SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 209

POSTUPCI SINTEZE VOLUMNIH TEKSTURA I TEKSTURIRANJA OBJEKATA

Matija Forko

Zagreb, lipanj 2011.

Sadržaj

Popis slika

Popis Tablica

1. Uvod

Teksture se u računalnoj grafici koriste kao jednostavan i učinkovit način za odvajanje izgleda površine objekta od njegove geometrije. Dok se dvodimenzionalne teksture uobičajeno koriste za opisivanje vanjskog izgleda površine objekta, volumnim teksturama možemo opisati i izgled unutrašnjosti objekta.

Kod teksturiranja površine dvodimenzionalnom teksturom potrebno je pronaći parametrizaciju koja će povezati 3D koordinate površine objekta s 2D koordinatama teksture. Ovim postupkom, ovisno o kompleksnosti objekta, pojavljuju se veće ili male nepravilnosti kod prikaza. Kako bi se izbjegla potreba za traženjem parametrizacije, predložene su metode koje korištenjem informacija o geometriji objekta teksturu sintetiziraju direktno na površinu objekta. Iako se time izbjegava traženje parametrizacije, nedostatak ovog pristupa je što se jednom sintetizirana tekstura može koristiti samo na jednom objektu.

Upotrebom volumnih tekstura rješavamo oba navedena problema. Budući da je tekstura definirana u 3D koordinatama, nije potrebno tražiti parametrizaciju. Također, jednu volumnu teksturu možemo koristiti za teksturiranje različitih objekata. Upotreba volumnih tekstura ima prednosti u mnogim primjenama. Primjerice, zbog činjenice da volumne teksture pohranjuju podatke o unutrašnjosti objekta, omogućavaju vjerno simuliranje loma ili rezanja objekata. Nadalje, mnogi prirodni materijali, poput drva ili kamena, vjernije se prikazuju volumnim teksturama.

U ovom radu opisani su postupci sinteze dvodimenzionalnih tekstura te njihovo proširenje koje se koristi za sintezu volumnih tekstura. Također, opisan je i postupak teksturiranja objekata volumnim teksturama.

2. Teksture i sinteza tekstura

Kako bi dobili što vjerniji prikaz objekata na računalu potrebno je reproducirati brojne detalje površine objekta, npr. boju, refleksiju, prozirnost, izbočine. Jedan način za modeliranje detalja površine je korištenje poligona ili nekih drugih grafičkih primitiva. Jasno je da takav pristup nije praktičan ako želimo modelirati detaljnu površinu objekta. Drugi način je preslikavanje slike (teksture) na površinu objekta, odnosno teksturiranje.

Postupak teksturiranja možemo podijeliti u tri faze: fazu pribavljanja (dobivanja), preslikavanja i prikaza teksture. Budući da konačna kvaliteta prikaza najviše ovisi o kvaliteti pribavljene teksture, možemo reći da je upravo faza pribavljanja teksture najvažnija u cijelom postupku teksturiranja [1].

Teksture možemo dobiti na razne načine, primjerice s rukom crtanih slika ili fotografija. Iako rukom crtane slike mogu biti estetski zadovoljavajuće, njima je teško postići efekt fotorealizma. Nadalje, ne može svatko biti umjetnik i osmisliti dobru teksturu. Problem s fotografijama je što često nisu odgovarajuće veličine ili kvalitete, nisu jednoliko osvijetljene te nisu pogodne za direktno preslikavanje zbog pravilnog ponavljanja i mogućnosti pojavljivanja vidljivih šavova na spojevima.

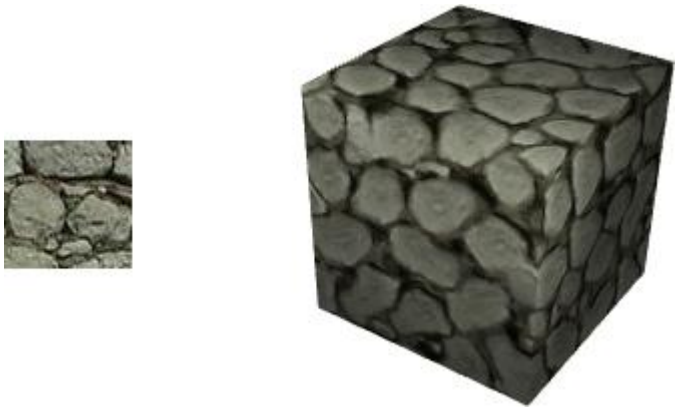
Postupak sinteze tekstura alternativni je način za dobivanje tekstura. Prednost postupaka sinteze tekstura nad prije spomenutim načinima dobivanja tekstura je jednostavnost upotrebe. Za dobivanje teksture korisnik mora podesiti malen skup parametara i zadati uzorak iz kojeg želi sintetizirati teksturu. Dobivena tekstura može biti proizvoljne veličine i bez vidljivih neprirodnih ponavljanja. Također, postupcima sinteze tekstura jednostavno se može dobiti ponavljajuća (engl. *tileable*) tekstura.

Zadatak postupka sinteze tekstura možemo opisati na sljedeći način. Iz zadanog uzorka potrebno je sintetizirati novu teksturu za koju možemo reći da je nastala korištenjem istog temeljnog postupka kao i uzorak. Primjer sintetizirane dvodimenzionalne teksture prikazan je na slici 1.

Slika 1. Sinteza dvodimenzionalne teksture. S lijeve strane je zadani uzorak, a s desne dobivena tekstura.

Budući da se dvodimenzionalne teksture sintetiziraju iz dvodimenzionalnih uzoraka, logičan zaključak je da se volumne teksture sintetiziraju iz trodimenzionalnih uzoraka. U praksi to nije slučaj zato što je do trodimenzionalnog uzorka teško doći. Na primjer, uzorak unutrašnjosti nekog materijala možemo dobiti rezanjem i fotografiranjem njegovih presjeka ili korištenjem CT (engl. *Computed Tomography*) skeniranja. Vidljivo je da te metode nisu baš praktične pa se zbog toga i volumne teksture sintetiziraju iz dvodimenzionalnih uzoraka.

Zadatak postupaka sinteze volumnih tekstura iz dvodimenzionalnih uzoraka formuliramo ovako. Potrebno je sintetizirati volumnu teksturu za koju možemo reći da je slična uzorku (odnosno da je nastala korištenjem istog temeljnog postupka) na svakom 2D presjeku teksture [2]. Primjer sintetizirane volumne teksture prikazan je na slici 2.



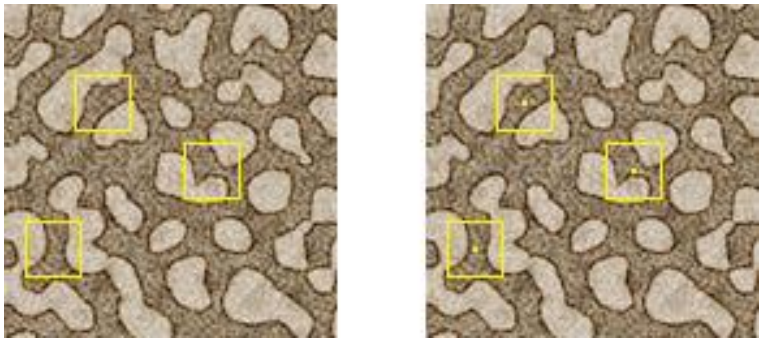
Slika 2. Sinteza volumne teksture. S lijeve strane je zadani uzorak, a s desne dobiven volumna tekstura.

2.1. Osnovni pojmovi

Prije opisivanja modela teksture pogodnog za sintezu teksture i osnovnih postupaka za sintezu dvodimenzionalnih tekstura, potrebno je objasniti neke osnovne pojmove potrebne za razumijevanje navedenih postupaka. Među najvažnije pojmove vezane za sintezu tekstura možemo uvrstiti pojam susjedstva i udaljenosti između dva susjedstva.

2.1.1. Susjedstva

U kontekstu postupaka za sintezu tekstura, pod pojmom susjedstvo smatramo skup slikovnih elemenata koji se nalaze u kvadratnom prozoru određene veličine. Susjedstvo možemo promatrati kao općenito (u skladu s prethodnom definicijom) ili kao susjedstvo nekog slikovnog elementa. Ako promatramo susjedstvo nekog slikovnog elementa, tada se taj slikovni element nalazi u sredini susjedstva (Slika 3).



Slika 3. Prikaz općenitih susjedstava i susjedstava slikovnih elemenata.

Iako je općenito susjedstvo jednako susjedstvu slikovnog elementa ako sadrže jednake slikovne elemente, ponekad je jednostavnije promatrati susjedstva ako ih ne povezujemo sa slikovnim elementom koji se nalazi u njihovom središtu.

2.1.2. Udaljenost između dva susjedstva

U postupcima sinteze tekstura često je potrebno nekom susjedstvu pronaći najbliže susjedstvo, odnosno najbližeg susjeda. Zbog toga je potrebno definirati što je udaljenost između dva susjedstva.

Ako je jedan slikovni element predstavljen s *k* komponenti boje (ili kanala), tada susjedstvo širine *w* možemo zapisati kao vektor dimenzionalnosti *n*.

$$n = w^2 \cdot k$$

Ako su $\boldsymbol{x}$ i $\boldsymbol{y}$ susjedstva zapisana u obliku vektora, njihovu udaljenost računamo kao euklidsku udaljenost ta dva vektora, odnosno kao L2 normu njihove razlike.

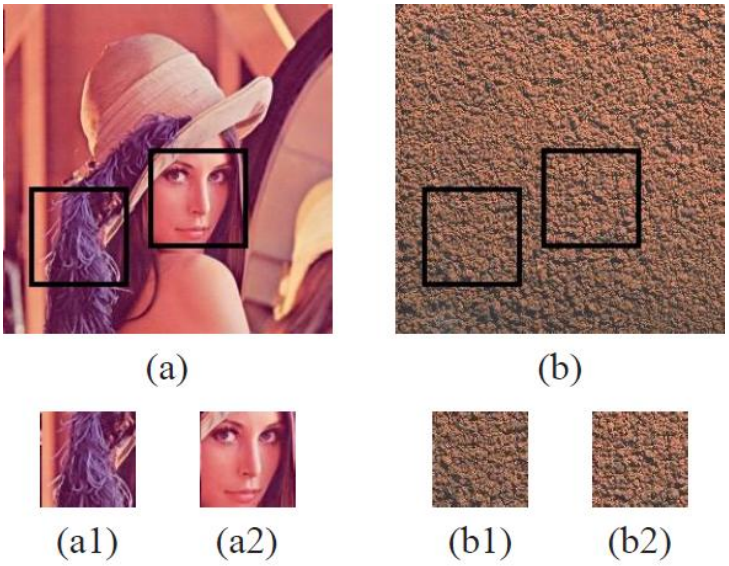
$$udaljenost = \sqrt{\sum_{i=1}^{n} |\boldsymbol{x_i} - \boldsymbol{y_i}|^2}$$

$$udaljenost = \|\boldsymbol{x} - \boldsymbol{y}\|$$

2.2. Matematički model teksture

Intuitivno je jasno da se ne može svaki uzorak iskoristiti za sintetiziranje teksture. Zbog toga je potrebno definirati model teksture te za sintezu koristiti one uzorke koji zadovoljavaju pretpostavke tog modela. Za većinu primjena u računalnoj grafici model temeljen na Markovljevim slučajnim poljima pokazao se kao najuspješniji [1].

Metode Markovljevih slučajnih polja modeliraju teksturu kao rezultat lokalnog i stacionarnog slučajnog procesa. Što znači da je svaki slikovni element teksture određen malim skupom slikovnih elemenata u svojoj okolini (susjedstvu) i da su okoline (susjedstva) svih slikovnih elemenata teksture međusobno slične. Ovo možemo objasniti na sljedeći način. Ako neku sliku promatramo kroz malen pomični prozor, kako se prozor pomiče vidljivi su različiti dijelovi slike. Ako su za neku veličinu prozora svi dijelovi slike koje vidimo kroz prozor slični, tada slika koju promatramo zadovoljava svojstvo stacionarnosti. Nadalje, ako možemo predvidjeti vrijednost nekog slikovnog elementa promatrajući samo njegovo susjedstvo, a ne i ostatak slike, tada ta slika zadovoljava svojstvo lokalnosti. Za neku sliku možemo reći da je tekstura ako zadovoljava svojstva stacionarnosti i lokalnosti. Slika 4 prikazuje razliku između općenite slike i teksture. Vidljivo je da su različiti dijelovi (b1) i (b2) teksture (b) vizualno slični, dok se dijelovi (a1) i (a2) općenite slike bitno razlikuju. Također, svaki slikovni element teksture (b) ovisan je samo o slikovnim elementima u svojoj okolini.



Slika 4. Usporedba općenite slike (a) i teksture (b).

Svi postupci sinteze tekstura temeljeni na modelu Markovljevih slučajnih polja sintetiziraju teksturu na takav način da za svako susjedstvo u sintetiziranoj teksturi možemo naći barem jedno slično susjedstvo u uzorku. Veličina susjedstva je parametar koji zadaje korisnik i trebala bi biti takva da susjedstva mogu obuhvatiti sva bitna svojstva teksture.

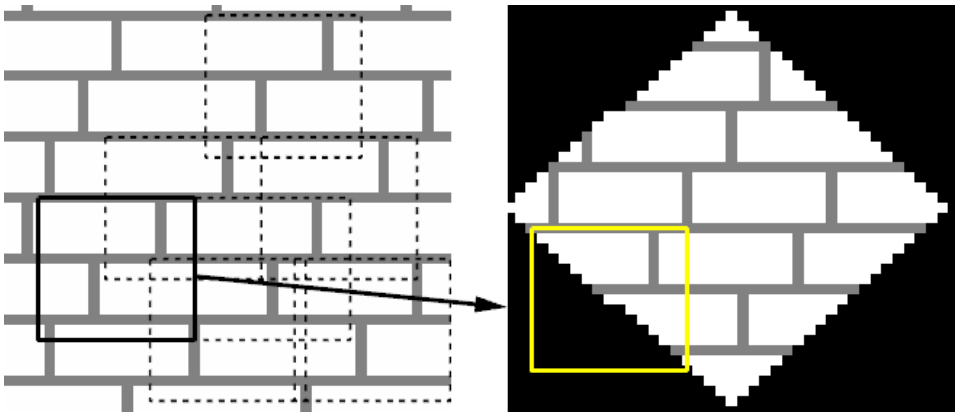
2.3. Osnovni postupci za sintezu tekstura

U nastavku će biti opisani osnovni postupci za sintezu dvodimenzionalnih tekstura koji su temeljeni na modelu Markovljevih slučajnih polja. Ovi osnovni postupci će kasnije biti prošireni i korišteni za sintezu volumnih tekstura.

2.3.1. Sinteza temeljena na slikovnim elementima (engl. *pixel-based synthesis*)

Zajedničko svim postupcima temeljenim na slikovnim elementima je da novu teksturu sintetiziraju jedan po jedan slikovni element. Jedan od prvih uspješnih postupaka za sintezu tekstura radi na takvom principu i opisan je u radu [3]. Ideja ovog postupka vrlo je jednostavna. Izlazna tekstura inicijalizira se kopiranjem nasumično odabranog prozora veličine 3x3 u sredinu teksture. Izlazna tekstura se zatim nadograđuje jedan po jedan slikovni element. Da bi se odredila vrijednost novog slikovnog elementa, potrebno je prvo pronaći sva susjedstva uzorka koja su slična susjedstvu tog slikovnog elementa (Slika 5). Zatim se slučajnim odabirom

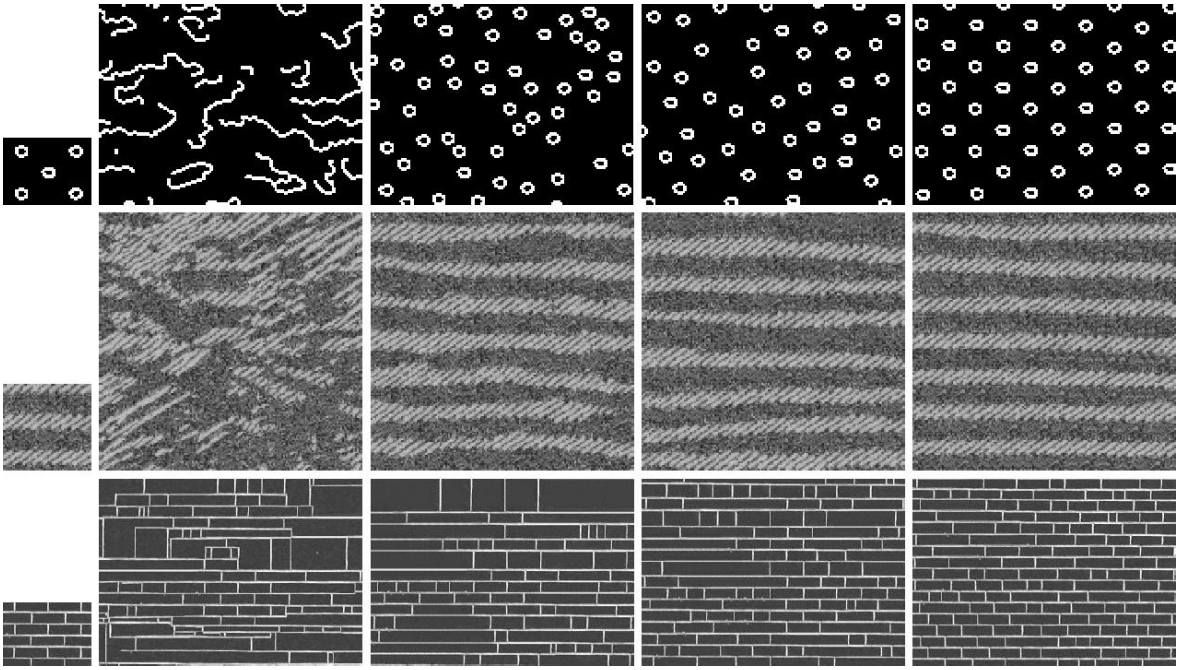
izabere jedno susjedstvo i za vrijednost slikovnog elementa uzima se vrijednost slikovnog elementa u sredini tog susjedstva. Mjera sličnosti dvaju susjedstava je suma kvadratnih udaljenost slikovnih elemenata pomnožena s dvodimenzionalnom Gaussovom jezgrom. Na taj način povećava se utjecaj slikovnih elemenata bližih centru susjedstva, a smanjuje utjecaj rubnih slikovnih elemenata. Kao što prikazuje slika 5, prilikom sintetiziranja teksture nisu poznate vrijednosti svih slikovnih elemenata u susjedstvu. Zbog toga se prilikom računanja udaljenosti dvaju susjedstava u obzir uzimaju samo poznate vrijednosti slikovnih elemenata.



Slika 5. Ilustracija algoritma. Desno je tekstura koje se sintetizira s označenim susjedstvom slikovnog elementa. Lijevo je uzorak s označenim sličnim susjedstvima.

Opisani algoritam jednostavno je shvatiti i implementirati, i ostvaruje dobre rezultate na različitim teksturama. Također, jednostavan je za upotrebu od strane korisnika zbog toga što je veličina prozora jedini parametar koji je potrebno zadati za rad algoritma. Ovisnost rezultata sinteze o veličini prozora prikazuje slika 6. Vidljivo je da dobivamo nasumične rezultate ako je veličina prozora premala te da pravilnost rezultata raste s veličinom prozora. Veliki nedostatak ovog algoritma, kao i ostalih algoritama temeljenih na slikovnim elementima, je vrijeme izvođenja. Pretraživanje svih susjedstava uzorka kako bi se našlo najbolje za svaki slikovni element oduzima puno vremena, i zbog toga sintetiziranje teksture može trajati od nekoliko sati pa do nekoliko dana, ovisno o veličini prozora i željene teksture. Prikaz vremena izvođenja postupka sinteze u ovisnosti o veličini uzorka, željenoj

veličini teksture, i veličini susjedstva slikovnog elementa (oznaka *N*) dan je u tablici 1 (na računalu s Intel Core2Duo procesorom na 2.26 GHz i 2GB radne memorije).



Slika 6. Rezultati sinteze tekstura opisanim algoritmom. Lijevo je uzorak, a desno rezultati sinteze s veličinama prozora 5x5, 11x11, 15x15, 23x23 [3].

Tablica 1. Trajanje postupka sinteze (format zapisa sati:minute:sekunde).

			Željena veličina teksture	
			128x128	**256x256**
Veličina uzorka	**64x64**	**N=5x5**	00:02:35	00:10:03
		N=11x11	00:09:21	00:35:55
		N=15x15	00:14:09	00:56:47
		N=23x23	00:22:06	01:33:15
	128x128	**N=5x5**	-	00:52:50
		N=11x11	-	02:58:04

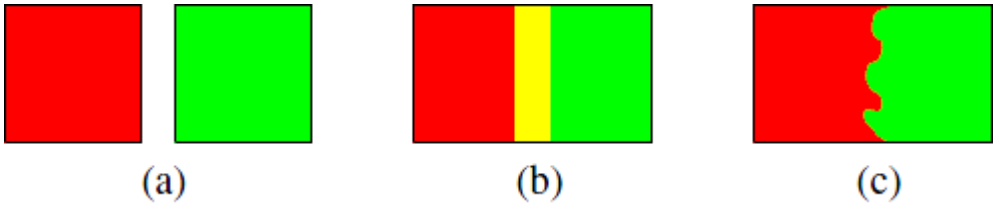
2.3.2. Sinteza temeljena na zakrpama (engl. *patch-based synthesis*)

Glavni problem postupaka sinteze temeljenih na slikovnim elementima je brzina izvođenja. Kvaliteta sinteze i brzina izvođenja može se povećati ako se umjesto kopiranja jednog po jednog slikovnog elementa izlazna tekstura slaže kombiniranjem većih dijelova uzorka – zakrpa ili blokova. Samim time što od jednom kopiramo više slikovnih elemenata, odnosno cijelo susjedstvo koje čini jednu zakrpu, povećavamo brzinu sinteze. Također, budući da bi slikovni elementi u susjedstvu koje kopiramo već trebali biti usklađeni jedni s drugima, za očekivati je da će se povećati i kvaliteta sintetizirane teksture, potrebno je samo osigurati dobro poklapanje zakrpi [1].

Postupci sinteze temeljeni na zakrpama mogu se promatrati kao proširenje postupaka temeljenih na slikovnim elementima, u smislu da umjesto slikovnih elemenata kopiramo zakrpe. Bitna razlika između tih postupaka je u načinu na koji jednu jedinicu sinteze (slikovni element ili zakrpu) kopiramo u izlaznu teksturu. Kod postupaka temeljenim na slikovnim elementima kopija je samo kopija, dok se kod postupaka temeljenim na zakrpama taj pojam malo komplicira. Budući da je jedna zakrpa veća od jednog slikovnog elementa, slaganjem više zakrpi dolazi do preklapanja više slikovnih elemenata.

Algoritmi koji koriste zakrpe kao jedinicu sinteze razlikuju se po načinu na koji određuju vrijednosti slikovnih elemenata koji se preklapaju. Najjednostavniji pristup je jednostavno ignoriranje prethodnih slikovnih elemenata i kopiranje nove zakrpe. Naravno da takav pristup dovodi do pojave vidljivih šavova na mjestima spajanja zakrpi. Primjer boljih pristupa tom problemu prikazuje slika 7. Na slici (a) prikazane su dvije zakrpe koje treba preklopiti. Na slici (b) nova vrijednost slikovnih elementa izračunata je kao aritmetička sredina vrijednosti slikovnih elemenata koji se preklapaju. Iako je bolji od prvog pristupa, ako su razlike među slikovnim elementima koji se preklapaju velike, ovaj pristup može uzrokovati pojavu zamućenja na područjima preklapanja zakrpi. Kao najbolji pokazao se pristup prikazan na slici (c). Kod ovog pristupa traži se put kroz područje preklapanja na kojem je udaljenost slikovnih elemenata najmanja, odnosno na kojem je greška

najmanja. Taj problem možemo riješiti dinamičkim programiranjem ili Dijkstrinim algoritmom. Dobiveni put predstavlja novu granicu između zakrpi.



Slika 7. Prikaz različitih pristupa spajanja zakrpi.

Kao što je i za očekivati, rezultati sinteze dobiveni na ovaj način brzinom i kvalitetom nadmašuju postupke bazirane na slikovnim elementima.

2.3.3. Sinteza tekstura optimizacijom teksture (engl. *texture optimization*)

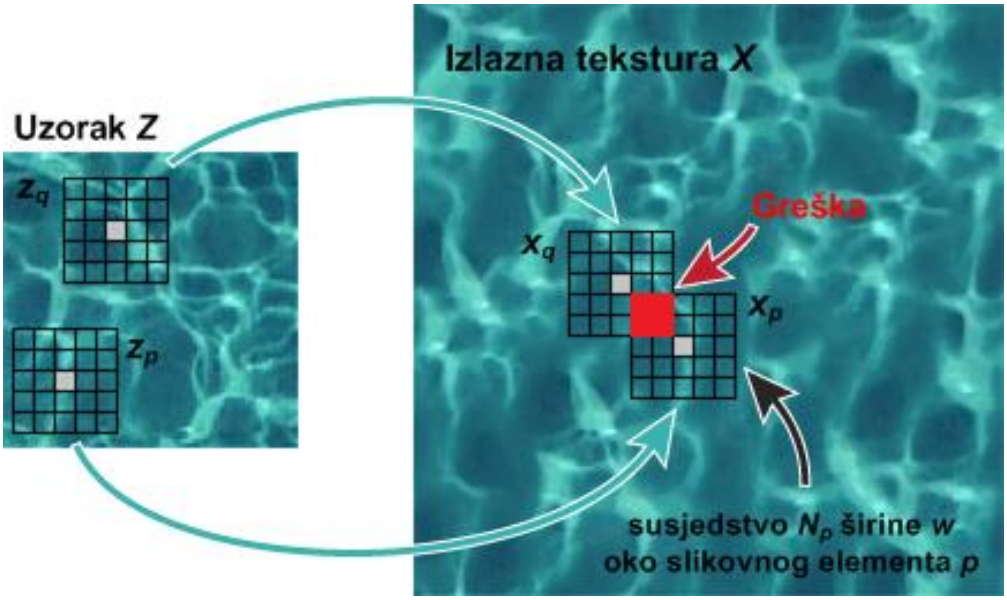
Postupak sinteze tekstura optimizacijom teksture prvi put prikazan je u članku [4]. Tekstura se sintetizira iterativnim postupkom optimizacije funkcije energije izlazne teksture. Energija jednog susjedstva izlazne teksture definira se kao kvadrat euklidske udaljenosti između tog susjedstva i njemu najbližeg susjedstva u danom uzorku. Energija teksture definira se kao suma energija svih pojedinačnih susjedstva teksture.

Formalno, neka je X izlazna tekstura, a Z zadani uzorak. Neka $\boldsymbol{x}$ označava vektoriziranu inačicu od X, dobivenu nadovezivanjem vrijednosti slikovnih elemenata. Neka za neku zadanu širinu susjedstva N_p predstavlja susjedstvo iz X centrirano oko slikovnog elementa p. Tada $\boldsymbol{x_p}$ označava pod-vektor od $\boldsymbol{x}$ koji odgovara slikovnim elementima u N_p. Neka $\boldsymbol{z_p}$ označava vektor susjedstva iz Z koje je najbliže $\boldsymbol{x_p}$ po euklidskoj normi. Tada prema [4], definiramo energiju teksture X kao:

$$E_t(\boldsymbol{x}; \{\boldsymbol{z_p}\}) = \sum_{p \in X'} \|\boldsymbol{x_p} - \boldsymbol{z_p}\|^2$$

Ilustracija opisanih oznaka prikazana je na slici 8. Pokazalo se da je nepotrebno i računski skupo tražiti najbliža susjedstva u Z za sva susjedstva u X. Zbog toga se traže najbliža susjedstva u Z samo za ona susjedstva u X koja su međusobno

udaljena za četvrtinu širine susjedstva. *X'* sastoji se od slikovnih elemenata koji su centri tih susjedstava [4].

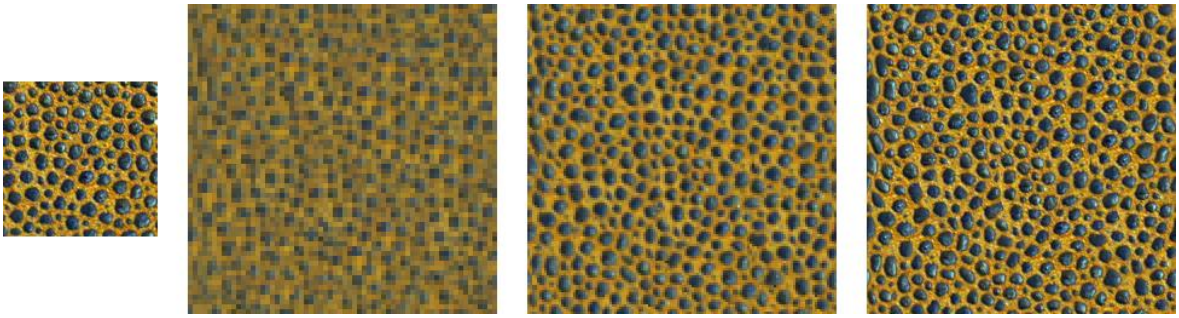


Slika 8. Ilustracija opisanog postupka.

Postupak sinteze teksture može se podijeliti u dvije faze koje se izmjenjuju iz iteracije u iteraciju - fazu pretraživanja i fazu optimizacije. U fazi pretraživanja za svako susjedstvo $\boldsymbol{x_p}$ u izlazu traži se odgovarajuće najbliže susjedstvo $\boldsymbol{z_p}$ u uzorku. U fazi optimizacije minimizira se dobivena funkcija energije, odnosno izračunavaju se vrijednosti slikovnih elemenata koji minimiziraju tu funkciju. Vidljivo je da ovakav postupak predstavlja mješavinu ranije opisanih postupaka baziranih na slikovnim elementima i zakrpama. U fazi pretraživanja prolazimo kroz teksturu susjedstvo po susjedstvo, što možemo poistovjetiti s postupcima baziranim na zakrpama, gdje zakrpa zapravo predstavlja susjedstvo. U fazi optimizacije izračunava se jedna po jedna vrijednost svakog slikovnog elementa što je karakteristika postupaka baziranim na slikovnim elementima.

Postupci sinteze teksture optimizacijom teksture obično su višerezolucijski. U početku se sintetizira tekstura manje rezolucije (govori se o tzv. *pixel resolution*, odnosno broju slikovnih elemenata), koja se kasnije interpolacijom povećava. Takva povećana slika koristi se kao inicijalizacija za sljedeću razinu. Tekstura manje rezolucije sintetizira se iz smanjenog uzorka. Smanjeni uzorci dobiveni su generiranjem Gaussove piramide iz originalnog uzorka. Ovaj postupak će biti

objašnjen kasnije. Tekstura se obično sintetizira u tri razine, prikazane na slici 9. Iz četiri puta smanjenog uzorka sintetizira se tekstura četiri puta manje rezolucije od zadane. Zatim se iz uzorka smanjenog dva puta sintetizira tekstura dva puta manje rezolucije. I konačno se iz originalnog uzorka sintetizira tekstura željene rezolucije. Rezultati sinteze za prve dvije razine koje prikazuje slika 9 povećani su korištenjem interpolacije najbližeg susjeda (engl. *nearest neighbour interpolation*) četiri, odnosno dva puta kako bi se lakše usporedili s konačnim rezultatom. Na svakoj razini mogu se koristiti različite veličine susjedstva. Ovakav postupak sinteze koristi se zato što se na početnim razinama, iz smanjenog uzorka, s manjom veličinom susjedstva može obuhvatiti više značajki teksture [4].



Slika 9. Sinteza tekstura optimizacijom teksture. S lijeve strane je zadani uzorak, a prema desno rezultati sinteze na kraju svake razine.

3. Sinteza volumnih tekstura iz dvodimenzionalnih uzoraka

Prethodno opisani postupci za sintezu dvodimenzionalnih tekstura mogu se proširiti i upotrijebiti za sintezu volumnih tekstura. U nastavku će biti opisan postupak sinteze volumnih tekstura, po uzoru na [5], dobiven proširenjem postupka sinteze optimizacijom teksture.

3.1. Pregled postupka

Ovaj postupak sinteze volumnih tekstura kombinira postupak optimizacije teksture s metodom podudaranja histograma. Cilj optimizacijskog procesa je minimiziranje funkcije globalne energije teksture koja predstavlja mjeru u kojoj se susjedstva sintetizirane volumne teksture razlikuju od susjedstva zadanog uzorka. Međutim, postoji opasnost da postupak zapne u lokalnom minimumu, npr. ponavljanjem jednakih susjedstva uzorka, te da ne iskoristi svu raznolikost uzorka. Za rješavanje tog problema koristi se metoda podudaranja histograma koja osigurava da globalne značajke sintetizirane volumne teksture odgovaraju zadanom uzorku. Podudaranje histograma povećava brzinu konvergencije postupka optimizacije, omogućava korištenje manjih susjedstva konstantne veličine prilikom sinteze (koriste se susjedstva veličine *8x8*), i zbog toga ubrzava cijeli postupak sinteze. Kod dvodimenzionalnih i volumnih tekstura sintetiziranih ovim postupkom vidljiva su poboljšanja u kvaliteti u odnosu na klasičan postupak sinteze optimizacijom teksture.

3.2. Optimizacija volumnih tekstura

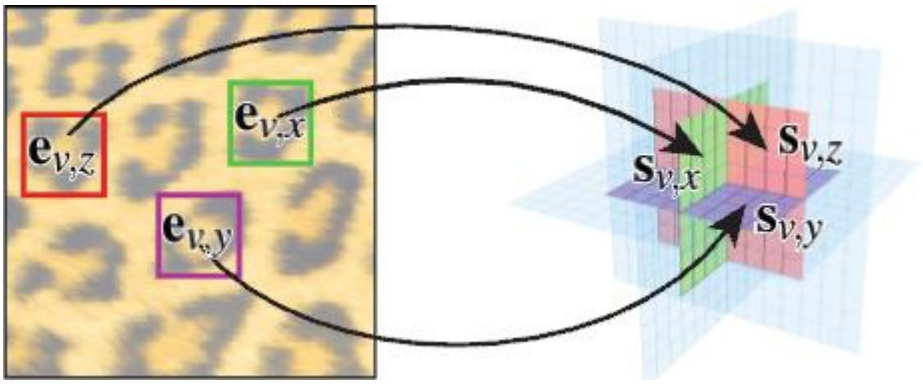
Na početku postupka optimizacije, potrebno je inicijalizirati volumnu teksturu tako da svakom elementu volumena (engl. *voxel*) dodijelimo vrijednost nasumično odabranog slikovnog elementa uzorka. Zatim se postupno povećava sličnost volumne teksture i uzorka iterativnim postupkom minimizacije funkcije energije koja je određena njihovom razlikom. Htjeli bi da svako susjedstvo na svakom 2D presjeku volumne teksture bude slično nekom susjedstvu uzorka. Budući da je računski skupo računati razlike na svim presjecima, razlike se računaju samo na tri presjeka koja su okomita na koordinate osi volumna teksture. Unatoč ovom

pojednostavljenju, dobivene volumne teksture slične su uzorku na proizvoljnim presjecima.

Ako s ***e*** (engl. *examplar*) označimo uzorak i sa ***s*** sintetiziranu volumnu teksturu, prema [5] energiju koju želimo minimizirati definiramo kao

$$E(\boldsymbol{s}, \{\boldsymbol{e}\}) = \sum_v \sum_{i \in \{x,y,z\}} \left\| \boldsymbol{s}_{v,i} - \boldsymbol{e}_{v,i} \right\|^r .$$

Gdje je $\boldsymbol{s}_v$ jedan element volumena, a $\boldsymbol{s}_{v,x}$, $\boldsymbol{s}_{v,y}$ i $\boldsymbol{s}_{v,z}$ su vektori susjedstva elementa *v* na presjecima koji su okomiti na osi *x*,*y* i *z*. Susjedstvo iz uzorka koje je najbliže susjedstvu $\boldsymbol{s}_{v,i}$ označeno je s $\boldsymbol{e}_{v,i}$ (Slika 10). Pokazalo se da je za eksponent *r = 0.8* postupak optimizacije otporniji na grube pogreške [4].



Slika 10. Prikaz tri susjedstva volumnog elementa i njima najbližih susjedstava uzorka.

Energiju teksture minimizira se iterativnim postupkom koji možemo podijeliti u dvije faze – pretraživanja i optimizacije. Ove dvije faze detaljnije će biti opisane u nastavku. Kao i postupak sinteze dvodimenzionalnih tekstura optimizacijom, i ovaj postupak je višerezolucijski, s tri razine sinteze. Smanjeni uzorci dobiveni su generiranjem Gaussove piramide, a na kraju svake razine volumnu teksturu povećavamo trilinearnom interpolacijom.

Treba napomenuti da se ovim postupkom mogu sintetizirati teksture iz uzorka proizvoljnog broja komponenti, tako da je moguće uz standardni uzorak s tri komponente boje zadati i mapu značajki koja naglašava bitne značajke uzorka te na taj način poboljšava rezultate sinteze.

3.2.1. Faza pretraživanja

U fazi pretraživanja za sva susjedstva $\boldsymbol{s_{v,i}}$ svih elemenata volumena *v* tražimo najbliže susjedstvo $\boldsymbol{e_{v,i}}$ uzorka. Najjednostavniji način za traženje najbližeg susjeda nekom susjedstvu je računanje udaljenosti od tog susjedstva do svakog susjedstva uzorka. Najveća mana ovakvog pretraživanja grubom silom je mala brzina pretraživanja.

Broj susjedstva u uzorku možemo izračunati po sljedećoj formuli.

$$broj\_susjedstva = (širina\_uzorka - širina\_susjedstva + 1)^2$$

Broj elemenata koje svako susjedstvo sadrži možemo izračunati po formuli.

$$broj\_elemenata = širina\_susjedstva^2 \cdot broj\_komponenti$$

Tako za uzorak širine i visine 128 slikovnih elemenata koji se sastoji od tri komponente boje, i za susjedstvo širine 8 dobivamo sljedeće brojke. Za svako susjedstvo u volumnoj teksturi potrebno je pretražiti 14641 susjedstvo u uzorku i svaki puta izračunati udaljenost između dva vektora od koji se sastoje od 192 elementa.

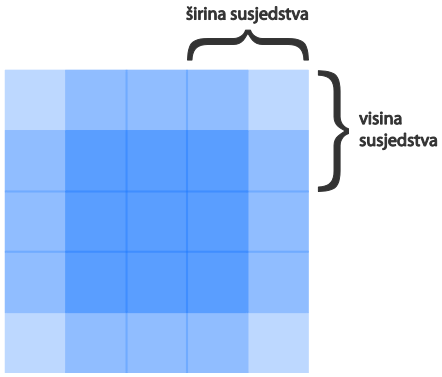
Vidljivo je da je upravo faza pretraživanja najsporiji dio postupka sinteze volumne teksture i zbog toga itekako ima smisla razmišljati o njenom ubrzanju. Ubrzanje možemo dobiti na tri načina:

1. Smanjenjem broja susjedsatva u uzorku koja ćemo pretražiti kod traženja najbližeg susjeda.
2. Smanjenjem dimenzionalnosti susjedstava.
3. Smanjenjem broja susjedstava volumne teksture za koja tražimo najbliže susjede.

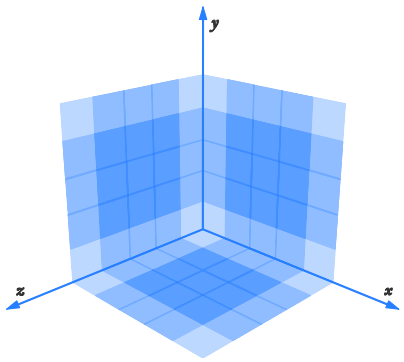
Broj susjedstava u uzorku koja ćemo pretražiti kod tražnja najbližeg susjeda možemo smanjiti korištenjem struktura podataka za pretraživanje umjesto pretraživanja grubom silom. Također, umjesto traženja pravog najbližeg susjeda, možemo dopustiti malu pogrešku i tražiti približno najboljeg susjeda (engl. *approximate nearest neighbour,* ANN).

Dimenzionalnost susjedstava možemo smanjiti postupkom analize glavnih komponenti (engl. *principal component analysis,* PCA). Ova dva načina ubrzanja biti će opisana u zasebnom odjeljku.

Treći način ubrzanja jednostavnije je objasniti ako promatramo općenita susjedstva umjesto susjedstva volumnih elemenata. Sva 2D susjedstva (Slika 11) volumne teksture nalaze se na presjecima koji su okomiti na koordinatne osi (Slika 12). Za volumnu teksturu veličine *128x128x128* takvih presjeka ima *3·128*. Odnosno, imamo 128 presjeka za svaku koordinatnu os. Svaki takav presjek možemo promatrati kao zasebnu dvodimenzionalnu teksturu. Budući da se susjedstva preklapaju, nije potrebno tražiti najbliže susjede za svako susjedstvo. Dovoljno je tražiti najbliže susjede za susjedstva koja su razmaknuta za četvrtinu širine susjedstva. Odnosno u našem slučaju, za fiksnu širinu susjedstva od osam slikovnih elemenata, tražimo najbliže susjede za susjedstva koja su razmaknuta za dva slikovna elementa. Ovime smo fazu pretraživanja ubrzali dva puta.



Slika 11. Prikaz 2D susjedstva na jednom presjeku. Na mjestima gdje se susjedstva preklapaju plava boja je većeg intenziteta.



Slika 12. Prikaz 2D susjedstva na različitim presjecima koji su okomiti na koordinatne osi.

3.2.2. Faza optimizacije

U fazi optimizacije određuje se nova vrijednost svakog elementa volumena na temelju prethodno pronađenih najbližih susjeda. Nove vrijednosti elemenata volumena moraju biti takve da minimiziraju prethodno navedenu funkciju energije. Budući da se susjedstva preklapaju, svaki element volumena će se u funkciji pojaviti više puta, jednom za svako susjedstvo u kojem se nalazi.

Kod traženja minimuma funkcije poznate su vrijednosti slikovnih elemenata koji pripadaju susjedstvima $\boldsymbol{e_{v,i}}$ uzorka i njih možemo tretirati kao konstante. Ako funkciju deriviramo po jednom elementu volumena čiju vrijednost tražimo i izjednačimo s nulom, možemo dobiti izraz po kojem se računaju vrijednosti elemenata volumena.

$$\boldsymbol{s_v} = \frac{\sum_{i\in\{x,y,z\}} \sum_{u\in N_i(v)} \boldsymbol{e_{u,i,v}}}{\sum_{i\in\{x,y,z\}} \sum_{u\in N_i(v)} 1}$$

Ovdje $N_i(v)$ označava susjedstvo elementa volumena v na presjeku koji je okomit na os koju označava i, dok $\boldsymbol{e_{u,i,v}}$ označava slikovni element uzorka u susjedstvu $\boldsymbol{e_{u,i}}$ koji odgovara elementu volumena v. Iz ovoga slijedi da je optimalna vrijednost elementa volumena jednaka prosjeku slikovnih elemenata iz različitih susjedstava uzorka koji odgovaraju elementu volumena v. Budući da smo promijenili vrijednosti elemenata volumena, u sljedećoj iteraciji će se najbliža susjedstva vjerojatno promijeniti. Česte promjene najbližih susjedstava dovode do velikih promjena vrijednosti elemenata volumena iz iteracije u iteraciju što za posljedicu ima sporu konvergenciju postupka optimizacije. Kako bi se smanjile promjene vrijednosti elemenata volumena i povećala brzina konvergencije, u fazi pretraživanja se svakom pronađenom najbližem susjedstvu dodjeljuje težina koja ovisi o udaljenosti. Točnije, svakom slikovnom elementu u susjedstvu dodjeljuje se izračunata težina susjedstva. Prema tome član funkcije energije možemo napisati na sljedeći način [5].

$$\left\|\boldsymbol{s_{v,i}} - \boldsymbol{e_{v,i}}\right\|^r = \left\|\boldsymbol{s_{v,i}} - \boldsymbol{e_{v,i}}\right\|^{r-2} \cdot \left\|\boldsymbol{s_{v,i}} - \boldsymbol{e_{v,i}}\right\|^2 = \boldsymbol{\omega_{v,i}} \cdot \left\|\boldsymbol{s_{v,i}} - \boldsymbol{e_{v,i}}\right\|^2$$

Sada minimiziramo sljedeću funkciju

$$E(\boldsymbol{s}, \{\boldsymbol{e}\}) = \sum_v \sum_{i\in\{x,y,z\}} \sum_{u\in N_i(v)} \boldsymbol{\omega_{v,i,u}} \cdot \left(\boldsymbol{s_{v,i,u}} - \boldsymbol{e_{v,i,u}}\right)^2.$$

Jednakim postupkom, deriviranjem po jednom elementu volumena i izjednačavanjem s nulom dolazimo do izraza za izračun vrijednosti jednog elementa volumena [5].

$$\boldsymbol{s_v} = \frac{\sum_{i\in\{x,y,z\}} \sum_{u\in N_i(v)} \boldsymbol{\omega_{u,i,v}} \cdot \boldsymbol{e_{u,i,v}}}{\sum_{i\in\{x,y,z\}} \sum_{u\in N_i(v)} \boldsymbol{\omega_{u,i,v}}}$$

Prema tome, optimalna vrijednost elementa volumena jednaka je ponderiranom prosjeku slikovnih elemenata iz različitih susjedstava uzorka koji odgovaraju elementu volumena *v*.

3.2.3. Uvjet zaustavljanja

Faza pretraživanja i faza optimizacije izmjenjuju se sve dok nije zadovoljen uvjet zaustavljanja. Kada je uvjet zadovoljen, postupak sinteze prelazi na sljedeću razinu, ili završava ako se nalazi na posljednjoj razini. Prelazak na sljedeću razinu podrazumijeva povećanje volumne teksture trilinearnom interpolacijom i nastavak sinteze s većim uzorkom iz prethodne razine Gaussove piramide.

Uvjet zaustavljanja formuliran je tako da postupak sinteze prelazi na sljedeću razinu kada je promjena volumne teksture dovoljno mala da možemo reći da je postupak optimizacije konvergirao. Konkretno, računa se prosječna promjena volumne teksture po elementu volumena. Uvjet je zadovoljen kada prosječna promjena padne ispod određene razine (kada je promjena vrijednosti po elementu volumena manja od 1; maksimalna promjena iznosi 255) ili kada je relativna promjena prosječne promjene manja od 1%.

3.3. Podudaranje histograma

Kao što je već spomenuto, metodom podudaranja histograma želimo smanjiti mogućnost da proces optimizacije zapne u lokalnom optimumu. Histogram prikazuje distribuciju vrijednosti komponenta boje slikovnih elemenata po pojedinim intervalima. Ako osiguramo podudaranje histograma uzorka i sintetizirane volumne teksture, tada će volumna tekstura biti sličnija uzorku, i svi dijelovi uzorka biti će prisutni u sintetiziranoj volumnoj teksturi.

Procesom traženja najbližih susjeda u fazi pretraživanja osiguravamo dobro lokalno poklapanje uzorka i sintetiziranje teksture. Praćenjem histograma u

postupak sinteze uvodimo i globalne značajke čijim poklapanjem povećavamo kvalitetu sintetizirane teksture [5].

Podudaranje postižemo na sljedeći način. Konstruiramo 16-stupčani histogram za svaku komponentu boje uzorka i volumne teksture. Zatim u fazi optimizacije, prilikom računanja nove vrijednosti elementa volumena, smanjujemo težinu svakog slikovnog elementa koji bi mogao doprinijeti povećanju razlike između histograma uzorka i histograma volumne teksture.

Formalno, neka $H_{s,j}$ i $H_{e,j}$ označavaju histograme uzorka i volumne teksture s indeksom *j*. Vrijednost stupca *b* (engl. *bin*) histograma *H* označena je s *H(b)*. Nadalje, neka $b_j(c)$ označava stupac koji sadrži boju *c* (engl. *color*). Novu težinu računamo kao [5]:

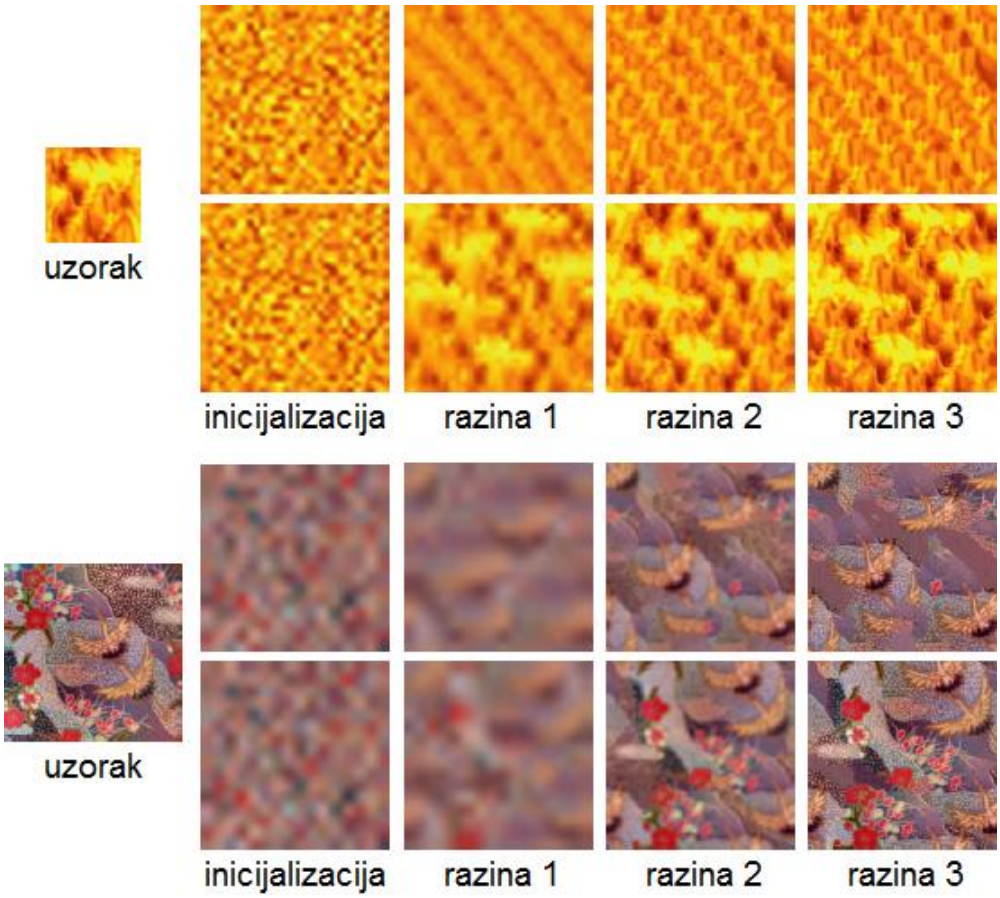
$$\omega'_{u,i,v} = \frac{\omega_{u,i,v}}{1 + \sum_{j=1}^{k} \max\left[0, H_{s,j}\left(b_j\left(\boldsymbol{e_{u,i,v}}\right)\right) - H_{e,j}\left(b_j\left(\boldsymbol{e_{u,i,v}}\right)\right)\right]}$$

Jednadžbu možemo protumačiti na sljedeći način. Svaki slikovni element $\boldsymbol{e_{u,i,v}}$ svojom težinom više ili manje „vuče“ novu vrijednost elementa volumena prema stupcu histograma u kojem se nalazi vrijednost tog slikovnog elementa. To je poželjno ako taj stupac u histogramu volumne teksture ima manju vrijednost nego u histogramu uzorka ($H_{s,j}(b_j(\boldsymbol{e_{u,i,v}})) < H_{e,j}(b_j(\boldsymbol{e_{u,i,v}}))$) jer se tada povećanjem vrijednost tog stupca histogram volumne teksture približava histogramu uzorka. U protivnom, ako je $H_{s,j}(b_j(\boldsymbol{e_{u,i,v}})) > H_{e,j}(b_j(\boldsymbol{e_{u,i,v}}))$, povećanjem vrijednosti tog stupca razlika između histograma bi se povećala. U tom slučaju smanjujemo težinu dodijeljenu slikovnom elementu $\boldsymbol{e_{u,i,v}}$ kako bi imao manji utjecaj na konačan rezultat.

Histogram volumne teksture mora se osvježavati kod svake promjene vrijednosti elementa volumena. U protivnom, kada bi se osvježavao jednom na kraju svake iteracije, moglo bi doći do grešaka i neželjenih rezultata. Također, nove vrijednosti elemenata volumena potrebno je računati nasumičnim redoslijedom kako bi se izbjegao neželjeni utjecaj pravilnog obilaska na kvalitetu rezultata.

Usporedba rezultata sinteze dvodimenzionalnih tekstura s i bez korištenja metode podudaranja histograma prikazana je na slici 13. Prvi i treći red prikazuju

rezultate sinteze bez korištenja metode podudaranja histograma. Vidljivo je da prilikom sinteze nisu iskorištena sva svojstva uzorka te da se rezultat ne može usporediti s danim uzorkom. Drugi i četvrti red prikazuju bolje rezultate sinteze koji su dobiveni korištenjem metode praćenja histograma.



Slika 13. Usporedba rezultata sinteze sa i bez korištenja metode praćenja histograma. S lijeve strane je uzorak, a prema desno redom nasumična inicijalizacija i rezultati sinteze na kraju svake razine [5].

3.4. Ubrzavanje izvođenja

U odjeljku o fazi pretraživanja spomenuta su tri načina za ubrzavanje pretraživanja. Jedan način objašnjen je u tom odjeljku, dok će preostala dva biti objašnjena u nastavku ovog odjeljka.

3.4.1. Ubrzavanje izvođenja smanjenjem dimenzionalnosti susjedstava

Dimenzionalnost susjedstava možemo smanjiti postupkom analize glavnih komponenti (engl. *principal component analysis,* PCA). Analizom glavnih komponenti vektore susjedstava projiciramo u prostor manje dimenzionalnosti na način koji maksimizira varijancu vektora smanjenje dimenzionalnosti [6].

Da bi proveli analizu glavnih komponenti, potrebno je izračunati srednju vrijednost i kovarijacijsku matricu vektora susjedstava. Srednju vrijednost računamo kao:

$$\boldsymbol{m} = \frac{1}{N}\sum_{k=1}^{N} \boldsymbol{e_k}$$

Gdje je $\boldsymbol{m}$ vektor srednje vrijednosti, *N* ukupan broj susjedstva i $\boldsymbol{e_k}$ vektor susjedstva s indeksom *k*.

Kovarijacijsku matricu računamo kao:

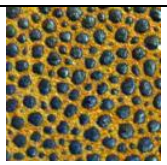
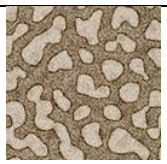
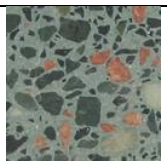
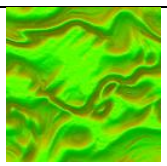
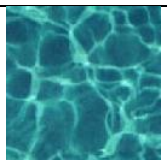
$$\boldsymbol{K} = \frac{1}{N-1}\sum_{k=1}^{N} (\boldsymbol{e_k} - \boldsymbol{m}) \cdot (\boldsymbol{e_k} - \boldsymbol{m})^{\boldsymbol{\tau}}$$

Kako bi dobili projekciju vektora susjedstava u prostor manje dimenzionalnosti, potrebno je odrediti transformacijsku matricu. Transformacijska matrica kojom radimo projekciju *r* dimenzionalnog prostora u *n* dimenzionalni prostor, gdje je $n < r$, određena je svojstvenim vektorima koji odgovaraju *n* najvećih svojstvenih vrijednosti kovarijacijske matrice $\boldsymbol{K}$. Želimo odabrati takav *n* da sačuvamo 95% varijance uzorka. Za taj zadatak potrebno je izračunati sumu svojstvenih vrijednosti i zatim odrediti koliko je najvećih svojstvenih vrijednosti potrebno da njihova kumulativna suma bude veća ili jednaka 95% ukupne sume. Dobiveni broj svojstvenih vrijednosti je traženi *n* [7].

Projekcija u *n* dimenzionalni prostor dobiva se množenjem matrice koja sadrži sve vektore susjedstava uzorka, kojima smo prethodno oduzeli srednju vrijednost (matrica je dimenzija *N x r*), s transformacijskom matricom (dimenzija *r x n*). Rezultat je matrica dimenzija *N x n* koja sadrži vektore susjedstava smanjene dimenzionalnosti.

Ovisno o uzorku, dimenzionalnost susjedstva smanjuje se sa 192 elementa (za uzorke s 3 komponente boje) na 5 do 50 elemenata. Budući da je sačuvano 95% varijance uzorka, možemo reći smanjenjem dimenzionalnosti nismo izgubili na kvaliteti sinteze. Tablica 2 prikazuje dimenzionalnost susjedstava uzoraka nakon provođenja postupka analize glavnih komponenti na pojedinim razinama postupka sinteze.

Tablica 2. Dimenzionalnost susjedstava uzoraka na pojedinim razinama postupka sinteze.

	1. razina	2. razina	3. razina
	55	43	46
	27	30	36
	43	32	29
	37	25	19
	26	18	12
	10	7	5

Prikaz vremena izvođenja postupka sinteze za različite veličine uzorka, s i bez korištenja postupka analize glavnih komponenti, i uz pretraživanje grubom silom dan je u tablici 3. Sva mjerenja rađena su za sintetiziranu teksturu veličine *128x128x128*. U prva tri stupca prikazano je prosječno vrijeme potrebno za izvođenje jedne iteracije postupka sinteze na pojedinoj razini. U zadnjem stupcu prikazano je ukupno vrijeme izvođenja ili procjena ukupnog vremena izvođenja. Sva mjerenja rađena su na računalu s Intel Core2Duo procesorom na 2.26 GHz i 2GB radne memorije.

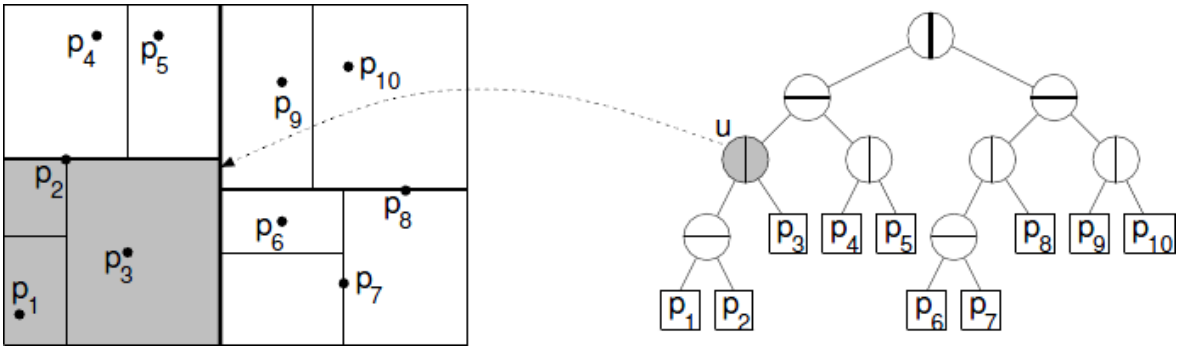
Tablica 3. Trajanje postupka sinteze sa i bez korištenja PCA (format zapisa sati:minute:sekunde).

			Iteracija 1. razine	Iteracija 2. razine	Iteracija 3. razine	Ukupno
Veličina uzorka	64x64	Bez PCA	00:00:01	00:00:40	00:31:26	>04:00:00
		Sa PCA	00:00:01	00:00:09	00:02:32	00:21:42
	128x128	Bez PCA	00:00:05.6	00:04:06	02:17:45	>10:00:00
		Sa PCA	00:00:01.8	00:00:30	00:11:50	00:40:05

Vidljivo je da se korištenjem postupka analize glavnih komponenti dobiva zamjetno ubrzanje postupka sinteze volumne teksture. Samim korištenjem ovog postupka, postupak sinteze postaje upotrebljiv jer se ukupno vrijeme izvođenja smanjuje s reda veličine od nekoliko sati na nekoliko desetaka minuta. Treba napomenuti da sam postupak analize glavnih komponenti traje od svega nekoliko sekundi na prve dvije razine pa do desetak sekundi na posljednjoj razini, i izvršava se samo na početku svake razine postupka sinteze. Iz ovoga je jasno da je trošak dodatnog računanja koji unosimo postupkom analize glavnih komponenti višestruko isplativ zbog ubrzanja postupka sinteze kojeg njime dobivamo.

3.4.2. Ubrzavanje izvođenja smanjenjem broja susjedstava u uzorku koja ćemo pretražiti kod traženja najbližeg susjeda

Smanjene vektore susjedstava uzorka možemo iskoristiti za stvaranje efikasne strukture podataka za traženje najbližeg susjeda. Struktura podataka korištena za pretraživanje je kd-stablo koje se temelji na rekurzivnom dijeljenju prostora dimenzionalnosti *k* u razdvojene hiperpravokutne regije koje se zovu ćelije [8]. Počevši od korijenskog čvora, prostor se hiperravninama rekurzivno dijeli na ćelije sve dok svaka ćelija ne sadrži samo jednu točku, odnosno vektor susjedstva. Primjer postupka u dvodimenzionalnom prostoru prikazuje slika 14.



Slika 14. Kd-stablo i pripadajući 2D prostor podijeljen na ćelije.

Da bi se smanjio broj čvorova koje je potrebno posjetiti prilikom pretraživanja i samim time još više ubrzalo pretraživanje, ne tražimo pravog najbližeg susjeda, već približno najbližeg susjeda (engl. *approximate nearest neighbour*, ANN). Za parametar ε, približno najbliži susjed je onaj koji je od pravog najbližeg susjeda udaljen najviše (1 + ε) puta. Vrijednost paramatra $\varepsilon = 2$ daje dobar kompromis između kvalitete sinteze i brzine izvođenja [5].

Za generiranje kd-stabla i pretraživanje koristi se besplatna biblioteka *ANN*. Detalji vezani za način generiranja stabla i pretraživanje mogu se naći u [8].

Prikaz vremena izvođenja postupka sinteze za različite veličine uzorka, uz korištenje postupka analize glavnih komponenti, s korištenjem strukture kd-stabla za pretraživanje, i uz različite načine pretraživanja (traženje najbližeg i približno najbližeg susjeda uz parametar $\varepsilon = 2$) dan je u tablici 4. Kao i za prethodnu tablicu, sva mjerenja rađena su za sintetiziranu teksturu veličine *128x128x128*. Radi lakše usporedbe prepisani su i rezultati iz prethodne tablice za pretraživanje grubom

silom uz korištenje postupka analize glavnih komponenti. Sva mjerenja rađena su na računalu s Intel Core2Duo procesorom na 2.26 GHz i 2GB radne memorije.

Tablica 4. Trajanje postupka sinteze uz različite vrste pretraživanja (format zapisa sati:minute:sekunde)

			Iteracija 1. razine	Iteracija 2. razine	Iteracija 3. razine	Ukupno
Veličina uzorka	64x64	PCA	00:00:01	00:00:09	00:02:32	00:21:42
		PCA + kd-stablo	00:00:01	00:00:07.5	00:00:47	00:07:38
		PCA + kd-stablo + ANN	00:00:00.9	00:00:05.5	00:00:38	00:05:30
	128x128	PCA	00:00:01.8	00:00:30	00:11:50	00:40:05
		PCA + kd-stablo	00:00:01.8	00:00:16	00:02:59	00:12:09
		PCA + kd-stablo + ANN	00:00:01.2	00:00:08	00:01:06	00:05:06

Vidljivo je da se korištenjem strukture kd-stabla postiže daljnje ubrzanje postupka sinteze. Uz minimalan utjecaj na kvalitetu sinteze, još veće ubrzanje može se dobiti traženjem približno najbližeg susjeda. Naravno da ukupna vremena izvođenja ovise o dimenzionalnosti susjedstava nakon izvođenja postupka analize glavnih komponenti i o brzini konvergencije postupka optimizacije pa se za različite uzorke mogu bitno razlikovati. Unatoč tome, prethodne dvije tablice daju dobru ilustraciju utjecaja metoda za ubrzanje na trajanje postupka sinteze.

3.5. Ostali pojmovi

U dosadašnjim opisima spomenuti su neki pojmovi koji još nisu objašnjeni. Najvažniji među njima su Gaussove piramide i trilinearna interpolacija.

3.5.1. Gaussova piramida

Pojam Gaussove piramide spominje se kod generiranja smanjenih uzoraka za višerezolucijske postupke sinteze tekstura. Gaussova piramida sastoji se od niza slika koje su dobivene iz početne slike filtriranjem niskopropusnim filtrom i smanjenjem filtrirane slike na polovicu originalne veličine slike [9].

Slika se filtrira 1D konvolucijskim filtrom u oba smjera. Vrijednosti jezgre filtra su:

$$[0.05\ 0.25\ 0.4\ 0.25\ 0.05].$$

Budući da ove vrijednosti aproksimiraju Gaussovu jezgru, piramide koje su generirane korištenjem ovakvog filtra nazivamo Gaussove piramide. Nakon filtriranja sliku je potrebno smanjiti. To ćemo obaviti tako da jednostavno odbacimo svaki drugi redak i stupac filtrirane slike. Svaka sljedeća slika dobiva se ponavljanjem ovog postupka na zadnjoj dobivenoj slici. Slika 15 prikazuje niz slika koje predstavljaju Gaussovu piramidu generiranu iz početne slike.

Slika 15. Niz slika koje predstavljaju Gaussovu piramidu.

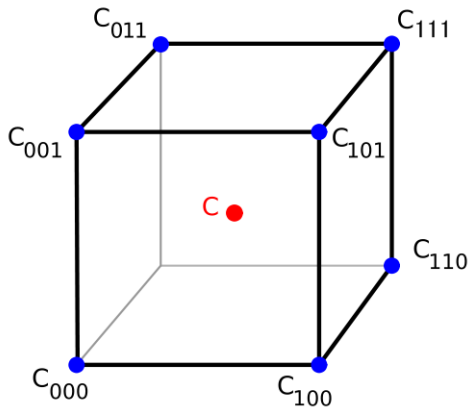
Smanjivanjem uzoraka na ovakav način postupno gubimo detalje, ali zbog filtriranja niskopropusnim filtrom smanjeni uzorci ostaju kvalitetni te su kao takvi

pogodni za sintezu tekstura. Prilikom sinteze generiraju se i koriste samo tri razine piramide koje sadrže originalni uzorak i dva filtrirana i smanjena uzorka.

3.5.2. Trilinearna interpolacija

Kao i Gaussove piramide, trilinearna interpolacija koristi se u višerezolucijskim postupcima sinteze volumnih tekstura. Iako se u tom slučaju koristi samo za povećanje volumne teksture dva puta, u općenitom slučaju trilinearnom interpolacijom mogli bi volumen skalirati proizvoljnim faktorom. Trilinearna interpolacija je trodimenzionalno proširenje jednodimenzionalne linearne interpolacije, odnosno dvodimenzionalne bilinearne interpolacije [10].

Za interpolaciju vrijednosti potrebno je odrediti osam vrijednosti koje okružuju točku u kojoj želimo izračunati interpoliranu vrijednost. Možemo zamisliti da tih osam točaka čini lokalni koordinatni sustav unutar kocke s koordinatama od (0,0,0) do (1,1,1). Na slici 16 prikazana je kocka s označenim vrhovima i točkom interpolacije. Oznake C_{000} do C_{111} označavaju boju, odnosno vrijednost elementa volumena.



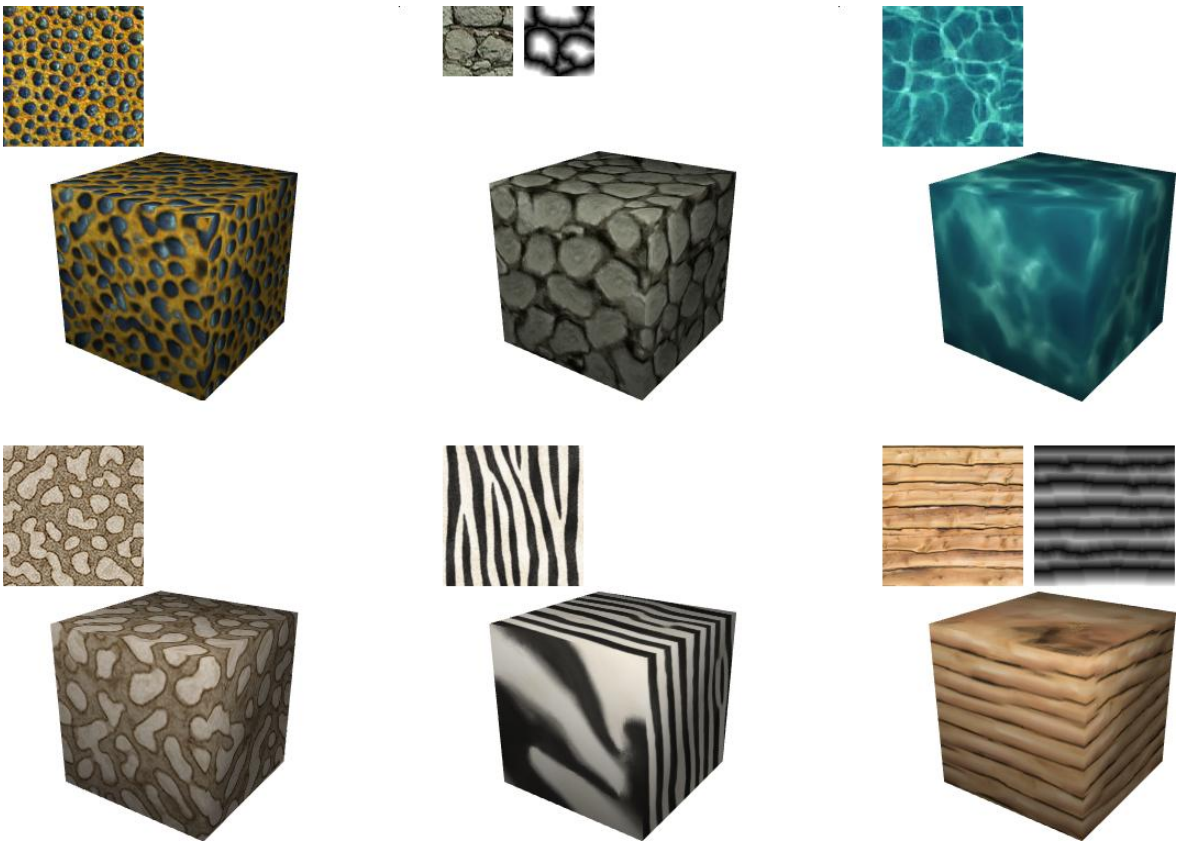
Slika 16. Osam točaka koje okružuju točku interpolacije čine kocku.

Ako točka *C* ima koordinate (x,y,z) u lokalnom koordinatnom sustavu kocke, njenu vrijednost možemo izračunati na sljedeći način.

$$C = C_{000} \cdot (1-x) \cdot (1-y) \cdot (1-z) + C_{001} \cdot (1-x) \cdot (1-y) \cdot z +$$
$$C_{010} \cdot (1-x) \cdot y \cdot (1-z) + C_{011} \cdot (1-x) \cdot y \cdot z +$$
$$C_{100} \cdot x \cdot (1-y) \cdot (1-z) + C_{101} \cdot x \cdot (1-y) \cdot z +$$
$$C_{110} \cdot x \cdot y \cdot (1-z) + C_{111} \cdot x \cdot y \cdot z$$

3.6. Rezultati sinteze volumnih tekstura

Na slici 17 prikazani su rezultati sinteze volumnih tekstura dobiveni prethodno opisanim postupkom. Pored pojedine volumne teksture prikazan je uzorak iz kojeg je tekstura sintetizirana te mapa značajki, ako je korištena.



Slika 17. Rezultati sinteze.

Sve sintetizirane teksture su ponavljajuće (engl. *tileable*) u svim smjerovima. Predzadnja i zadnja tekstura sintetizirane su na način da samo dva presjeka okomita na koordinatne osi odgovaraju zadanom uzorku, dok na svim ostalim sva tri presjeka odgovaraju uzroku. Vidljivo je da su rezultati sinteze bolji na nepravilnim teksturama, dok se na pravilnim teksturama mogu javiti male greške. Iako je rezultat sinteze posljednje teksture vizualno zadovoljavajući, na njemu nije očuvana pravilnost uzorka.

Dobiveni rezultati su u smislu kvalitete i brzine postupka sinteze usporedivi s rezultatima dobivenim u [5]. Značajnije razlike vidljive su jedino kod pravilnih tekstura (drvo, cigla) koje su u rezultatima iz [5] bitno kvalitetnije. Trajanje postupka sinteze za volumnu teksturu veličine *128x128x128* na računalu s Intel

Core2Duo procesorom na 2.26 GHz iznosi od 5 do 45 minuta, ovisno o veličini uzorka, korištenju mape značajki, i uspješnosti smanjenja dimenzionalnosti susjedstava, i brzini konvergencije postupka optimizacije, što je svakako usporedivo s 10 do 90 minuta na 2.4GHz procesoru iz [5].

Na kraju, potrebno je osvrnuti se na radove [4] i [5] koji su u najvećoj mjeri korišteni kod izrade rada. Iako je u njima sama ideja postupaka dobro objašnjena, u opisima pojedinih dijelova nema previše detalja što otežava izradu odgovarajuće implementacije. To se naročito odnosi na sam princip rada višerezolucijskih postupaka sinteze tekstura i na uvjet zaustavljanja postupka optimizacije koji u ovim radovima nije naveden. Unatoč tome, uz malo eksperimentiranja formuliran je dobar uvjet zaustavljanja i pokazalo se da generiranje Gaussove piramide uzorka daje bolje rezultate od korištenja bilinearne interpolacije. Također, u [5] spomenuto je korištenje algoritma grupiranja kako bi se smanjilo zamućenje rezultata, što u konačnici nije implementirano zbog manjka konkretnih informacija. Budući da je to jedina bitna razlika između [5] i implementiranog postupka, moguće je da korištenje algoritma grupiranja poboljšava rezultate sinteze pravilnih tekstura.

4. Teksturiranje objekata volumnim teksturama

Kao jedna od glavnih prednosti volumnih tekstura nad dvodimenzionalnim teksturama istaknuta je jednostavnost teksturiranja objekata, odnosno preslikavanja teksture na objekt.

Slično kako se dvodimenzionalne teksture preslikavaju u *(u,v)* koordinate u rasponu od *(0,0)* do *(1,1)*, tako se volumna tekstura preslika u *(u,v,w)* koordinate u rasponu od *(0,0,0)* do *(1,1,1)*. Budući da se volumna tekstura može nastavljati u svim smjerovima, 3D koordinate površine objekta jednostavno se preslikaju u 3D koordinate volumne teksture te se na taj način teksturira objekt.

Slika 18 prikazuje objekte teksturirane volumnim teksturama. Posebno su zanimljive teksture koje prikazuju prirodne materijale poput drva ili kamena. Tako zmaj izgleda kao da je izgrađen slaganjem kamenčića, a medo kao da je izrezbaren iz drva.

Volumnim teksturama možemo jednostavno postići mnoge zanimljive efekte koji se ne mogu ostvariti dvodimenzionalnim teksturama. Na primjer, ako pomičemo objekt, a volumnu teksturu ostavimo fiksnom, dobit ćemo animaciju prolaska kroz teksturu na površini objekta. Ako objekt ostavimo na fiksnom položaju, možemo interaktivno namještati teksturu da pronađemo najbolji položaj.

Iz svega navedenog vidljivo je da volumne teksture uz jednostavnost korištenja pružaju i puno veću fleksibilnost od dvodimenzionalnih.

Slika 18. Objekti teksturirani volumnim teksturama.

5. Programska implementacija

Opisani postupak sinteze dvodimenzionalnih i volumnih tekstura implementiran je u programskom jeziku C++ uz korištenje *Qt* razvojnog okvira programske podrške (engl. *application framework*). *Qt* omogućava jednostavno kreiranje grafičkih korisničkih sučelja uz jednostavnu integraciju s OpenGL-om koji se koristi za teksturiranje i prikaz rezultata.

5.1. Pregled klasa

Klasa *Texture2D* objedinjuje svu potrebnu funkcionalnost za učitavanje, spremanje i sintezu dvodimenzionalnih tekstura, te sintezu volumnih tekstura iz učitane teksture. Učitavanje slike (uzorka) i pretvorba formata zapisa format pogodan za obradu (24 bitni RGB) obavlja se pomoću klase *QImage Qt* razvojnog okvira. Pomoću iste klase obavlja se i spremanje slike u željeni format. Od ostalih funkcionalnosti klasa podržava generiranje Gaussove piramide iz slike i skaliranje slike korištenjem bilinearne interpolacije.

Klasa *Texture3D* objedinjuje funkcionalnosti za učitavanje, spremanje, manipulaciju volumnih elemenata i sintezu volumnih tekstura. Za učitavanje i spremanje koristi se format zapisa *vol* koji će biti opisan u nastavku. U ovoj klasi implementirano je i skaliranje volumne teksture korištenjem trilinearne interpolacije.

Klase *ImageHistogram* i *ImageHistogram3D* koriste se za kreiranje i praćenje histograma dvodimenzionalne, odnosno volumne teksture. Histogram volumne teksture izveden je od po jednog histograma dvodimenzionalne teksture za svaki presjek okomit na koordinatne osi volumena.

U klasi *TexureSynthesis* korištenjem prethodno navedenih klasa implementirani su postupci sinteze dvodimenzionalnih i volumnih tekstura.

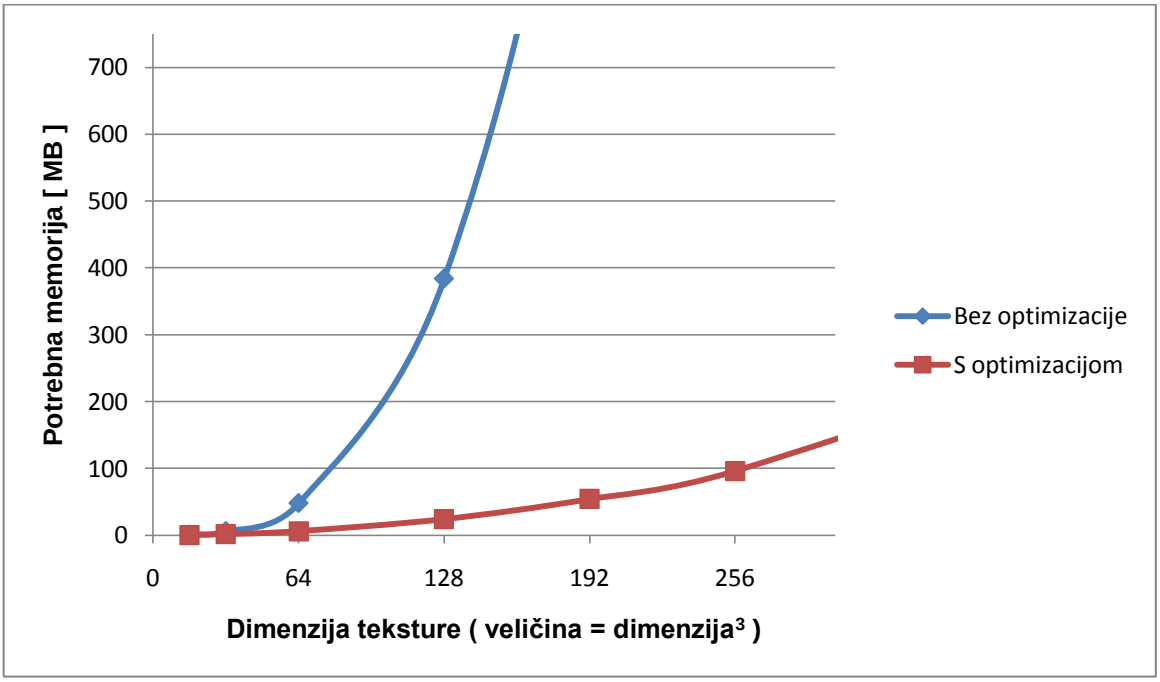
Klasa *GLWindow* objedinjuje funkcionalnosti potrebne za prikaz i teksturiranje objekata.

5.2. Optimizacija zauzeća memorije

Jedan od najvećih izazova prilikom izrade programske implementacije bio je smanjiti količinu memorije koja je potrebna za postupak sinteze volumnih tekstura. Bez optimizacije, zauzeće memorije je preveliko već kod tekstura veličine *128x128x128*. Naime, u fazi pretraživanja za svaki element volumena potrebno je pamtiti sve vrijednosti i težine slikovnih elemenata koje u fazi optimizacije ulaze u izračun novih vrijednosti elementa volumena. Za susjedstva veličine *8x8*, za svaki element volumena potrebno je pamtiti 48 vrijednosti i težina slikovnih elemenata. Odnosno, uz navedenu veličinu teksture potrebno je čak 768 MB za pohranu navedenih podataka (384 MB za vrijednosti i za težine).

Zauzeće memorije može se smanjiti ako jednu iteraciju postupka sinteze podijelimo na više faza pretraživanja i optimizacije. Odnosno, ako se ne traže odjednom najbliža susjedstva uzorka za sva susjedstva volumne teksture i zatim računaju nove vrijednosti svih elementa volumena. Ideja je da tražimo najbliža susjedstva uzorka za susjedstva volumne teksture koja se nalaze unutar volumena veličine $dimenzija \cdot dimenzija \cdot širinaSusjedstva$, i da pomicanjem tog volumena prođemo cijelu volumnu teksturu. Kako se volumen pomiče po volumnoj teksturi skupljaju se vrijednosti i težine slikovnih elemenata, kada se za element volumena skupe svi potrebi podaci izračunava se njegova nova vrijednost.

Na ovaj način se prostorna složenost postupka sinteze smanjuje s kubične na kvadratnu. Za pohranu podataka za teksturu veličine *128x128x128*, umjesto 768 MB, potrebno je samo 48 MB (24 MB za vrijednosti i za težine). Slika 19 prikazuje ovisnost potrebne količine memorije za pohranu vrijednosti slikovnih elemenata (ili težina) o dimenziji teksture, uz pretpostavku da su sve dimenzije volumne teksture jednake, za postupak bez optimizacije i s optimizacijom zauzeća memorije.



Slika 19. Ovisnost potrebne količine memorije o veličini teksture.

5.3. Format zapisa volumnih tekstura (*.vol*)

Radi lakšeg testiranja i uspoređivanja rezultata format zapisa odgovara formatu opisanom u [5]. Prvih 4096 bajtova *vol* datoteke sastoji se od zaglavlja i dopune nulama nakon čega slijedi polje vrijednosti elemenata volumena. Zaglavlje odgovara sljedećoj strukturi.

```
struct VolumeHeader
{
    char magic[4];
    int version;
    char texName[256];
    bool wrap;
    int volSize;
    int numChannels;
    int bytesPerChannel;
};
```

Da bi se zadržala kompatibilnost s [5] elementi zaglavlja moraju sadržavati sljedeće podatke:

- *magic* – mora biti „VOLU“.
- *version* – mora biti 4.
- *texName* – proizvoljno ima teksture.

- *wrap* – *true* ako je volumna tekstura ponavljajuća, inače *false*.
- *volSize* – veličina jedne dimenzije volumne teksture. Sve dimenzije trebaju biti jednake.
- *numChannels* – broj komponenti boje (tri za RGB).
- *bytesPerChannel* – mora biti 1.

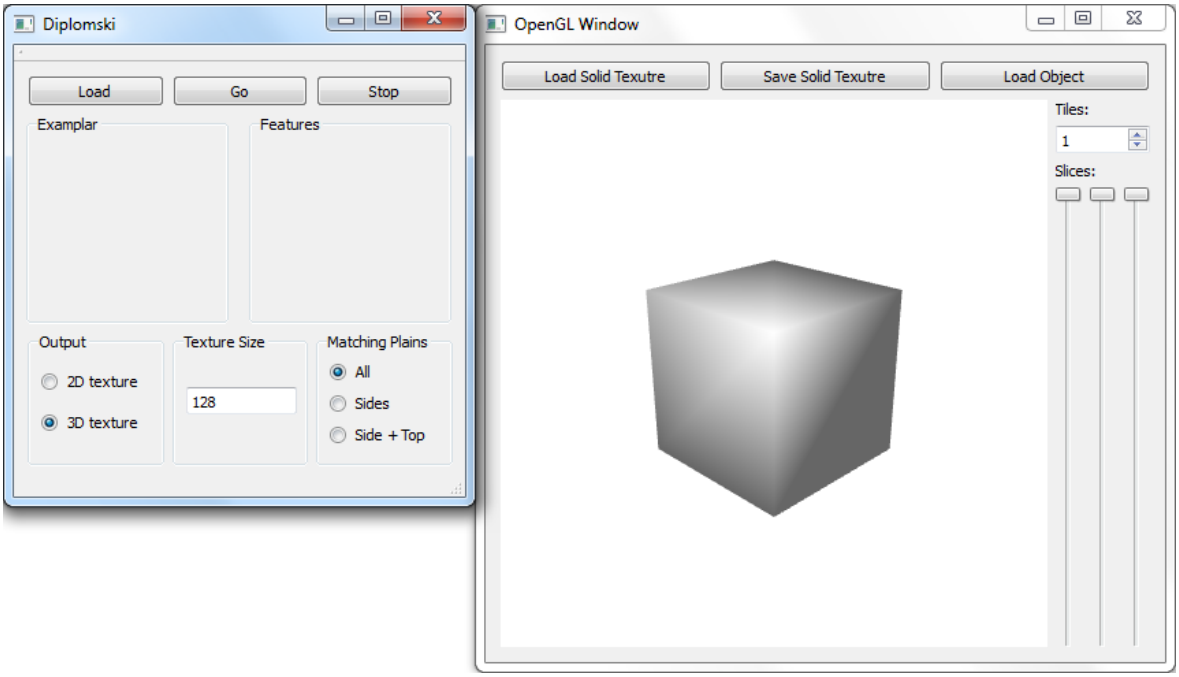
Polje vrijednosti elemenata teksture tipa je *unsigned char* i veličine:

$$size = volSize \cdot volSize \cdot volSize \cdot numChannels \cdot bytesPerChannel$$

5.4. Korisničko sučelje

Izgled korisničkog sučelja nakon pokretanja programa prikazuje slika 20. Aplikacija se sastoji od dva prozora. U prvom prozoru s naslovom „*Diplomski*“ učitavaju se uzorci i podešavaju parametri postupka sinteze tekstura. Pritiskom na gumb *Load* otvara se prozor u kojem je potrebno odabrati uzorak. Nakon odabira uzorka otvara se još jedan prozor u kojem se može odabrati odgovarajuća mapa značajki, ako je želimo koristiti prilikom sinteze. Ove slike se nakon učitavanja skaliraju na veličinu *128x128*.

U donjem dijelu prozora podešavaju se parametri postupka sinteze. Moguće je odabrati željeni izlaz, dvodimenzionalnu ili volumnu teksturu i veličinu teksture (zadaje se samo jedna dimenzija, sve ostale dimenzije su jednake). Zadnji parametar ima utjecaj samo ako je odabrana sinteza volumnih tekstura. Njime odabiremo presjeke koji su okomiti na koordinatne osi za koje želimo da se poklapaju s uzorkom. Moguće je odabrati sva tri presjeka ili jednu od kombinacija dva presjeka, dva bočna ili jedan bočni i gornji.



Slika 20. Izgled korisničkog sučelja programa.

Drugi prozor s naslovom „*OpenGL Window*“ služi za prikaz volumnih tekstura na kocki koja je prikazana nakon pokretanja. U ovom prozoru prikazuje se volumna tekstura za vrijeme postupka sinteze i na taj način može se pratiti napredak postupka. Sintetizirana tekstura može se spremiti pritiskom na gumb *Save Solid Texture*. Pritiskom na *Load Solid Texture* može se učitati prethodno spremljena volumna tekstura. Pomoću tri klizača s desne strane mogu se pregledavati različiti presjeci volumne teksture. Pritiskom na *Load Object* može se učitati drugi objekt koji će biti teksturiran trenutno prikazanom volumnom teksturom. Podržani format za objekte je *wavefront obj* (*.obj*). Promjenom broja u kućici mijenja se broj ponavljanja teksture kod teksturiranja. Pritiskom lijevog gumba i pomicanjem miša unutar prostora za prikaz moguće je rotirati objekt, a pritiskom desnog gumba i pomicanjem miša moguće je pomicati teksturu.

Postupak sinteze pokreće se pritiskom na gumb *Go*, a nakon pokretanja postupak se može zaustaviti pritiskom na *Stop*.

U slučaju sinteze dvodimenzionalnih tekstura, tekstura se sintetizira u novom prozoru iz kojeg se nakon završetka sinteze može spremiti (Slika 21).

Slika 21. Prozor za prikaz rezultata postupka sinteze dvodimenzionalnih tekstura.

6. Zaključak

Postupci sinteze tekstura predstavljaju zanimljivo i zahtjevno područje istraživanja računalne grafike. U ovom radu opisani su osnovi postupci sinteze dvodimenzionalnih tekstura i jedan postupak sinteze volumnih tekstura koji je implementiran u praktičnom dijelu rada.

Opisanim postupkom sinteze mogu se dobiti volumne teksture koje prilikom teksturiranja objekata daju zanimljive vizualne rezultate. Iako su rezultati postupka sinteze volumnih tekstura iz nekih pravilnih uzorka daleko od idealnih, iz velikog broja uzoraka dobivene su kvalitetne volumne teksture koje se mogu koristiti za teksturiranje objekata. Otklanjanje problema u postupku sinteze iz pojedinih uzorka svakako predstavlja izazov za daljnji rad na ovom području.

Budući da je opisani postupak sinteze fleksibilan i nema nikakvih ograničenja na broj uzorka koji će se koristiti prilikom postupka sinteze i na broj komponenti pojedinog uzorka, malim modifikacijama u implementaciji mogu se dobiti razni zanimljivi efekti. Na primjer, moguće je zadati različite uzorke za pojedine presjeke volumne teksture i na taj način utjecati na unutarnju strukturu volumena. Također, moguće je zajedno s uzorkom zadati i pripadajuću mapu normala te na taj način dobiti mapu normala volumne teksture koja se može iskoristiti za preslikavanje izbočina prilikom teksturiranja.

Zbog brzine postupka sinteze i fleksibilnosti, implementirani postupak je dobar temelj za daljnje nadogradnje i modifikacije.

7. Literatura

1. Wei, L.-Y., Lefebvre, S., Kwatra, V., Turk, G.: State of the Art in Example-based Texture Synthesis, *Eurographiscs '09 State of the Art Reports (STARs)*, 2009.,
   <http://research.microsoft.com/apps/pubs/default.aspx?id=79621>.
2. Pietroni, N., Cignoni, P., Otaduy, M.A., Scopigno, R.: A survey on solid texture sythesis, *IEEE Computer Graphics and Applications,* svezak 30, broj 4, 2010., stranice 74-89.
3. Efros, A.A., Leung, T.K.: Texture Synthesis by Non-parametric Sampling, *IEEE International Conference of Computer Vision*, 1999.,
   <http://graphics.cs.cmu.edu/people/efros/research/EfrosLeung.html>.
4. Kwatra, V., Essa, I., Bobick, A., Kwatra, N.: Texture Optimization for Example-based Synthesis, *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2005)*, 2005.,
   <http://www.cc.gatech.edu/cpl/projects/textureoptimization>.
5. Kopf, J., Fu, C.-W., Cohen-Or, D., Deussen, O., Lischinski, D., Wong, T.-T.: Solid Texture Synthesis from 2D Exemplars, *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)*, 2007.,
   <http://johanneskopf.de/publications/solid/index.html>.
6. Ribarić, S.: Predavanja iz kolegija „Raspoznavanje uzoraka“, ZEMRIS, FER, ak. god. 2009/2010.
7. „Principal component analysis“, Wikipedia, 19.5.2011.,
   <http://johanneskopf.de/publications/solid/index.html>.
8. Mount, D.M.: ANN Programming Manual, 2010.,
   <http://www.cs.umd.edu/~mount/ANN>.
9. Kutulakos K.: Predavanja iz kolegija „Introduction to Visual Computing“, predavanje „Gaussian and Laplacian Pyramids“, University of Toronto, 2011.,
   <http://www.cs.toronto.edu/~kyros/courses/320>.
10. „Trilinear interpolation“, Wikipedia, 30.5.2011.,
    <http://en.wikipedia.org/wiki/Trilinear_interpolation>.

Postupci sinteze volumnih tekstura i teksturiranja objekata

Sažetak

U ovom radu opisani su osnovni postupci za sintezu dvodimenzionalnih tekstura i postupak za sintezu volumnih tekstura iz dvodimenzionalnih uzorka. Tematika rada ukratko je opisana u uvodnom poglavlju. U drugom poglavlju opisani su osnovni pojmovi vezani za sintezu tekstura i dan je pregled osnovnih postupaka za sintezu dvodimenzionalnih tekstura. U trećem poglavlju opisan je postupak sinteze volumnih tekstura korištenjem optimizacije teksture uz dodatak metode podudaranja histograma koja poboljšava rezultate sinteze. U nastavku poglavlja opisane su metode kojima je moguće ubrzati postupak sinteze i prikazani su dobiveni rezultati postupka sinteze i teksturiranja objekata. Zadnje poglavlje ukratko opisuje programsku implementaciju i korisničko sučelje.

Ključne riječi: analiza glavnih komponenti, podudaranje histograma, sinteza tekstura, traženje najbližeg susjeda, volumne teksture

Solid texture synthesis and texture mapping

Abstract

This paper describes the basic methods for 2D texture synthesis and the method for solid texture synthesis from 2D exemplars. The theme is briefly described in the introduction. The second chapter describes the basic concepts related to texture synthesis and basic methods for 2D texture synthesis. The third chapter describes the method for solid texture synthesis which uses texture optimization approach with histogram matching to improve synthesis results. In the continuation of the chapter there is a description of methods used to speed up the synthesis process, and the results of synthesis process and texture mapping are shown. The last chapter briefly describes the implementation and user interface.

Keywords: histogram matching, nearest neighbour search, principal component analysis, solid textures, texture synthesis