SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 214

POSTUPAK SINTEZE IZRANJAJUĆIH SLIKA

Marina Tajić

Zagreb, lipanj 2011.

ZAHVALA

Ovim putem želim se zahvaliti svima koji su mi svojim strpljenjem, znanjem i potporom pomogli tijekom studiranja i izrade ovoga rada, posebice mojim roditeljima i mentorici prof. dr. sc. Željki Mihajlović.

Sadržaj

"The whole is greater than the sum of the parts"

1. Uvod

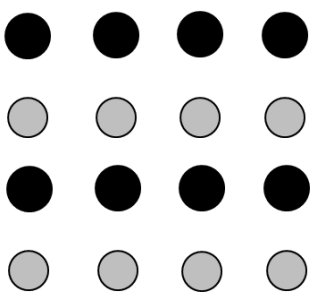
Pojavom interneta i različitih usluga koje nam on pruža pojavili su se i programi za automatizirano raspoznavanje namijenjeni upravo za iskorištavanje i/ili ugrožavanje tih usluga, bilo to opetovano ostavljanje poruka na nekom blogu ili forumu, trošenje resursa nekog portala neprestanom registracijom novih korisničkih računa, automatizirana potraga za podacima i prijenos pronađenih podataka na lokalno računalo ili pak zatrpavanje nečijeg e-maila nepoželjnim porukama. Kao rješenje problema automatiziranih programa javili su se testovi koji mogu razlikovati da li zahtjev postavlja čovjek ili računalo. Najpoznatiji i najrašireniji način raspoznavanja čovjeka i računala je CAPTCHA, vrsta autentifikacije pri kojoj servis koji nudi uslugu postavlja pitanje i očekuje na njega odgovor, a ovisno o odgovoru može reći da li je odgovor ponudio čovjek ili računalo. Porastom računalne moći i razvojem algoritama za raspoznavanje uzoraka razvijeno je nekoliko načina za zaobilaženje ove sigurnosne granice. Iako je CAPTCHA još uvijek uglavnom sigurna način zaštite od automatiziranih programa javlja se potreba za boljim načinima zaštite. Ovaj rad bavi se jednim od mogućih načina zaštite, izranjajućim slikama, tj. slikama iz kojih „izranja“ neka podcjelina koja čovjeku ima značenje i smisao, dok automatiziranim programima predstavlja samo skupinu slučajnih uzoraka. Sinteza izranjajućih slika obavlja se teksturiranjem ljudima poznatog objekta, npr. 3D model konja, različitim „mrljama“ nakon čega se dio tih mrlja kopira u okolinu teksturiranog objekta stvarajući smetnje. Gledajući tu sliku čovjek može uočiti u gomili mrlja objekt koji mu je poznat. U nastavku ovoga rada prvo se objašnjavaju Gestalt psihologija i jedan on temeljnih principa te psihologije, princip izranjanja. U nastavku slijede poglavlja s objašnjenim postupkom sinteze izranjajućih slika, opisom implementacije i raspravom o dobivenim rezultatima. Na kraju se nalaze zaključak, sažetak i popis korištene literature.

2. Gestalt psihologija

Gestalt psihologija ili gestaltizam je teorija uma nastala na Berlinskoj školi eksperimentalne psihologije koja tvrdi da je operacijski princip mozga holistički, paralelan i analogan sa tendencijama samoorganiziranja [1]. Termin *gestalt* označava „ujedinjenu cjelinu“. Gestalt percepcija pokušava objasniti kako ljudi organiziraju vizualne elemente u grupe ujedinjenih cjelina primjenom određenih zakona koji su sadržani u temeljenom principu Gestalt psihologije zvanom prägnanz zakon. Prägnanz zakon govori da ljudi nastoje organizirati svoja iskustva na način koji je kontinuiran, uređen, simetričan i jednostavan. Postoji nekoliko zakona koji su podskup prägnanz zakon, a to su:

- sličnost (*similarity*)
- kontinuiranost (*continuity*)
- zatvaranje (*closure*)
- blizina (*proximity*)
- simetričnost (*symmetry*)
- zajednička sudbina (*common fate*)

Zakon sličnosti govori da ljudi često objekte koji su slični doživljavaju kao grupu objekata ili uzorak (slika 1).



Slika 1. Primjer slike za zakon sličnosti

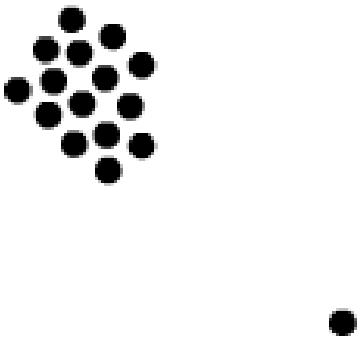
Zakon kontinuiranosti vrijedi za situacije kada je oko primorano prijeći s trenutno promatranog elementa na idući kako bi mozak nastavio kontinuitet uzorka (slika 2).

Slika 2. Primjer za zakon kontinuiranosti

U slučaju kada je objekt nepotpun ili prostor nije do kraja popunjen mozak nadopunjava informacije koje nedostaju kako bi percipirao cjelinu (slika 3). Tada govorimo o **zakonu zatvaranja**.

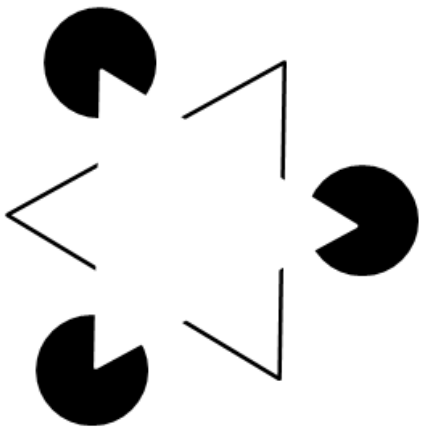
Slika 3. Primjer za zakon zatvaranja

Prostorna ili vremenska blizina objekata može nagnati mozak da ih percipira kao cjelinu o čemu govori **zakon blizine** (slika 4).



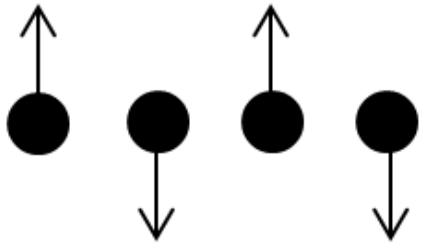
Slika 4. Primjer za zakon blizine

Zakon simetričnosti nalaže da mozak simetrične objekte percipira kao cjelinu neovisno o međusobnoj udaljenosti simetričnih objekata (slika 5).



Slika 5. Primjer za zakon simetričnosti

Prema **zakonu zajedničke sudbine** svi objekti koji se kreću u istom smijeru percipiraju se kao cjelina (slika 6).



Slika 6. Primjer za zakon zajedničke sudbine

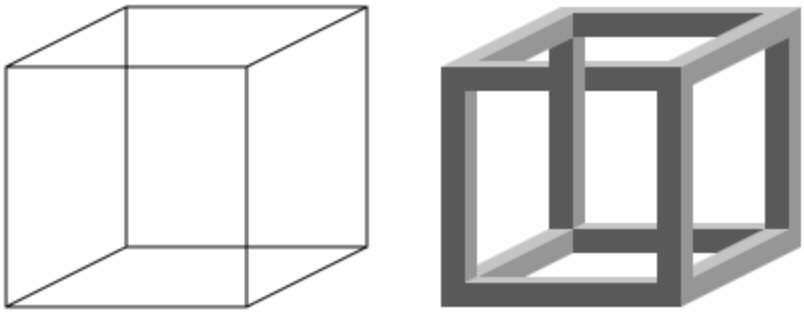
Osim prägnanz zakona gestalt percepcije postoje i četiri osnovna principa gestalt sustava:

- izranjanje (*emergence*)
- multistabilnost (*multistability*)
- invariantnost (*invariance*)
- konkrezitam (*reification*)

Princip izranjanja je proces stvaranja složenih uzoraka koristeći skup jednostavnih pravila. Upravo ovaj princip korišten je prilikom izrade ovoga rada i opisan je detaljnije u slijedećem poglavlju.

Jedan od oblika precepcijskih fenomena je i **princip multistabilnosti**. Ovaj princip predstavlja tendenciju neke vizualne pobude da ju doživim na dva ili više načina.

Postoji nekoliko poznatih primjera multistabilnosti, a najpoznatiji su Neckerova kocka (slika 7.a) i Rubinova figura (slika 7.b).



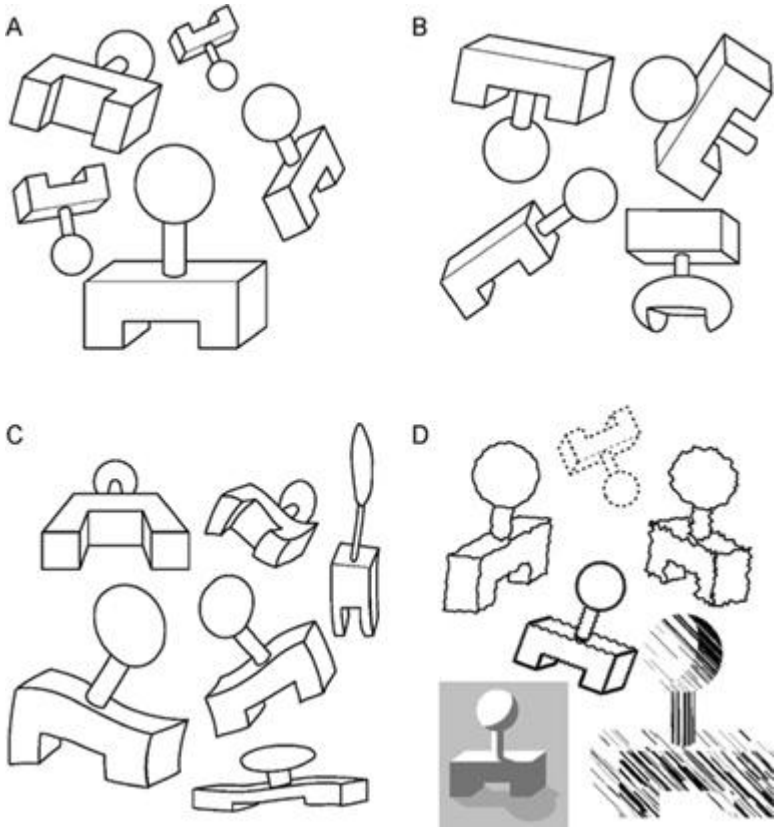
a)

b)

Slika 7. Primjer za princip multistabilnosti, a) Neckerova kocka, b) Rubinova figura

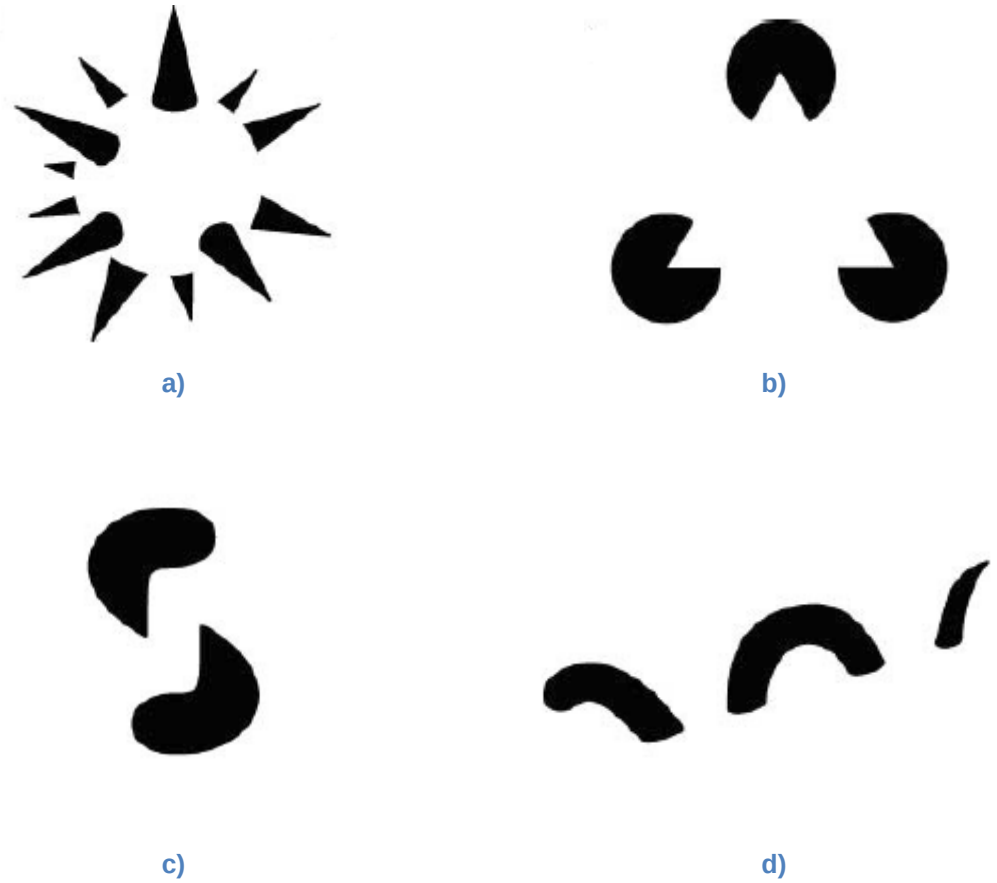
Treći princip gestalt sustava je **princip invarijantnosti** koji govori da jednostavne geometrijske likove prepoznajemo neovisno o njihovoj rotaciji, translaciji, skaliranju, drugačijem osvjetljenju, elastičnim deformacijama i sl. Na slici 8., koja prikazuje primjer za princip invarijantnosti, jasno je vidljivo da su objekti skupine A različiti od objekata skupine B i to neovisno o rotaciji i skaliranju predmeta u grupama. Iako su objekti u grupi C elastično deformirani, a oni u grupi D iscrtani

različitim tehnikama još je uvijek jasno vidljivo da su to isti elementi kao oni u grupi A.



Slika 8. Primjer za princip invarijantnosti (primjer preuzet iz [3])

Konkretizam je četvrti princip gestalt sustava i konstruktivan (generativan) aspekt percepcije zbog kojega percipiramo eksplicitnije prostorne informacije od informacija koje dobivamo gledanjem slike, tj. apstraktne objekte doživljavamo kao stvarne, konkretne objekte. Nekoliko primjera principa konkretizma prikazano je na slici 9.



Slika 9. Primjer za princip konkretizma, a) Idesawina bodljikava kugla, b) Standardni Kanisza trokut, c) Volumetrični crv Petera Tsea, d) Morsko čudovište Petera Tsea

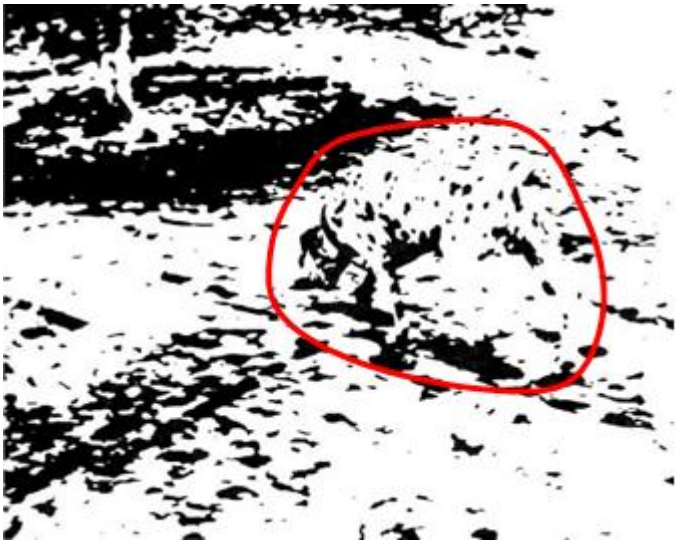
Navedeni principi gestalt sustava nisu nužno međusobno odvojeni već mogu biti različiti aspekti složenih sustava.

3. Izranjanje

Kao što je rečeno u prethodnom poglavlju jedan od temeljnih principa gestalt sustava i teorijski temelj ovoga rada je princip izranjanja. Ovaj fenomen može se definirati kao mogućnost čovjeka da percipira objekte na slici prepoznajući ih kao cjelinu umjesto kao skup dijelova objekta. Stroža definicija izranjanja bila bi da je izranjanje proces formacije kompleksnih uzoraka pomoću jednostavnih pravila. Najpoznatiji primjer izranjanja je slika dalmatinskog psa koji njuška u sjeni drveća (slika 10). Na prvi pogled čovjek na slici ne vidi ništa osim skupine naizgled nasumičnih mrlja.

Slika 10. Izranjajuća slika psa

Pomnijim promatranjem promatraču odjednom postaje vidljiv lik psa spuštene glave koji njuška zemlju (slika 11). Čovjek ne prepoznaje psa na slici na način da prvo prepozna njegove dijelove, npr. uši ili noge, već ga percipira kao cjelinu.



Slika 11. Izranjajuća slika psa s označenim likom psa

Izranjanje je jedan od ključnih principa Gestalt sustava prema kojem objekti *izranjaju* samo kada su važni dijelovi prikazani zajedno dajući osjećaj cjeline, tj. cijelog objekta. Gledajući samo dijelove slike oni nam se čine besmisleni i nasumični, ali gledajući cijelu sliku objekt izranja i percipiramo ga kao cjelinu. Teorija Gestalt sustava ne objašnjava zašto se fenomen izranjanja događa već samo što se događa. Razlog tomu je činjenica da način na koji percipiramo takve objekte još uvijek nije razjašnjen. Vjeruje se da je kompleksnost ljudskog procesuiranja ovakvih i sličnih pojava vrlo velika zbog čega se pretpostavlja da su izranjajuće slike automatskim algoritmima teške za segmentiranje, identificiranje i prepoznavanje, te je vjerojatnost da će automatizirani sustavi moći obavljati prepoznavanje ovakvih slika u skoroj budućnosti vrlo malena, ako ne i nepostojeća [2].

4. Postupak sinteze

Ostvareni postupak sinteze izranjajućih slika zasniva se na algoritmu opisanom u [2], uz neke promjene postupka.

Prilikom postupka sinteze izranjajućih slika postoje dva osnovna zahtjeva koje generirane slike i/ili sekvence moraju zadovoljavati [2]:

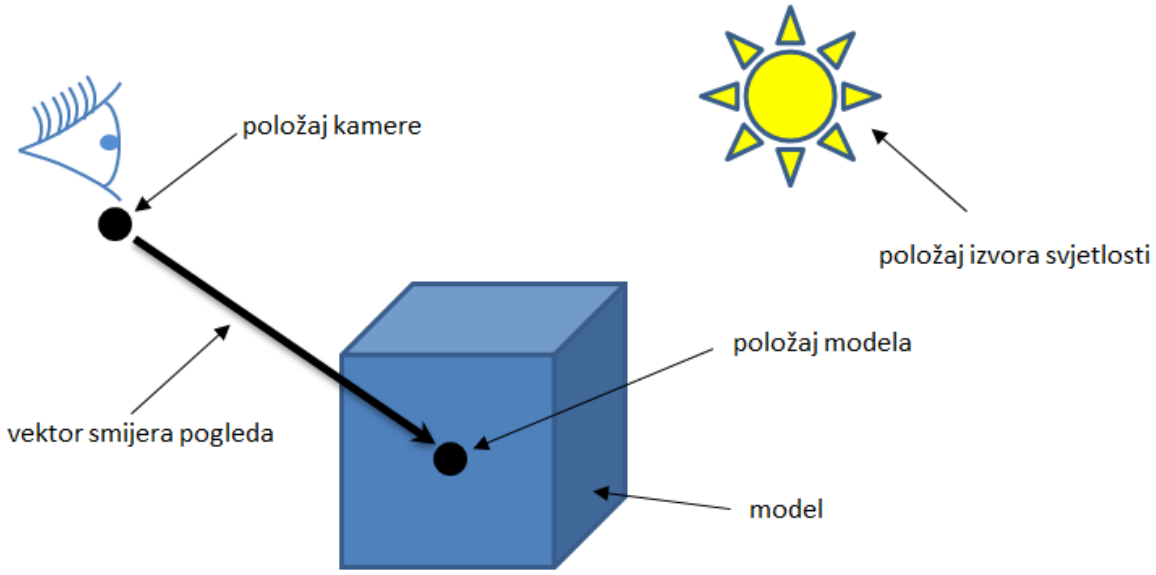
- čovjek mora moći prepoznati objekt na generiranim slikama
- automatizirani program nesmije moći išta prepoznati na generiranim slikama

Osnovna ideja je iscrtati 3D model objekta na način da je sastavljen od osnovnih elemenata nazvanih mrlje. Mrlje same za sebe nemaju neko posebno značenje i ne nose korisne informacije, kako čovjeku tako ni automatiziranom sustavu. Zbog ovog svojstva algoritmi za detekciju rubova, silueta i sl. nisu u mogućnosti dobiti smislene rezultate. Raspored mrlja na slici nije potpuno nasumičan već polu-nasumičan pošto je ipak potrebno poštivati siluetu i oblik modela kako bi se mogle generirati slike s efektom izranjanja. Studije napravljene za potrebe originalnog rada [2] pokazale su da ljudsko prepoznavanje objekata na izranjajućim slikama ovisi o položaju objekta i njihovom upoznatošću s objektom, tj. ako je objekt na slici objekt s kojim se susreću svaki dan i taj objekt je u položaju koji je za njega uobičajen ljudi će ga lakše i brže percipirati na izranjajućoj slici.

Implementirani algoritam sinteze izranjajućih slika sastoji se od nekoliko koraka prikazani pseudokodom:

```
I_M ← MapaVažnostiVrhova(M, v, l)
OdaberiVrhove(n)
foreach veličina prozora do
        OdaberiPodskupMrlja()
foreach poskup mrlja do
        for i in [0,k] do
                kopiraj na novu poziciju oko objekta
                skaliraj
                rotiraj oko pozicije slučajno odabrane mrlje iz skupine
                dodaj u listu mrlja koje stvaraju „smetnju“ oko objekta
IscrtajObjektOdMrlja()
IscrtajMrljeSmetnje()
```

Postupak sinteze izranjajućih slika započinje generiranjem mape važnosti vrhova. Mapa važnosti vrhova svakom vrhu pridodaje vrijednost koja označava važnost vrha, a generira se na temelju dvije mape: mape siluete i mape sjenčanja. Obje mape generiraju se upotrebom modela M, smijera pogleda v i položaja izvora svjetlosti l (slika 12). Mapa siluete izgrađuje se na način da se svakom vrhu za koji je njegova normala skoro okomita ili okomita na odgovarajući vektor pogleda (vektor između položaja vrha i položaja očišta) pridodjeljuje jedinična vrijednost, a svim njegovim susjedima pola jedinične vrijednosti. Prilikom testiranja okomitosti jedinična vrijednost pridodaje se vrhovima kod kojih kut između normale i vektora pogleda iznosi $90° \pm 5°$. Ako vrh već ima pridodijeljenu vrijednost, nova vrijednost se zbraja s prethodnom.



Slika 12. 3D scena

Druga mapa koja se koristi za generiranje mape važnosti vrhova je mapa sjenčanja. Ova mapa izrađuje se na način da se za svaki vrh pohrani vrijednost izraza:

$$(1 - n.l),$$

gdje su:

- n – jedinični vektor normale vrha
- l – jedinični vektor smjera svjetlosti

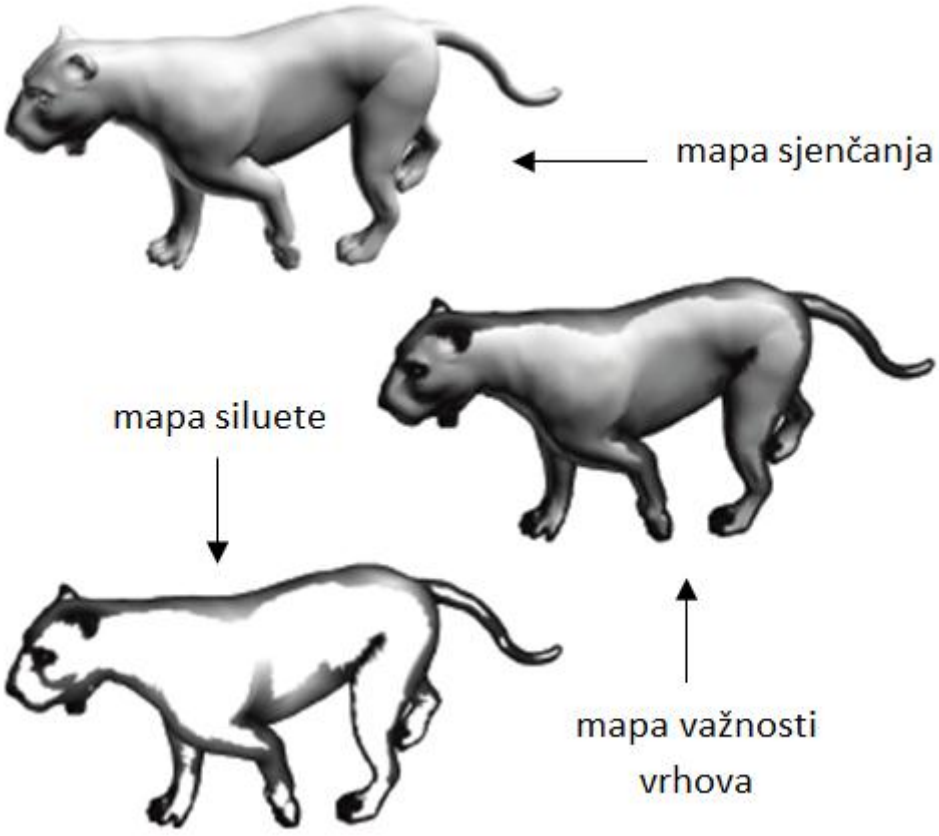
Po završetku generiranja mape siluete i mape sjenčanja gradi se mapa važnosti vrhova na način da se u nju za svaki vrh pohranjuje vrijednost:

$$\max(\mathrm{M}_{\mathrm{silhouette}}, \mathrm{M}_{\mathrm{shading}}),$$

gdje su:

- $M_{silhouette}$ - mapa siluete
- $M_{shading}$ - mapa sjenčanja

Na slici 13 prikazan je izgled mape sjenčanja, mape siluete i mape važnosti vrhova dobivene spajanje mapa siluete i sjenčanja. Iz slike mape važnosti vrhova vidljivo je da će za vrhove koji nose više informacija za sjenčanje objekta prevagnuti podatak iz mape sjenčanja, dok će za vrhove koji nose informaciju o silueti prevagnuti informacija iz mape siluete.



Slika 13. Mape važnosti vrhova, siluete i sjenčanja (slike preuzete iz [2])

Nakon izgradnje mape važnosti vrhova izabire se prvih n% vrhova na čijim će se pozicijama iscrtati mrlje. Broj vrhova koji će biti izabrani ovisi o izabranoj težini koja govori koliko će okolinskih smetnji biti u okolini objekta i koliko će vrhova biti izabrano za lokacije mrlja čime se čovjeku otežava ili olakšava prepoznavanje objekta na izranjajućoj slici. Prije odabira prvih n% vrhova vrhovi se sortiraju prema važnosti, tj. prema vrijednostima u mapi važnosti vrhova i to od najvažnijih, onih s najvećom vrijednošću, prema najmanje važnima, onima s najmanjom vrijednošću. Kada su izabrani vrhovi čiji će položaji poslužiti za iscrtavanje mrlja, pomoću tri veličine prozora izabiru se podskupovi iscrtanih mrlja. Svaki od tri podskupa se zatim kopira u okolinu objekta iscrtanog pomoću mrlja i skalira ovisno o izabranoj težini. Nakon skaliranja položaj jedne mrlje iz podskupa odabire se kao os rotacije oko koje se cijeli podskup rotira. Ovaj postupak se ponavlja za svaki podskup mrlja određen broj puta koji je definiran za svaki podskup posebno i u ovisnosti o izabranoj težini. Prilikom rotacije svake kopije podskupa os rotacije izabire se nasumično i ne ovisi o osi drugih kopija podskupa. Ovime završava postupak sinteze izranjajuće slike. Operacija kopiranja i perturbiranja podskupa mrlja osigurava da svi dijelovi generirane slike imaju slična statistička svojstva čime se automatizitanom programu otežava prepoznavanje izranjajućeg objekta. U ovom radi implementiran je postupak generiranja sekvenci izranjajućih slika koji za generiranje prve slike koristi upravo opisani postupak. Ostale slike generiraju se na način da se mrlje koje iscrtavaju objekt translatiraju na novi položaj i od tri već izabrana podskupa mrlja generiraju se nove okolinske smetnje na način kao i za prvu sliku sekvence. Dakle, za svaku sliku od druge slike na dalje koriste se isti skupovi mrlja, bilo za iscrtavanje objekta bilo za okolinske smetnje, samo na različitim lokacija i u slučaju okolinskih smetnji s različitim perturbacijama.

5. Implementacija

Programska implementacija izrađena je pomoću Visual Studio 2010 razvojne okoline. Od programskih jezika korišten je C# 4.0 programski jezik. Iscrtavanje grafike implementirano je korištenjem je Microsoft XNA 4 programskog okruženja (engl. *framework*). Prilikom prilagodbe i izrade 3D modela korišten je program Autodesk Maya 2009. Teksture mrlja izrađene su pomoću CorelDRAW Graphics Suite X5 programa za grafički dizajn.

5.1. Microsoft XNA

Microsoft XNA je sučelje za programiranje aplikacija (engl. *Application Programming Interface*) namijenjeno prvenstveno za razvoj računalnih igara. Ovo programsko okruženje omogućava programerima da razvijaju programe korištenjem naredbi koje u sebi imaju vrlo malo ili uopće nemaju platformski specifične dijelove zbog čega programer ne mora voditi detalje o implementaciji na određenoj platformi već se može usredotočiti na logiku i sadržaj programa. O implementacijskim detaljima na raznim platformama se brine programsko okruženje. Osim oslobađanja programera implementacijskih detalja, XNA također pokušava minimizirati pisanje repetitivnog koda koji je vrlo malo ili uopće nije promijenjen. Od razvojnih platforma ovaj *framework* podržava slijedeće platforme:

- Microsoft Windows XP/Vista/7
- Xbox 360
- Zune
- Windows Phone 7

XNA podržavaj razvoj i u 2D i u 3D prostoru i sadrži veliki broj biblioteka specifičnih za razvoj računalnih igara.

5.2 Autodesk Maya

Autodesk Maya (skraćeno Maya) je 3D grafički program koji se koristi za izradu interaktivnih 3D sadržaja. Izrađeni sadržaji upotrebljavaju se u animiranim filmovima, računalnim igrama, televizijskim serijama, arhitekturi itd. Prilikom rada s Maya programom korisnik definira virtualni radni prostor ili scenu u kojoj zatim može implementirati nove ili uređivati već postojeće objekte projekta. Izrađeni objekti mogu se pohraniti u raznim formatima kao npr. .fbx ili .mb (Maya Binary). Komponente ovog programa uključuju:

- Fluid Effects – realistični simulator fluida (tekućine, dim, eksplozije itd.)
- Classic Cloth – simulator tkanina i odjeće na animiranim objektima
- Fur – simulator krzna i objekata sličnih krznu (npr.trava)
- Hair – simulator realistične ljudske kose
- idr.

Osim vizualne radne plohe Autodesk Maya pruža mogućnost rada pomoću svojeg ugrađenog skriptnog jezika pod nazivom Maya Embedded Language ili MEL. Osim za pisanje skripti ovaj jezik je namijenjen i za prilagođavanje osnovnog programa.

5.3. CorelDRAW X5

CorelDRAW je vektorski grafički editor kojeg je razvila tvrtka *Corel Corporation.* Razvijen je primarno za *Windows* operativne sustave. CorelDRAW nudi korisnicima razne mogućnosti kao npr. korištenje CMYK formata boja umjesto RGB ili alat za uređivanje čvorova koji radi drugačije na drugačijim objektima. Svaki od CorelDRAW grafičkih paketa sadrži nekoliko različitih programa, ovisno o verziji. Korišteni CorelDRAW Graphics Suite X5 sadrži slijedeće:

- CorelDRAW - editor za vektorsku grafiku
- Corel PHOTO-PAINT - program za stvaranje i izmjenu rasterske grafike
- Corel CONNECT - organizator sadržaja
- Corel CAPTURE - program koji podržava različite metode hvatanja slika
- Corel PowerTRACE - konverter između rasterske i vektorske grafike
- Bitstream Font Navigator
- SB Profiler

Od navedenih alata iz CorelDRAW grafičkog paketa korišteni je CorelDRAW editor za vektorsku grafiku.

5.4. Struktura programa

Kao i svi Microsoft XNA projekti glavna klasa programa i mjesto gdje je smještena glavna upravljačka logika programa je klasa Game1. Osim ove klase stvoreno je još 8 klasa u kojima se nalaze funkcije potrebne za iscrtavanje scene, modela i izranjajućih slika, kao i funkcije koje obavljaju izračune potrebne za generiranje izranjajućih slika. Popis svih klasa i njihov opis nalazi se u tablici 1.

Tablica 1. Popis klasa i njihov opis

Ime klase	Opis klase
Game 1	Glavna klasa programa u kojoj se nalazi upravljačka logika programa
Painter	Klasa u kojoj su sadržane funkcije za iscrtavanje scene i objekata u sceni.
Camera	Klasa koja sadrži programski kod za upravljanje kamerom.
Model3D	Ova klasa sadrži učitani model, funkcije za transformaciju model i ostale podatke potrebne za iscrtavanje modela. Također, sadrži i podatke o mrljama koje se iscrtavaju prateći siluetu objekta.
SplatModel	Klasa koja sadrži jedan od modela mrlja, funkcije za transformaciju modela i ostale podatke potrebne za iscrtavanje mrlje.
SplatCentersGenerator	Funkcije koje se bave odabirom pozicija mrlja kojima se iscrtava objekt implementirane su u ovoj klasi.
SplatChooserModel	Model prozora kojim se izabiru podskupovi mrlja pohranjen je u instanci ove klase kao i funkcije za njegovu transformaciju.
VertexNeighbour	Instance ove klase pohranjuju podatak o vrhu i njegovim susjedima.
EmergingImageGenerator	Dio logike postupka sinteze izranjajućih slika i logika upravljanja potupkom sinteze nalazi se u ovoj klasi.

Klasa Program nije opisana niti je ubrojena u popis klasa pošto je obavezna za bilo koji projekt .NET programskog okvira. Dijagram klasa prikazan je na slici 14 i na njemu nisu prikazane veze između klasa zbog njihovog prevelikog broja. Iz istog razloga nisu prikazani niti atributi i metode implementiranih klasa.

Slika 14. Dijagram klasa

Izvođenje programa počinje prikazivanjem početne scene korisniku na kojoj je vidljiv inicijalno postavljen objekt, razne upute i informacija o odabranoj težini (slika 15). Kao što je već rečeno, težina govori koliko će biti okolinskih smetnji i koliko će vrhova objekta biti izabrano da se na njihovoj lokaciji iscrta mrlja.

Slika 15. Početni ekran

Nakon što je korisnik odabrao težinu i model, te postavio model u željenu pozu prelazi se u drugu fazu sinteze izranjajućih slika. U klasi Game1 instancira se klasa EmergingImageGenerator, a konstruktoru se predaju inasnaca klase 3D model koja sadrži podatke o modelu i instanca klase Painter koja sadrži podatke potrebne za iscrtavanje na ekran. Nakon toga poziva se funkcija EmergingImageGenerator klase koja postavlja parametar težinu na opciju koju je izabrao korisnik. Ponuđene razine težine su:

- lagano
- srednje teško
- teško

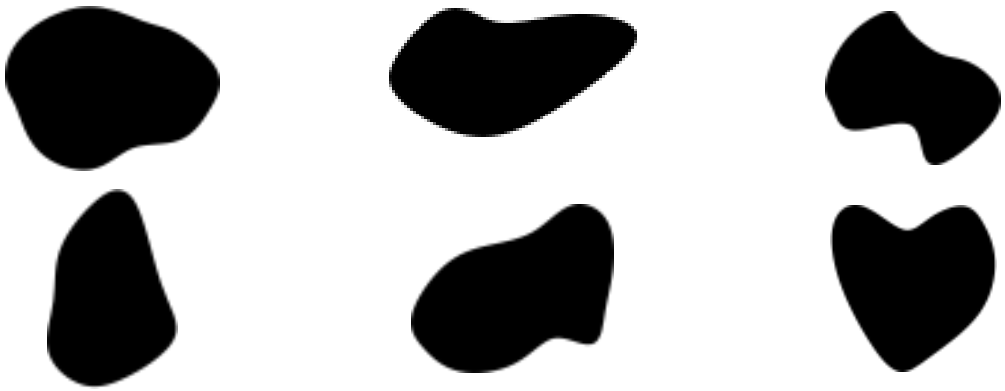
Namještanjem težine odabiru se vrijednosti nekih postavki. U tablici 2 prikazane su postavke i njihove vrijednosti za svaku od mogućih težina. Postotak vrhova označava koliko će se najboljih vrhova rangiranih po vrijednostima u mapi važnosti vrhova uzeti kao pozicije mrlja. Faktor skaliranja je parametar koji govori koliko će se mrlje koje se preslikavaju smanjiti skaliranjem u ovisnosti o težini. Vrijednost za

koju će se smanjiti faktor skaliranja mrlje se dobiva tak da se slučajno generira broj iz intervala, koji ovisi o odabranoj težini, i onda se podijeli sa 200. Dobiveni broj oduzme se od trenutnog faktora skaliranja mrlje. Nakon ovog podešavanja veličine mrlje potrebno je ponovno izračunati matricu svijeta mrlje pošto je svaka mrlja zasebni objekt. Zadnji parametar je broj kopiranja podskupa mrlja koji govori koliko puta će se neki podksup mrlja kopirati u okolinu objekta. Kako je već rečeno u prethodnom poglavlju, nakon odabira pozicija mrlja koje će se iscrtati umjesto objekta izabiru se tri podskupa mrlja koji će se kopirati okolo objekta i time stvoriti smetnje. Veličina podskupa ovisi o veličini prozora kojim se odabiru mrlje pa tako imamo mali podskup, srednji podskup i veliki podskup.

Tablica 2. Postavke parametara za različite težine

Težina	Parametri		
	Postotak vrhova (n%)	Faktor skaliranja (min, max)	Broj kopiranja podksupa mrlja (mali/srednji/veliki)
Lagano	100	(-10, -7)	(60/25/20)
Srednje	50	(-5, -3)	(90/35/25)
Teško	35	(-1, 0)	(100/55/30)

Po uspješnom postavljanju navedenih parametara težine, algoritam prelazi na generiranje položaja mrlja koje će se iscrtati umjesto objekta. Algoritam odabire prvih n% najboljih vrhova prema mapi važnosti vrhova i sprema u listu podatak o koordinati pozicije vrha. Nakon ekstrakcije koordinata pozicija mrlja koje će iscrtavati model, instanca klase painter dobiva te podatke i određuje nasumično za svaku koordinatu koja mrlja će biti iscrtana na tom mjestu. Svaka mrlja je zapravo 3D model spremljen u instanci klase SplatModel. Korišteno je ukupno 6 modela mrlja čije su teksture prikazane na slici 16. Model mrlje je u obliku 2D plohe teksturirane teksturom koja je prozirna svugdje oko crne mrlje.



Slika 16. Teksture mrlja

Nakon slučajnog odabira mrlja koje će iscrtavati objekt, one se i iscrtaju u scenu (slika 17) prikazujući model. Iz slike 17 također je vidljivo da se okolina (pravokutna mreža) iz početne scene ne koristi u procesu sinteze izranjajućih slika već samo 3D model objekta.

Slika 97. Silueta objekta iscrtanim mrljama

Zajedno s objektom iscrtat će se i prvi od tri prozora koji služe za odabir podskupa mrlja koje će se koristiti za okolinske smetnje. Nakon što korisnik odabere prvi podskup mrlja postupak se ponavlja za još dvije veličine prozora (slika 18). Odabiranje podskupa mrlja prozorima funkcionira na način da se traži kolizija između kružne obujmice prozora i kružnih obujmica modela mrlja koje iscrtavaju objekt. Svi modeli mrlja čije su obujmice u koliziji s obujmicom prozora ulaze u podskup mrlja. Kada završi odabir sva tri podskupa mrlja slijedi završni dio algoritma, stvaranje okolinskih smetnji. Okolinske smetne stvaraju se na načina da se dobivena tri podskupa mrlja kopiraju određen broj puta u okolinu objekta, nakon čega se još dodatno skaliraju i rotiraju oko pozicije jedne mrlje iz podskupa. Os rotacije se izabire za svaku kopiranu instancu podskupa mrlja nasumično.

a)

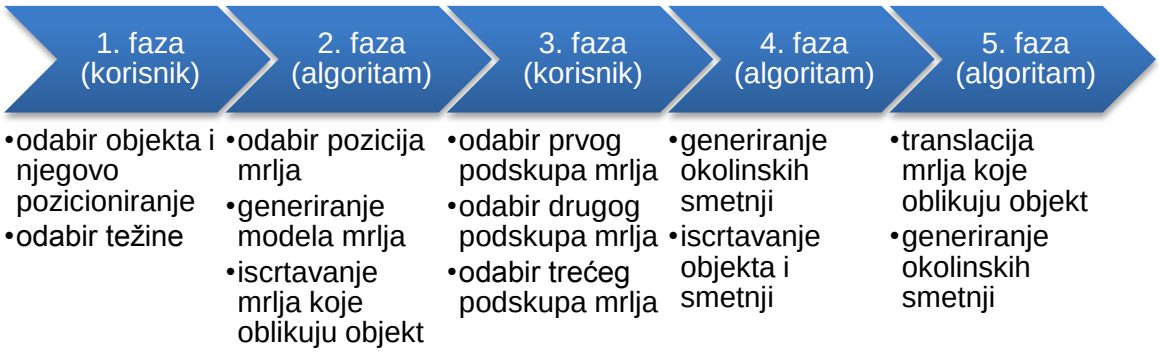
b)

c)

Slika 18. Tri veličine prozora za odabir podskupova mrlja, a) mali prozor, b) srednji prozor, c) veliki prozor

Prije nego se kopirani podskupovi mrlja dodaju na listu mrlja koje tvore okolinske smetnje potrebno je provjeriti da li se one preklapaju s nekom od mrlja koje iscrtavaju objekt. Provjeravanje preklapanja se obavlja pomoću detekcije kolizije kao i kod odabira podskupova mrlja. Svaki model mrlje okolinske smetnje ima svoju kružnu obujmicu za koju se provjerava da li je u koliziji s križnom obujmicom modela. Ako niti jedna mrlja iz podskupa nije u koliziji s obujmicom modela taj podskup dodaje se na listu okolinskim smetnji. U slučaju da je neka od mrlja iz podskupa u koliziji s obujmicom modela provjerava se kolizija svake mrlje podskupa sa svakom mrljom iz skupa onih koje iscrtavaju objekt. Ako nema kolizije niti jednog modela mrlje podskupa mrlja ta kopirana instanca podskupa mrlja dodaje se na listu mrlja koje tvore okolinske smetnje, u protivnome kopirana instanca podskupa mrlja se odbacuje kao neispravna skupina mrlja okolinskih smetnji. Kao što je već rečeno, broj kopiranih instanci mrlja okolinskih smetnji kao i njihova perturbacija ovise o odabranoj težini. Kada je generiranje okolinskih smetnji izvršeno jedino što preostaje je iscrtavanje mrlja koje oblikuju objekt i mrlja koje tvore okolinske smetnje. Ovime je dobivena prva slika sekvence. Ostale slike, od druge slike pa na dalje, dobivaju se translacijom mrlja koje tvore objekt na novu

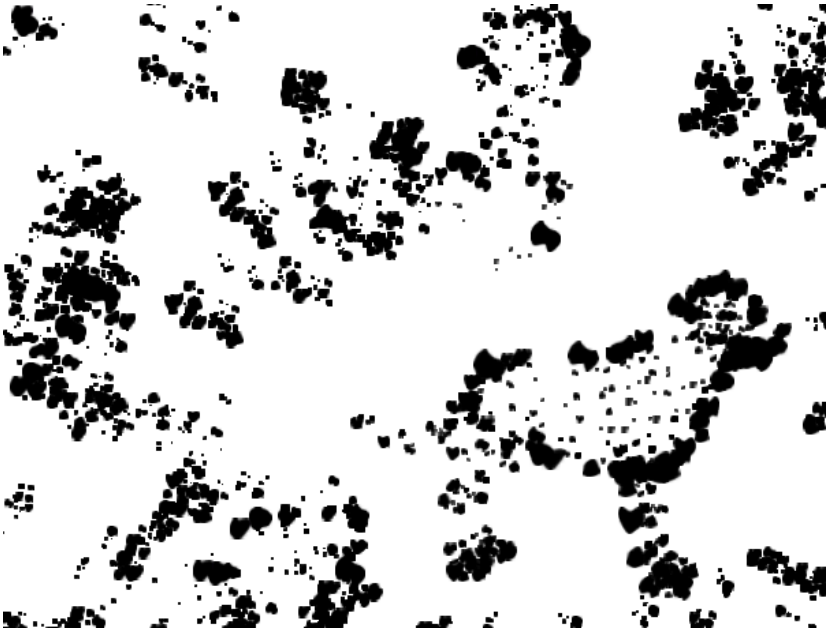
lokaciju i ponovnim generiranjem okolinskih smetnji korištenjem tri već postojeća podskupa mrlja, tj. jednom kada se generiraju modeli mrlja (za objekt i za tri podskupa mrlja od koji se generaju okolinske smetnje) oni ostaju konstantni za cijeli tijek izvođenja programa, jedino im se mijenja položaj i u slučaju okolinskih smetnji veličina i rotacija. Tijek izvođenja programa za generiranje sekvenci izranjajućih slika prikazan je na slici 19. Zadnja peta faza se neprestano ponavlja i time tvori sekvencu izranjajućih slika.



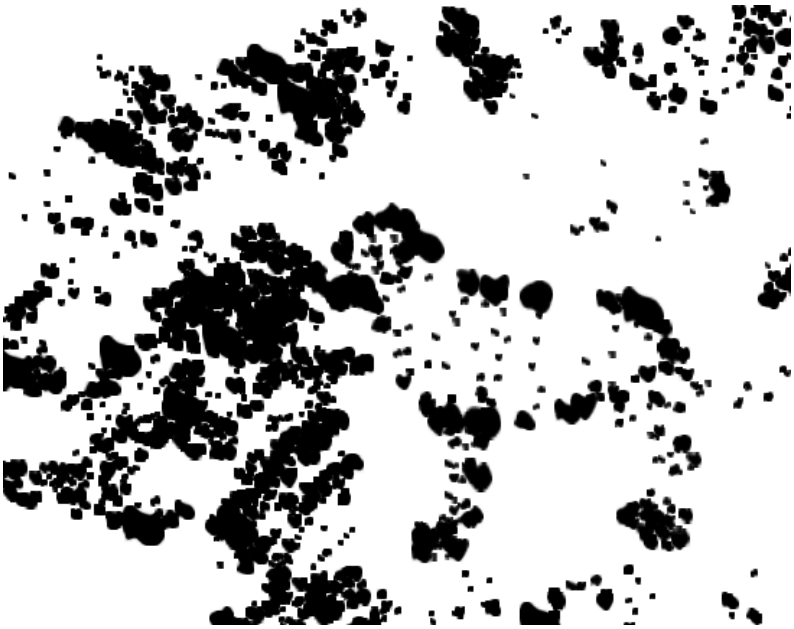
Slika 119. Tijek izvođenja programa

6. Rezultati

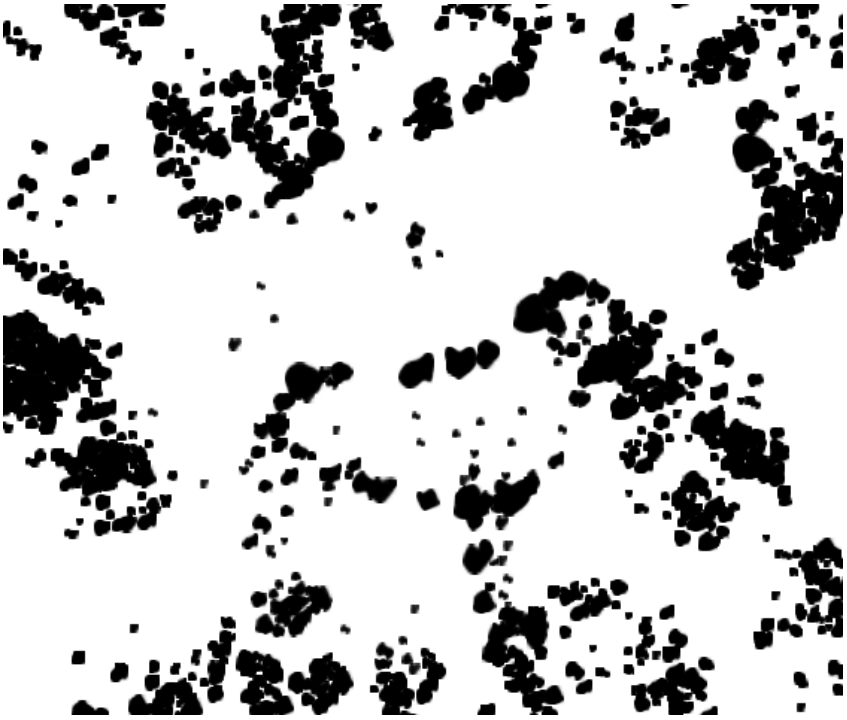
Izranjajuće slike dobivene opisanim postupkom sinteze za sve tri moguće težine prikazane su na slici 20.



a)



b)

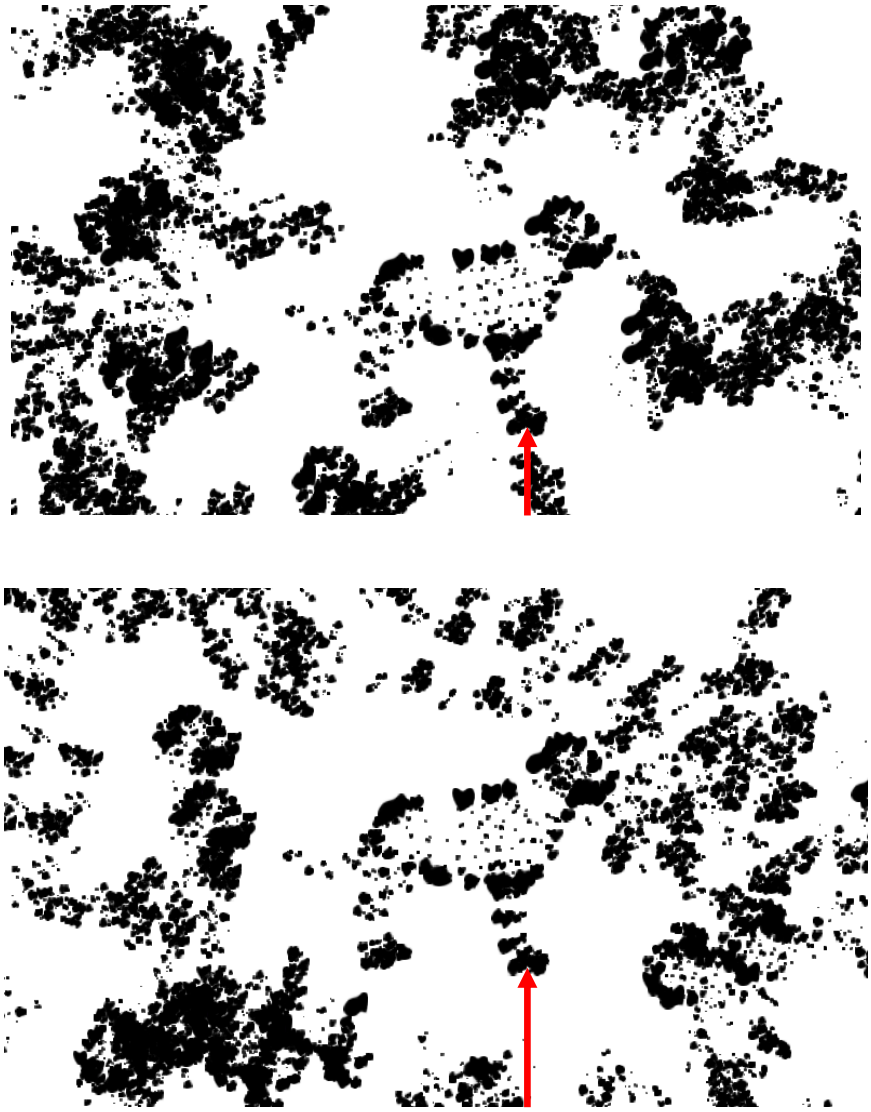


c)

Slika 20. Generirane izranjajuće slike u tri moguće težine, a) lagano, b) srednje, c) teško

Na slikama je vidljivo kako s težinom opada broj mrlja koje oblikuju objekt i kako raste broj mrlja koje čine okolinske smetnje. Upravo zbog toga je vrlo važno da objekt bude nešto što je korisniku poznato i lako prepoznatljivo, te u poziciji koja je karakteristična za taj objekt.

Generiranje sekvenci izranjajućih slika napravljeno je da bi se korisniku olakšalo prepoznavanje izranjajućeg objekta na slici. Sekvence olakšavaju prepoznavanje izranjajućeg objekta upravo zato što je objekt (iscrtan mrljama koje prate njegovu siluetu) uvijek isti nakon što završi postupak generiranja mrlja koje ga iscrtavaju, dok se pozadina mijenja, tj. iako se objekt kreće u sceni njegov izgled je statičan što ljudima olakšava prepoznavanje. Nekoliko izranjajućih slika sekvence prikazano je na slici 21. Kako se objekt na generiranim sekvencama kreće u smijeru gore-dolje, na slike su dodane strelice od ruba slike do pozicije objekta kako bi se lakše moglo utvrditi njegovo kretanje.



Slika 21. Nekoliko slika iz generirane sekvence izranjajućih slika

Iz slike 21 također je vidljivo da objekt iscrtan mrljama ostaje isti tijekom cijele sekvence samo se na svakoj slici nalazi na drugoj lokaciji, dok se okolinske smetnje za svaku sliku ponovno generiraju iz 3 podskupa mrlja koji sadrže uvijek iste modele mrlja za svaku sliku. Prilikom kretanja objekt se samo translatira pošto bi rotiranje u određenim trenucima rezultiralo položajem koji nije karakterističan za objekt, a kako je već rečeno važno je da se objekt nalazi u nekom od karakterističnih položaja kako bi se korisniku olakšalo prepoznavanje. Drugi načini prihvatljivih pokreta bi bili skaliranje i/ili animacija samog objekta na način da mu se dijelovi pomiču (npr. pas koji trči). Implementirano je samo kretanje

translacijom, ali zbog strukture programske implementacije u budućnosti je lagano dodati funkcije koje obavljaju neki drugi način kretanja objekta.

6.1. Testiranje rezultata

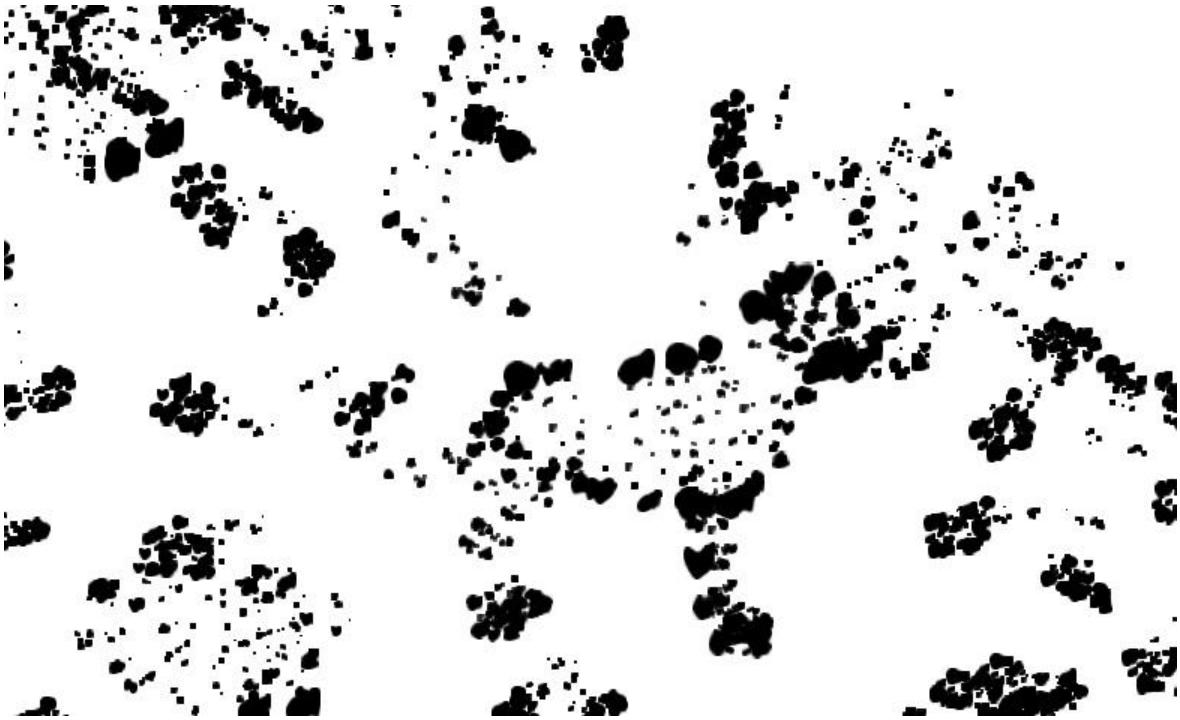
Prilikom definiranja postupka sinteze izranjajućih slika definirana su dva pravila koja izranjajuće slike moraju poštivati:

- čovjek mora moći prepoznati objekt na generiranim slikama
- automatizirani program nesmije moći išta prepoznati na generiranim slikama

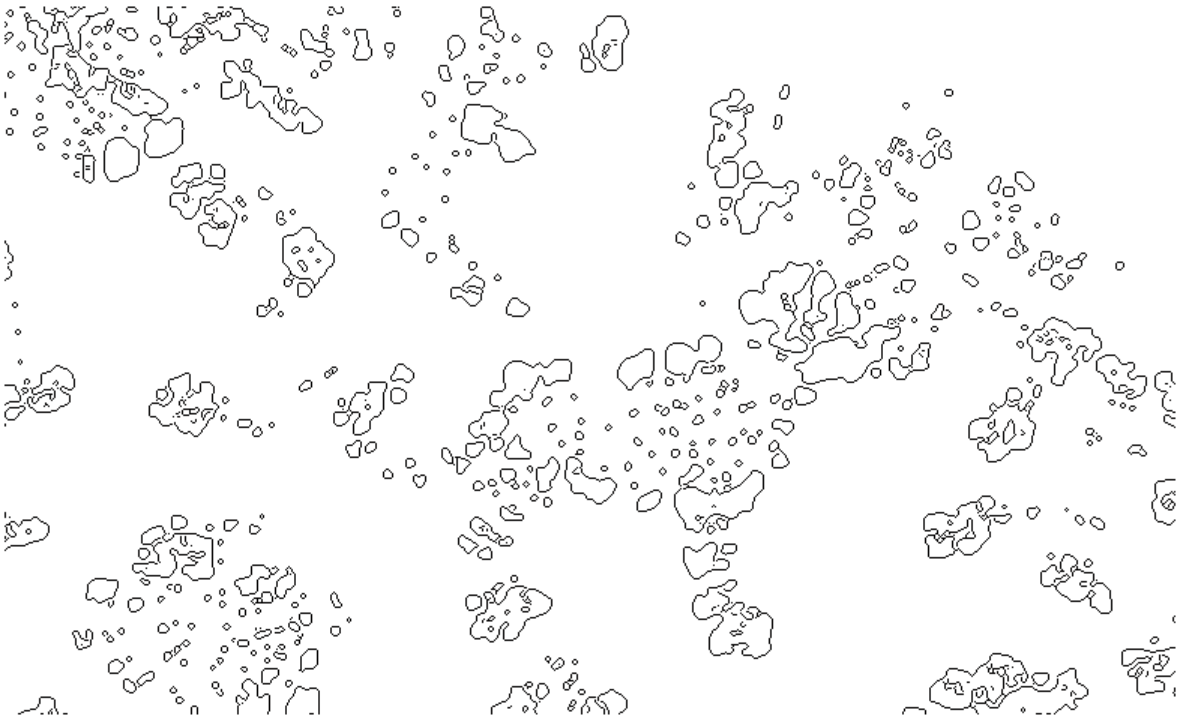
Prvo pravilo je zadovoljeno, tj. čovjek na slikama može prepoznati da se radi o psu. Drugo pravilo potrebno je provjeriti pomoću sredstava koje upotrebljava automatizirani program, a to su algoritmi za raspoznavanje objekata na slikama i sekvencama. Za testiranje dobivenih rezultata odabran je Canny algoritam za detekciju rubova i algoritam tekućeg prosjeka za detekciju kretanja na sekvencama izranjajućih slika. Testiranje je obavljeno za izranjajuće slike generirane na najlakšoj razini težine zato što ona sadrži najviše informacija o objektu, pa ako su rezultati tog testiranja zadovoljavajući trebaju biti i za druge dvije razine težine pošto one sadrže manje informacija o objektu. Detekcija rubova izabrana je zato što značajno umanjuje količinu informacija i izbacuje nepotrebne informacije, a zadržava važne strukturalne informacije slike. Osim toga, algoritmi za detekciju rubova vrlo su često prvi korak složenijih algoritama računalnog vida.

6.1.1. Canny algoritam

Canny algoritam raspoznavanja rubova poznat je kao i optimalan algoritam za raspoznavanje rubova na slici i sastoji se od nekoliko koraka u kojima se npr. uklanjaju smetnje na koje je algoritam jako osjetljiv, određuju usmjerenja rubova itd. Testiranje ovim algoritmom dalo je dobre rezultate (slika 22) pošto su jedini rubovi koje je algoritam pronašao rubovi mrlja ili u slučaju preklapanja više mrlja rub mrlje nastale preklapanjem. Iako je ljudima objekt na slici još donekle vidljiv, automatiziranom programu dobiveni podaci ne otkrivaju ništa o objektu.



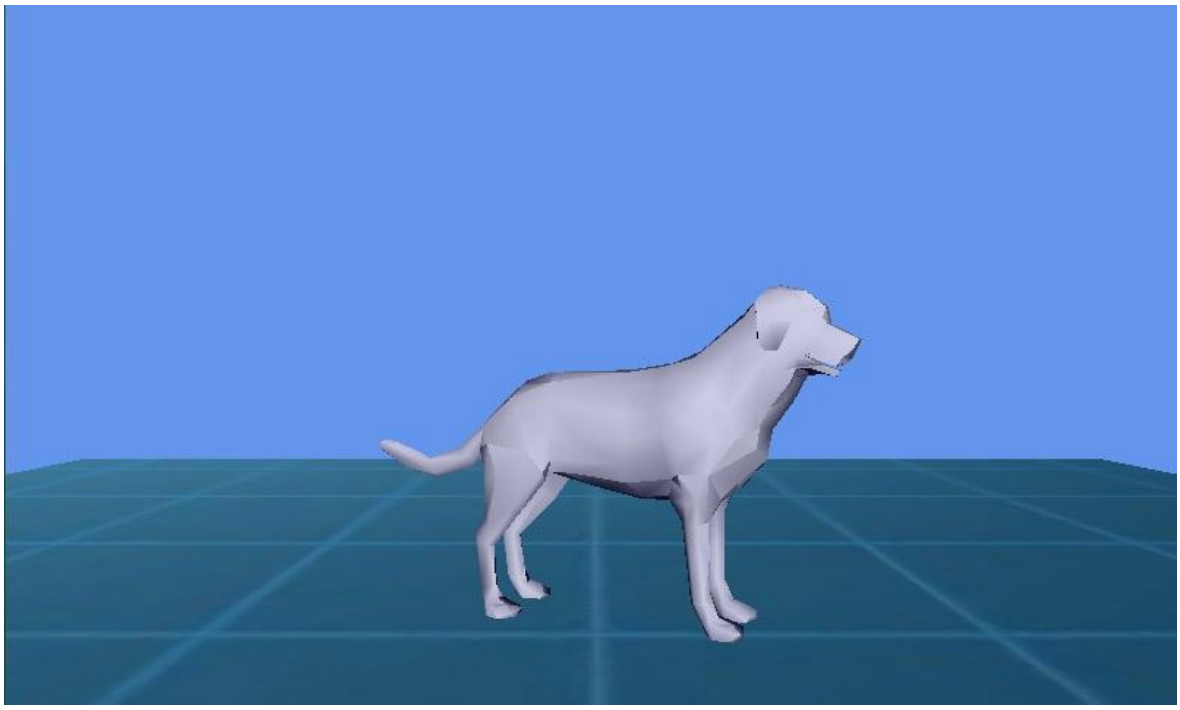
a)



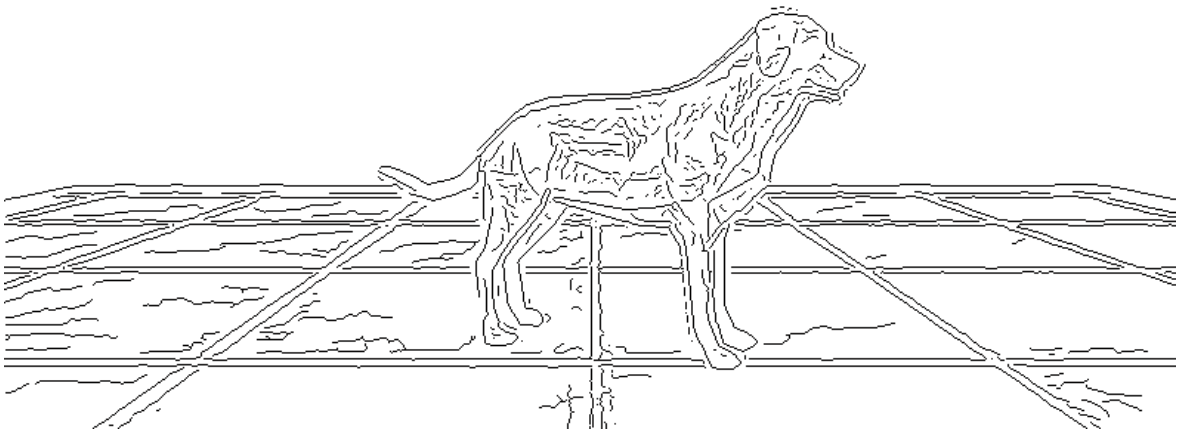
b)

Slika 22. Primjena Canny algortima na izranjajućoj slici, a) originalna slika, b) slika dobivena algorimom

Slika 23 prikazuje sliku početne scene, dakle sliku objekta i njegove okoline prije postupka sinteze izranjajuće slike, i sliku dobivenu izvođenjem Canny algoritma na toj slici. Usporedbom dvije slike dobivene Canny algoritmom vidljivo je kako je algoritam na slici početne scene vrlo dobro našao rubove i samim time siluetu lika, dok to nije slučaj na slici dobivenoj izvođenjem na izranjajućoj slici.



a)



b)

Slika 23. Primjena Canny algortima na slici početne scene, a) originalna slika, b) slika dobivena algorimom

6.1.2. Detekcija pokreta

Detekcija pokreta implementirana je u programu MATLAB. Implementirani algoritam je algoritam tekućeg prosjeka (engl. Running average). Na slici 24 prikazan je prednji plan (engl. foreground) jedne slike sekvence dobiven izvođenjem algoritma tekućeg prosjeka na sekvenci izranjajućih slika. Iz slike vidljivo je da algoritam ima dojam da se veliki dio slike kreće upravo zbog toga što su smetnje u okolini objekta vrlo dinamične, tj. mijenjaju se sa svakom novom slikom u sekvenci. Bijela područja na slici označavaju mjesta na kojima algoritam detektira kretanje. Čovjeku je objekt još uvijek djelomično vidljiv na slici zbog njegovog sporog kretanja, dok automatizirani program ne može iz ove slike detektirati objekt koji se kreće.

Slika 15. Prednji plan algoritma tekućeg prosjeka za izranjranjajuću sliku

7. Zaključak

U ovome radu opisan je implementirani postupak sinteze izranjajućih slika. Prilikom izrade programske implementacije bilo je potrebno voditi računa o dva pravila koja moraju vrijediti za izranjajuće slike. Prvo pravilo govori da na generiranim slikama čovjek mora moći prepoznati izranjajući objekt, dok drugo pravilo govori da automatizirani program ne smije moći prepoznati izranjajući objekt na generiranim slikama. Iz dobivenih rezultata jasno je vidljivo da je prvo pravilo zadovoljeno. Testiranjem Canny algoritmom za detekciju rubova i algoritmom za detekciju pokreta pokazano je da je zadovoljeno i drugo pravilo. Također, vrlo je važno da je izranjajući objekt poznat promatraču kao i da se nalazi u njemu karakterističnoj pozi. U protivnome moglo bi se dogoditi da promatrač ne percipira izranjajući objekt. Sekvence izranjajućih slika pomažu čovjeku da percipira izranjajući objekt pošto on ostaje isti (oblikom isti, ali se kreće) na svim slikama sekvcence dok se okolina mijenja. Zbog toga bilo je potrebno testirati i detekciju kretanja objekta na sekvencama. Upotrijebljen je algoritam tekućeg prosjeka iz čijih se rezultata može zaključiti da algoritam nije u stanju izdvojiti objekt na slikama. Algoritam može samo reći da li se neki dio slike kreće ili ne, a kako su cijela okolina objekta i sam objekt dinamični algoritam detektira kretanje na većini slike, tj. kretanje okoline maskira kretanje objekta.

Ovakav postupak generiranja izranjajućih slika vrlo je siguran čak i u slučaju kada automatizirani program pozna algoritam koji generira slike upravo zbog činjenice da ne zna vrijednosti parametara i izgled scene.

Osiguravanje izranjajućim slikama protiv automatiziranih programa za sada je još nepraktično zato što bi za njegovo korištenje trebale postojati velike baze imena objekta na svim jezicima web stranica na kojima se upotrebljava. Ponuđeni odgovori nisu opcija pošto u tom slučaju automatizirani program može izabrati npr. odgovor a i nastaviti izabirati taj odgovor sve dok odgovor zaista i ne bude a.

8. Literatura

1. Wikipedia contributors, Gestalt psychology, Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/wiki/Gestalt_psychology
2. Niloy J. Mitra, Hung-Kuo Chu, Tong-Yee Lee, Lior Wolf, Hezy Yeshurun, Daniel Cohen-Or, Emerging Images, ACM Transactions on Graphics, svezak 28, broj 5 (2009)
3. Lehar S., The World In Your Head, Lawrence Erlbaum Associates, Mahwah, 2003.
4. Charles Humphrey, Basic HLSL lighting tehniques, 11.11.2008., *Basic HLSL lighting tehniques – Episode 1: Ambient and Diffuse*, http://xna-uk.net/blogs/randomchaos/archive/2008/11/11/basic-hlsl-lighting-techniques-episode-1-ambient-and-diffuse.aspx, 25.3.2011.
5. Charles Humphrey, Basic HLSL lighting tehniques, 11.11.2008., *Basic HLSL lighting tehniques – Episode 2: Coloured*, *Specular &Multiple Lights* http://xna-uk.net/blogs/randomchaos/archive/2008/11/12/basic-hlsl-lighting-techniques-episode-2-coloured-specular-amp-multiple-lights.aspx, 25.3.2011.
6. Riemer Grootjans, Automaticali generating normals, http://www.riemers.net/eng/Tutorials/XNA/Csharp/ShortTuts/Normal_generation.php, 1.4.2011.
7. Shawn Hargreaves, Vertex Data in XNA Studio 4.0, 19.4.2010., *Game programming with the XNA Framework,* http://blogs.msdn.com/b/shawnhar/archive/2010/04/19/vertex-data-in-xna-game-studio-4-0.aspx, 5.4.2011.
8. Ron Visbord, Haim Nachmias, Dimitry Apter, EI Captcha – Emerging Images Captcha, http://www.cs.tau.ac.il/~haimnach/eicaptcha/index.php, 15.4.2011.
9. XNA Shader Programming – Tutorial 1, Ambient Light, 23.3.2010., *Xna Shader Programming*, http://digitalerr0r.wordpress.com/2009/03/23/xna-shader-programming-tutorial-1-ambient-light/, 25.3.2011.

10. XNA Shader Programming – Tutorial 2, Diffuse Light, 23.3.2010., *Xna Shader Programming*, http://digitalerr0r.wordpress.com/2009/03/23/xna-shader-programming-tutorial-2-diffuse-light/, 25.3.2011.
11. Gestalt Laws, *Collection of Optical Illusions* http://www.sapdesignguild.org/resources/optical_illusions/gestalt_laws.html, 22.5.2011.

9. Sažetak

(Postupak sinteze izranjajućih slika)

Porastom računalne moći postaje sve teže osmisliti načine zaštite web servisa od iskorištavanja pomoću programa za automatizirano raspoznavanje. Jedan od mogućih načina zaštite su upravo izranjajuće slike koje se temelje na pravilu izranjanja gestalt psihologije. Pravilo izranjanja govori da čovjek može percipirati objekte na slici prepoznajući ih kao cjelinu umjesto kao skup dijelova objekta. Slijedeći to pravilo postupak sinteze izranjajućih slika implementiran u ovom radu uzima model nekog čovjeku dobro poznatog objekta, postavlja ga u neku karakterističnih poza, iscrtava ga sastavljenog od mrlja i zatim od podskupova tih mrlja koje iscrtavaju objekt stvara okolinske smetnje. Ovakav način sinteze osigurava da čovjek na slici može prepoznati izranjajući objekt, a automatizirani program korištenjem raznih algoritama nemože prepoznati objekt. Implementirani postupak sinteze izranjajućih slika omogućava generiranje beskonačnog broja različitih izranjajućih slika što automatiziranom programu dodatno otežava prepoznavanje objekta na izranjajućoj slici.

Ključne riječi: gestalt psihologija, izranjanje, izranjajuće slike, postupak sinteze

Abstract

(Synthesis technique of Emerging Images)

Because of the fast growth of computation power it has become increasingly difficult to protect web services from exploitation by automated software. Emerging images represent one of the possible methods of protection. They are based on one of the pinciples of Gestalt psychology, emergence. The principle of emergence tells us that humans can percieve objects in an image as a whole and not by recognizing the object parts. The method of synthesizing emerging images implemented in this thesis follows the principle of emergence by taking a well know object, placing it in a recognizable position, rendering the object using splats and generates surrounding clutter from subsections of splats that follow the object silhoutte. This technique for generating emerging images allows us to generate infinite number of different emerging images making it harder for automated software to recognize the emerging object on the image.

Key words: gestalt psychology, emergence, emerging images, synthesis technique