ovakvim pristupom smo dobili **prosječni uspjeh od 57% pogodaka** uz 20 slika i 4 klase, pri čemu su za učenje bile korištene 4 od 5 slika po klasi. Iako nije bilo testirano na istom setu slika, razlika je dovoljno velika kako bi se primijetio napredak.

3.3.3 Konačan odabir izgradnje opisnika

Ipak, ponukani željom da poboljšamo svoj algoritam ekstenzivno su analizirani problemi trenutnog rješenja te je primjećena velika mana istog. Naime, kut je bilateralno simetričan s obzirom na simetralu kuta, dakle kut polučuje horizontalnu simetriju ako i samo ako je simetrala u smjeru y osi. Trivijalno slijedi kako je udio takvih kuteva u odzivu Harrisovog detektora relativno mali.

Dalje valja zaključiti kako bi neka točka bila središte lokalne horizontalne bilateralne simetrije, njen x komponenta mora biti približna nuli, dok nam vertikalna

komponenta nije bitna. Kako bi izbjegli slučaj detekcije simetrije na izrazito monotonim regijama želimo da u promatranom prozoru sa svake strane postoji dovoljno gradijenata relativno velikih normi te x komponenata. Takvi gradijenti ukazuju na promjenu intenziteta u smjeru x osi, što nam odgovara.

4. Opis programske implementacije rješenja

Za programsku implementaciju postupka rješavanja zadatka upotrijebio se programski jezik Python verzije 2.7 s bibliotekama Numpy, Scipy, Matplotlib i Sklearn. Python je odabran zbog nekoliko prednosti; velike jednostavnosti korištenja pripadajućih biblioteka za strojno učenje (Scipy) i zbog jednostavnog prikaza rezultata osmišljenog rješenja (Matplotlib).

Ipak, u obzir treba uzeti činjenicu da je zbog velikog broja izračuna značajki i opisnika, velikog broja slika za učenje i testiranje, brzina izvođenja programske implementacije mala. U sadašnjoj implementaciji napravljena je paralelizacija procesa u razini OS-a s obzirom da Python ne podržava paralelizaciju dretvi. Za brže rezultate trebalo bi koristiti brže programske jezike poput C++-a ili C-a u nekim krucijalnim mjestima implementacije, poput faze preprocesiranja ili izračunavanja opisnika simetrija za svaku sliku.

Radi bolje organizacije, za rad u timu iskorišten je programski alat git kao VCS i platforma Github za hosting programskog koda cjelokupne implementacije zadatka.

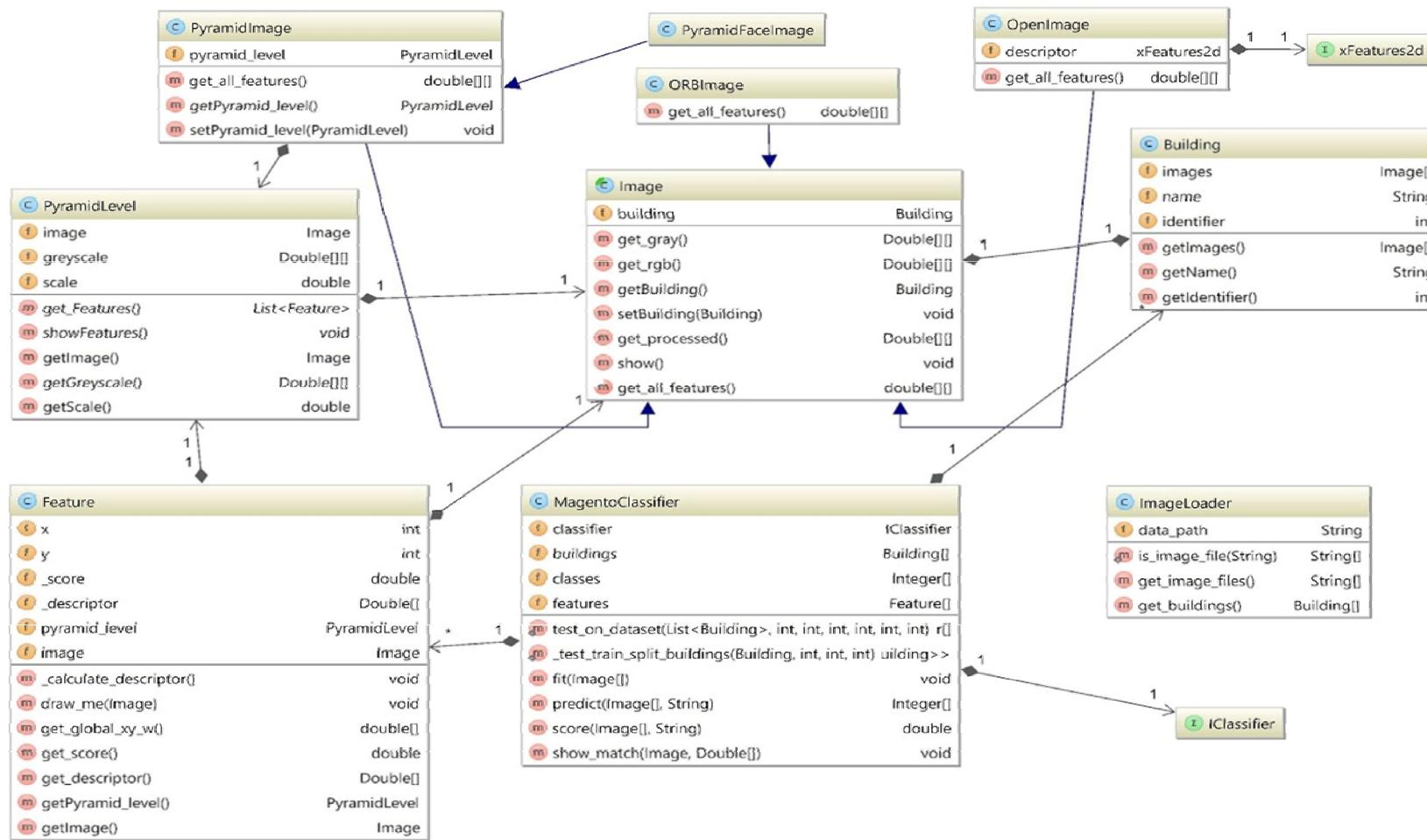
4.1 Korištenje implementacije

Korištenjem programskog jezika Python, izrađeno je nekoliko skripti koje se koriste za pokretanje programske implementacije. Radi lakšeg testiranja, unutar skripti postoje parametri koji se mogu mijenjati, a to su:

- Putanja do skupa slika za učenje
- Broj procesa koji se koriste prilikom izvršenja programa
- Razina ispisa (postoje četiri razine, najniža ispisuje samo rezultate, dok ostale označuju razinu ispisa ostalih informacija vezanih uz trenutnu izvedbu programa)
- Opcija kojem se može promijeniti prikazivanje značajka koje se podudaraju
- Broj različitih razreda građevina ili ljudskih obrazova koja se koriste prilikom izvedbe
- Broj slika za učenje po razredu
- Broj slika ispitivanje po razredu

4.2 Opis programskog rješenja

Program je izgrađen od nekoliko razreda kojima smo implementirali postupak. Radi jednostavnosti, razred koji ga implementira nazvan je Magento.



4.2.1 MagentoClassifier

MagentoClassifier sadrži metode koje omogućuju:

- Učenje instancirane jedinke MagentoClassifier sa skupom pripremljenih slika za učenje
- Testiranje sa skupom testnih slika urbane arhitekture ili ljudskih lica
- Ispitivanje naučene jedinke sa skupom slika; metoda vraća matricu pripadnosti za pojedine slike

4.2.2 ImageLoader

Ovaj razred se koristi za dohvatanje skupa slika koje će biti korištene prilikom daljnje obrade. Prilikom instanciranja se predaje putanja do datoteke koja sadrži željene slike. Razred ima dvije metode za pozvati:

- Metodu za dohvatanje svih putanja do pojedinih slika unutar predate putanje
- Metodu za dohvatanje svih slika zapakiranih u razred Building koji se nalaze unutar predate putanje

4.2.3 Feature

U ovom razredu se spremaju informacije za pojedinu značajku. Informacije su: položaj na slici, vrijednost opisnika, referenca na sliku kojoj pripada, dobrota značajke te razina piramide. Poželjno je imati razinu piramide jer je značajka specifično razvijena za postupak koji je razvijen u implementaciji, no moguće je koristiti ovaj razred uz bilo koji gotov algoritam za izradu značajki. Razred ima metode za dohvatanje pojedinih podataka o značajki te metodu za grafički prikaz.

4.2.4 Building

Razred Building se koristi za pohranu slika iste građevine u praktičniji format i sadrži jedinstvenu oznaku za taj skup slika, odnosno oznaku klase.

4.2.5 Image

Apstraktni razred koji ima metode za prikaz RGB te sive varijante slike. Slika se dohvaća iz putanje. Razredima koji proširuju razred Image preostaje da implementiraju metodu za dohvatanje značajki.

4.2.6 ORBImage

Proširenje razreda Image koje omogućuje dohvati svih značajki jedne slike. Značajke se računaju preko ORB sustava[19] za izvlačenje značajki.

4.2.7 OpenImage

Proširenje razreda Image koje omogućuje dohvati svih značajki jedne slike. Razred prima sustav za izvlačenje značajka. Jedini uvjet koji se postavlja na primljeni sustav je da ima metodu detectAndCompute, što imaju svi sustavi unutar OpenCV biblioteke.

4.2.8 PyramidImage

Proširenje razreda Image koje omogućuje dohvati svih značajki jedne slike. Razred ima nekoliko razina piramide i metodu za dohvati svih značajki sa svih razina piramide.

4.2.9 PyramidFaceImage

Proširena inačica razreda PyramidImage koja je posebno prilagođena za lice.

4.2.10 PyramidLevel

Razred koji predstavlja razinu piramide. Sadrži skaliranu sliku koja ju predstavlja te metode za dohvati podataka potrebnih u predobradi slike, metodu za izračun njezinih značajki te metodu za njihov grafički prikaz.

5. Zaključak

Podudaranje slika uporabom lokalne simetrije pokazao se kao nimalo jednostavan zadatak. Prilikom izrade našeg rješenja naišli smo na mnoge probleme. Neki od njih vezani su za kvalitetu analiziranih slika, na što, nažalost, ne možemo utjecati.

Također, u skupu slika lica koje smo koristili, slike su previše slične te nismo mogli ispitati robusnost razvijenog algoritma. Specifičniji problemi vezani za naš zadatak su bili pronalazak načina za izračun značajki i opisnika. Nakon nekoliko isprobanih rješenja opisanih u prethodnim poglavljima došlo se do konačnog algoritma koji vrlo uspješno izvršava podudaranje objekata.

Uspoređivanjem implementiranog algoritma i SIFT-a uočavamo da SIFT daje bolje rezultate. Ipak ne treba odbaciti ni dobiveni rezultat razvijene metode pogotovo imajući u vidu kako se koriste područja lokalne simetrije. Istovremeno u radu je pokazano kako tendenciju kvalitetnijih značajki imaju one koje imaju lošiju ocjenu simetrije.

Možemo isto tako zaključiti, da su uz veći broj slika za učenje rezultati još bolji. Promatrajući značajke za koje algoritam utvrđuje da se podudaraju ipak smo uočili neke pogreške.

Problem kod izvođenja algoritma je bila i brzina. Stvaranje piramide za svaku sliku te izračun značajki za svaku razinu i izračun pripadajućih opisnika iziskuje mnogo vremena i računalnih resursa. Implementaciju kritičnih dijelova bilo bi korisno prepisati u C/C++ ili iskoristiti grafičku karticu. Tada bismo zasigurno dobili znatno ubrzanje pa čak moguće i na razini klasifikacije u stvarnom vremenu.

Uz takav sustav dalje bismo mogli poboljšavati klasifikaciju iskorištavanjem slijednih slika. Konačan sustav bi onda vjerojatno bio sposoban sa zadovoljavajućom točnošću klasificirati snimljene građevine, ali i iskoristiti informaciju o njihovom položaju u prostoru za moguću konstrukciju trodimenzionalnog modela iste.

6. Literatura

- [1] Daniel Cabrini Hauagge, Noah Snavely: *Image Matching using Local Symmetry Features*, CVPR 2012: 206-213
- [2] Daniel Cabrini Hauagge, Noah Snavely: *Image Matching using Local Symmetry Features* , dataset <http://www.cs.cornell.edu/projects/symfeat/>, 19.1.2016.
- [3] Eli Shechtman, Michal Irani: *Matching Local Self-Similarities across Images and Videos*, CVPR'07
- [4] Zhenhua Wang, Bin Fan, Fuchao Wu: *Local Intensity Order Pattern for Feature Description*, ICCV '11, p603- p610
- [5] Richard Szeliski, *Computer Vision: Algorithms and Applications*, Springer-Verlag New York, Inc.
- [6] David G. Lowe. 2004. *Distinctive Image Features from Scale-Invariant Keypoints*. *Int. J. Comput. Vision* 60, 2 (November 2004), 91-110.
- [7] I. Sobel and G. Feldman. *A 3x3 isotropic gradient operator for image processing*. Never published but presented at a talk at the Stanford Artificial Project, 1968.
- [8] Sobelov operator:
<Http://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html?highlight=sobel>
(19.1.2016.)
- [9] Sobelov operator: http://opencv-python-tutorials.readthedocs.org/en/latest/py_tutorials/py_imgproc/py_gradients/py_gradients.html?highlight=sobel (19.1.2016.)
- [10] C. Harris and M. Stephens. *A combined corner and edge detection*. In Proceedings of The Fourth Alvey Vision Conference, pages 147–151, 1988
- [11] Harris edge detection: http://opencv-python-tutorials.readthedocs.org/en/latest/py_tutorials/py_feature2d/py_features_harris/py_features_harris.html , 19.1.2016.
- [12] Oxford dataset: (<http://www.robots.ox.ac.uk/~vgg/data/>) 19.1.2016
- [13] Gaussovsko zamučenje: http://opencv-python-tutorials.readthedocs.org/en/latest/py_tutorials/py_imgproc/py_filtering/py_filtering.html?highlight=gaussian%20blur 19.1.2016.

-
- [14] Gaussovska piramida:
<http://www.cs.toronto.edu/~kyros/courses/320/Lectures.2013s/lecture.2013s.09.pdf>, 19.1.2016.
- [15] K-NN:
<http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html> 19.1.2016.
- [16] Adaptivna ne-ekstremalna supresija:
<https://alliance.seas.upenn.edu/~cis581/wiki/Projects/project3Review.pdf>
19.1.2016.
- [17] Paul Viola and Michael J. Jones. 2004. Robust Real-Time Face Detection. Int. J. Comput. Vision 57, 2 (May 2004), 137-154.
- [18] Python implementacija Viola-Jones algoritma:
http://docs.opencv.org/master/d7/d8b/tutorial_py_face_detection.html
19.1.2016.
- [19] Python implementacija ORB sustava:
http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_feature2d/py_orb/py_orb.html 19.1.2016.