Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

David Novak

Zajem 3D predmetov s Kinectom

diplomsko delo

univerzitetni strokovni študijski program prve stopnje računalništvo in informatika

Mentor: doc. dr. Matija Marolt

Ljubljana 2012

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00016/2012
Datum: 02.04.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **DAVID NOVAK**

Naslov: **ZAJEM 3D PREDMETOV S KINECTOM**
3D SCANNING WITH KINECT

Vrsta naloge: Diplomsko delo univerzitetnega študija prve stopnje

Tematika naloge:

V diplomskem delu izdelajte sistem za zajem 3D predmetov s pomočjo senzorja Kinect. Preučite in primerjajte obstoječe rešitve. Seznanite se s senzorjem Kinect in njegovim načinom delovanja, ter z algoritmi, ki so potrebni za uspešno izgradnjo poligonske mreže iz zajetih globinskih slik. Izdelajte sistem za zajem 3D predmetov in analizirajte njegove lasnosti.

Mentor:

doc. dr. Matija Marolt

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Namesto te strani **vstavite** original izdane teme diplomskega dela s podpisom mentorja in dekana ter žigom fakultete, ki ga diplomant dvigne v študentskem referatu, preden odda izdelek v vezavo! Glej tudi sam konec Poglavja 2 na strani **??**.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani David Novak, z vpisno številko **63090102**, sem avtor diplomskega dela z naslovom:

Zajem 3D predmetov s Kinectom

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Matije Marolta,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 17. septembra 2012 Podpis avtorja:

Zahvaljujem se mentorju mojega diplomskega dela, docentu, dr. Matiji Maroltu, za strokovno vodenje pri izdelavi naloge.

Zahvaljujem se lektorici, Barbari Udrih, za strokovno opravljeno lektoriranje diplomskega dela.

Kazalo

Povzetek

To delo opisuje postopke in tehnologije, ki se uporabljajo pri zajemu tridimenzionalnih predmetov z uporabo optičnega čitalca Kinect. Za boljše razumevanje področja so bila preučena teoretična ozadja osnov računalniške grafike, projektivne geometrije, optike in grafične rekonstrukcije. Del vsebine je namenjen tudi delovanju in zgradbi Kinecta, predstavlja pa teoretično podlago za implementacijo lastnega ogodja za zajem tridimenzionalnih predmetov. Ogrodje je sprogramirano z upoštevanjem smernic, ki jih podajajo drugi razvijalci na tem področju. V ta namen so bile preučene tudi njihove programske rešitve in pristopi k reševanju problema zajema tridimenzionalnih predmetov in njihove rekonstrukcije. Omenjene so tudi ideje za nadaljnji razvoj rešitve problema.

Abstract

This paper describes the procedures and technologies used for scanning and three-dimensional reconstruction of objects using the optical scanner Kinect. Theoretical backgrounds of basic computer graphics, projective geometry, optics and graphical reconstruction were studied for better understanding of this field of computer science. A part of the content also describes the structure and operating of the Kinect and is used as a theoretical basis for implementing a new framework for three-dimensional object scanning. Methods and procedures of other developers were taken into account, when implementing the framework. Therefore their solutions and approaches to solving the problem were studied, to achieve better results. Described are also the ideas for further improvement of the solution.

Poglavje 1

Uvod

Možnost shranjevanja podatkov v digitalni obliki prinaša v današnjem času veliko prednost. Shranjevanje, obdelava in predstavljanje podatkov s pomočjo računalnika človeku prihrani ogromno časa ter nezaželjenega in nepotrebnega dela. Prihod tridimenzionalnih optičnih čitalnikov pa je omogočil tudi shranjevanje opisov oblike okolice in predmetov. Te naprave se široko uporabljajo v zabavni industriji, kjer danes produkcija računalniških iger in filmov narekuje vedno izrazitejšo uporabo digitalnih učinkov in tudi računalniško ustvarjenih in obdelanih entitet. Tovrstna tehnologija pa se ne uporablja zgolj na področju zabave, temveč tudi na področjih medicine (natančneje prostetike in ortotike), arhitekture, industrije in mnogih drugih. Zmožnost ustvarjanja navideznega modela in posledično tudi simulacije interakcije z modelom ima tolikšno uporabno vrednost, da je postala v modernem svetu skoraj nepogrešljiva. Med drugim omogoča tudi iskanje drobnih napak v gradnjah (razpoke in podobne nepravilnosti) in izobraževalne simulacije na mnogih področjih.

Rekonstrukcija okolice in predmetov s pomočjo optičnih čitalnikov pa ni edini pristop k ustvarjanju navideznih modelov. Na mnogih strokovnih področjih podobne rezultate dosegajo kar z 'ročnim' modeliranjem, ki ga omogočajo programi, kot sta Autodesk Maya in Autodesk 3ds Max. Čeprav je ta način pogosto časovno zelo potraten, ga odlikuje kakovost končnega

rezultata. Ta je namreč točno tak, kot si ga je zamislil oblikovalec, saj nanj ne vplivajo moteči dejavniki okolja. Upoštevati pa moramo tudi, da so ti grafični programi pogosto zelo dragi, prav tako pa potrebuje oblikovalec nekaj časa, da se jih nauči uporabljati. Nekoliko lažji in bolj avtomatiziran način rekonstrukcije se imenuje fotogrametrija. Gre za nekoliko starejši postopek, bistvo katerega je določitev geometrijskih značilnosti ciljnega predmeta ali okolja z uporabo fotografij. Razdaljo med točkama na ravnini, vzporedni s slikovno ravnino, lahko izračunamo z merjenjem razdalje med slikovnima pikama, ki jih predstavljata. Upoštevati moramo samo ustrezen koeficient, ki pove razmerje med zajeto sliko in dejanskim prizoriščem. Algoritmi, ki se uporabljajo v te namene, rešujejo predvsem problem zmanjševanja množice napak, ki se med postopkom pojavijo.

Čeprav so vsi pristopi k reševanju problema gradnje navideznih modelov zanimivi in vredni podrobnejšega pregleda, se to diplomsko delo osredotoča na zajem oblik z uporabo optičnih čitalcev. Teh je veliko vrst, ločimo jih po načinu delovanja, velikosti in cenah. Najbolj osnovna delitev razvršča čitalce glede na to, ali podatke o obliki pridobivajo na dotik ali na daljavo. Tako ločimo kontaktne (*contact*) in nekontaktne (*non-contact*), slednje pa lahko delimo naprej na aktivne in pasivne. Ključni del kontaktnih čitalcev so občutljive ročice, ki drsijo po površju predmeta in shranjujejo vrednosti, ki jih zaznajo. Ročice so lahko zgrajene na več načinov in se zato tudi uporabljajo v različnih primerih. Nekatere so toge in zato bolj primerne za površja z malo podrobnostmi, nekatere pa so sestavljene iz množice povezanih delov, za katere se neprestano računata rotacijo in kot v sklepih. S temi podatki se skozi zapletene matematične funkcije lahko pridobi uporabne informacije o površju. Nekontaktni čitalci obliko zajemajo s pomočjo valovanja, ki ga usmerijo proti prizorišču, nato pa zaznavajo njegov odboj. Valovanj je več vrst, tipično pa se za potrebe računalniške grafike uporabljajo rentgenski žarki, ultrazvok ali svetloba.

Te naprave imajo veliko pozitivnih lastnosti, nedvomno pa tudi nekaj negativnih. Še najbolj izstopa njihova cena. Ta lahko znaša od 30 € do

130.000 €, zares zmogljive in občutljive naprave pa dosežejo tudi 300.000 €. Ob takih vsotah večji del amaterskih in polprofesionalnih razvijalcev gotovo izgubi zanimanje. Na voljo imamo sicer tudi manjše ročne čitalce, vendar v tehnologiji cena z manjšanjem dimenzij pogosto le raste. S prihodom naprav, kot je optični čitalec Kinect, pa so se tudi prej omenjenim skupinam odprle nove poti v to vejo računalništva. Nesmiselno je pričakovati, da bi tovrstne naprave iz neprimerljivo nižjega cenovnega ranga[1] ponujale popolnoma enakovredne funkcionalnosti kot profesionalna znanstvena oprema, nudijo pa več kot dovolj za razvoj uporabnih aplikacij, tako za tridimenzionalno skeniranje kot tudi za marsikaj drugega.

Zdi se edino smotrno, da izkoristimo zmožnosti te nove tehnologije. Če lahko z uporabo Kinecta dosežemo prav tako zgledne rezultate pri zajemu podatkov o oblikah, potem je to zelo dober nadomestek za drago tehnološko opremo, ki si je ne more vsak privoščiti. To delo opisuje upravljanje s Kinectom, ki nudi rešitev problema tridimenzionalnega skeniranja predmetov in okolice. Cilj diplomske naloge je preučiti Kinect, njegov način delovanja in funkcionalnosti ter algoritme, ki so potrebni za uspešno izgradnjo mreže (*mesh*) iz vhodnih podatkov. Drugo poglavje dela sestavlja teoretično ozadje problema, s kratkimi opisi algoritmov, matematičnih postopkov in funkcij, ki se na tem področju uporabljajo. Sledi poglavje, ki opisuje že obstoječe rešitve, ki se lotevajo tega problema, končni rezultat dela pa predstavlja implementacija lastnega ogrodja za zajem tridimenzionalnih predmetov. Za zaključek pa so navedene še nadaljnje izboljšave dela, ki v okviru te diplomske naloge niso bile izvedene.

[1] Cene Kinecta se gibljejo med 120 € in 330 €.

Poglavje 2

Teoretično ozadje

2.1 Microsoft Kinect za XBOX 360

Orodje, s katerim je bila opravljena večina meritev, opisanih v tem diplomskem delu, se imenuje Microsoft Kinect za XBOX 360 [17]. Gre za optični čitalec podjetja Microsoft, uporablja pa se predvsem kot vnosna naprava pri igranju iger priljubljene, prav tako Microsoftove, igralne konzole XBOX 360. Mehanika omenjenih iger je zasnovana tako, da igralec dogajanje vodi z glasovnimi ukazi in telesnimi gibi, ki jih Kinect ustrezno prevede v vhodne podatke. Za uspešno delovanje takega sistema je zato potrebna učinkovita implementacija zaznavanja globine ter razčlenjevanja govora. To napravi med drugim omogoča prepoznavanje obrazov, govora in tudi snemanje video posnetkov.

2.1.1 Zgradba Kinecta

Kinect sestavlja pet glavnih delov: projektor infrardečih žarkov, dve kameri (infrardeča in barvna), nabor mikrofonov in motor, namenjen nagibanju kamer ter projektorja.

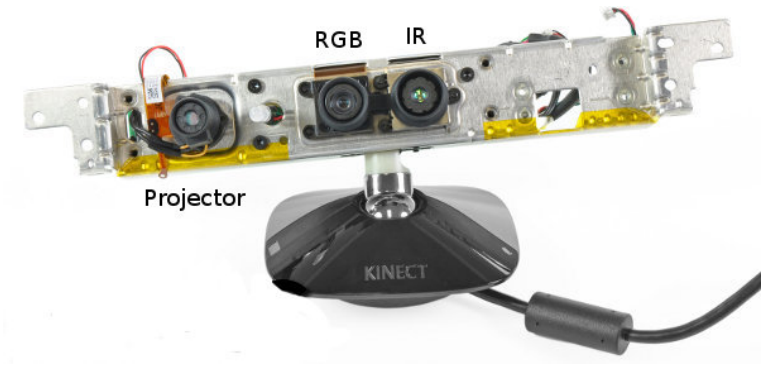
Naloga projektorja (Slika 2.2 levo) je metanje žarkov infrardeče svetlobe na okolico. Žarke lahko z uporabo ustrezne opreme (na primer kamera z možnostjo nočnega snemanja) vidimo kot majhne svetlobne pike (Slika 2.3),

Slika 2.1: Microsoft Kinect za XBOX 360.

razporejene po prostoru. Metanje svetlobe je izvedeno na podoben način kot pri laserju, kar preprečuje preveliko razširitev svetlobnih pik na bolj oddaljenih površinah. Kljub temu pa občasno pride do popačenja teh pik, še posebej takrat, ko prizorišče sestavljajo elementi, zaradi katerih se v sliki pojavijo veliki in nenadni prehodi med nivoji globin[1]. Pogost rezultat tega je, da nekaj žarkov dobi več kot en podatek o globini. Vendar to ni nujno slabo, saj nam v nekaterih primerih omogoča izračun večih prostorskih koordinat ali celo oblike predmetov. Takšni podatki se uporabljajo v tako imenovanih algoritmih sestavljene svetlobe. Gre za postopke, v okviru katerih ugotavljamo obliko predmeta ali postavitev teh v okolici, s preslikovanjem znanega vzorca slikovnih pik na prizorišče. Iz nove lege teh pik potem lahko izračunamo vse podatke, ki so potrebni za uspešno izgradnjo navideznega modela, vključno z globino. Do neke mere so ti algoritmi uporabljeni tudi pri Kinectu.

Z infrardečo kamero (Slika 2.2 desno) Kinect zaznava infrardečo svetlobo v okolju. To napravi omogoča določanje globine v posameznih delih slike. Za zajemanje video posnetkov in slik uporablja Kinect običajno barvno kamero (Slika 2.2 na sredini). Obe kameri omogočata ločljivost posnetkov do 640 ×

[1]Primer: imamo predmet v ospredju in steno v ozadju, žarek pa pade točno na rob predmeta.



Slika 2.2: Kinect brez plastičnega ohišja.

480 slikovnih pik, s frekvenco osveževanja 30 Hz. Kot že omenjeno, čitalec podpira tudi glasovne ukaze, ki jih sprejema preko nabora mikrofonov.

2.1.2 Delovanje Kinecta

Kinect uporablja poseben način določanja položaja točk v prostoru, imenovan stereo triangulacija (*stereo triangulation*). Običajno se ta postopek izvaja s pomočjo dveh kamer, v tem primeru pa je nekoliko drugače. Vendar kljub razlikam za izračun globine posamezne točke v obeh primerih potrebujemo dve sliki:

- Prva slika je posnetek, ki ga zajame infrardeča kamera. Gre za sliko množice po prostoru razporejenih svetlobnih pik, ki jih tvori infrardeči projektor. Vsaka pika predstavlja eno točko. Vsaka točka x se skozi gorišče O kamerine leče preslika v svojo slikovno piko (*pixel*) y_1 na ravnini posnetka. Če skozi točki y_1 in O potegnemo premico, bo ta premica nekje v prostoru potovala skozi točko x.

- Če bi ta postopek opravljali z dvema običajnima kamerama, potem bi z drugo kamero storili podobno kot s prvo. Posneli bi sliko prizora iz nekoliko drugačnega zornega kota[2] in določili drugo premico, ki bi se

[2]Običajno bi drugo kamero le premaknili vstran, vzporedno glede na prvo. Bistveno

s prvo sekala v prostoru točno tam, kjer se nahaja točka x. Vendar je pri Kinectu postopek nekoliko drugačen, saj druga slika uporabniku ni vidna. Žarki projektorja tvorijo namreč naključen, vendar ne enoten vzorec, ki je določen ob proizvodnji naprave. Narejen je s pomočjo filtra, ki svetlobo razprši. To Kinectu omogoča, da primerja vzorce manjših skupin projeciranih točk s celotnim vzorcem, ki ga je vgradil proizvajalec. Razlog za uporabo delno naključne razporeditve tu postane očiten, saj močno poenostavi iskanje ujemanj. Vsaka pika ima namreč edinstveno okolico sosednjih pik, zato jo lahko naprava lažje prepozna. Podobnosti med množicama naprava izračuna s pomočjo korelacije. Korelacija je statistična mera, ki izraža stopnjo linearne povezanosti med dvema spremenljivkama oziroma odvisnosti med dvema populacijama. Na ta način za vsako točko x določi, kje v projektorjevem vzorcu je njen izvor y_2.

Ti sliki sta podobni, vendar nista enaki, saj je med projektorjem in infrardečo kamero razmak (približno 7,5 centimetrov). Tako infrardeča kamera, projektor in točka v prostoru tvorijo trikotnik, kar napravi s pomočjo pridobljenih podatkov omogoča izračun globine z uporabo triangulacije. Triangulacija je matematični postopek, ki se uporablja v trigonometriji in geometriji. Cilj postopka je določitev oddaljenosti neke točke z merjenjem kotov v dveh znanih točkah, ki se nahajata na skupni premici. Ko določimo kota med premico in ciljno točko, je izračun z uporabo sinusnega izreka trivialen, saj razdaljo med točkama na premici že poznamo.

Kinect neobdelane podatke o globini vrne v binarni obliki. Dva bajta ali šestnajst bitov je potrebnih za zapis informacije, ki nam pove globino posamezne točke v milimetrih in številko zaznanega igralca, saj poleg računanja triangulacije Kinect sproti ugotavlja tudi, ali posamezna točka pripada igralcu. Tako vsakemu zaznanemu igralcu dodeli število, ki ga shrani v spodnjih treh bitih šestnajstbitnega rezultata, medtem ko v zgornjih trinajst bitov shrani razdaljo med projektorjem in vsako točko. Upoštevati moramo tudi, da je

je, da se obe kameri nahajata na skupni premici.

Slika 2.3: Pike infrardeče svetlobe, razporejene po prostoru.

učinkovito delovanje naprave odvisno od oddaljenosti predmetov od sprejemnika. V ta namen so proizvajalci novejšim različicam optičnih čitalcev Kinect dodali dva načina delovanja: tako imenovani *Near mode* ali bližji način in *Far mode* ali oddaljeni način. *Near mode* je namenjen sprejemanju podatkov zelo blizu kamere, *Far mode* pa se uporablja pri zajemanju podatkov na večjih[3] razdaljah. Pri obeh možnostih lahko glede na oddaljenost od kamere določimo štiri območja:

- Neznano območje je predel, kjer naprava uporabniku ne more vrniti uporabnih rezultatov. Globinske vrednosti tega dela so definirane kot neznane, pogosto pa so označene z -1.

- Območje, ki je v Tabeli 2.1 označeno kot 'preblizu', je definirano samo za *Far mode* način uporabe in določa, kdaj je objekt preblizu kamere.

- Kar se nahaja v območju običajnih vrednosti, naprava zajame in nato izračuna ustrezne globinske podatke.

- Podobno, kot če je predmet preblizu, naprava določi, kdaj je predmet predaleč. To so predmeti, ki se nahajajo v območju, ki je v Tabeli 2.1 označeno z besedo 'predaleč'.

Način delovanja	Neznano	Preblizu	Običajne vrednosti	Predaleč
Near mode	0 - 0,4 in od 8 naprej	0,4 - 0,8	0,8 - 4	4 - 8
Far mode	0 - 0,4 in od 8 naprej	/	0,4 - 3	3 - 8

Tabela 2.1: Načina delovanja Kinecta z določenimi mejami delovanja. Podane vrednosti so izražene v metrih.

Različica Kinecta, ki je bila uporabljena pri snovanju te diplomske naloge, deluje samo v oddaljenem načinu, vendar to ni bistveno vplivalo na razvoj ogrodja za rekonstrukcijo. Pri določanju prostorskih koordinat menim, da je še najbolj razumno, da globine, ki ne padejo v območje običajnih vrednosti, popolnoma izločimo. Veliko število neuporabnih podatkov lahko namreč končni rezultat pokvari ali pa negativno vpliva na hitrost delovanja.

2.2 Rekonstrukcija podatkov

Rekonstrukcija je v računalniški grafiki postopek gradnje digitalne predstavitve določene entitete. Ta je lahko del resničnega sveta, ni pa nujno. To diplomsko delo se osredotoča na rekonstrukcijo resničnih predmetov, s pomočjo optičnega čitalca Kinect. Preden se lahko lotimo gradnje navideznih modelov, moramo razumeti, kako sploh pridemo do podatkov, ki nam to omogočajo. Ko s kamero zajemamo slike, govorimo o točkah v dveh koordinatnih sistemih. Prvi je koordinatni sistem sveta, ki ga želimo zajeti s kamero, drugi pa je koordinatni sistem znotraj kamere (ravnina slike). Sistema povezuje matematična funkcija, imenovana preslikava. Ta narekuje, kako se točka iz tridimenzionalnega prostora prenese v dvodimenzionalni prostor in obratno. Preden se lahko lotimo rekonstrukcije, moramo najprej izračunati

[3]Relativno glede na Kinectovo območje delovanja.

točke v prostoru in ustrezno kalibrirati kamere.

2.2.1 Izračun točk

Preslikave lahko obravnavamo tudi kot transformacije. Transformacije predstavljajo enega izmed ključnih elementov računalniške grafike. Z njimi v grafiki namreč pretvorimo eno množico točk v drugo. Obravnavamo jih kot matematične funkcije oziroma preslikave, ki vsako točko preslikajo v neko drugo točko, v skladu z določenimi pravili. Poznamo transformacije v dvodimenzionalnem in tridimenzionalnem prostoru. Slednje se uporabljajo tudi v tridimenzionalni rekonstrukciji:

- Translacija (Enačba 2.1) je transformacija, ki jo najlažje opišemo kot togi premik točke. Z matematičnega vidika gre za prištevanje odmika trenutnim koordinatam.

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} \tag{2.1}$$

- Rotacijo (Enačbe od 2.2 do 2.5) določamo v odvisnosti od števila koordinatnih osi. V prostoru s tremi dimenzijami imamo tri osi, kar omogoča vrtenje predmeta v treh smereh: okoli osi x, okoli osi y in okoli osi z za kot θ. Rotacijo predmeta lahko opišemo tudi kot rotacijo koordinatnega sistema v nasprotni smeri. Če izračunamo skalarni produkt med vektorji pred transformacijo in vektorji po transformaciji, dobimo koordinate točke v novem sistemu.

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = R_{x,y,z}(\theta) \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.2}$$

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.3}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.4}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.5}$$

- Skaliranje (Enačba 2.6) je odvisno od faktorjev s_x, s_y in s_z. Če so faktorji enaki, govorimo o enakomernem, v nasprotnem primeru pa o neenakomernem skaliranju. Skaliranje še najlažje opišemo kot spreminjanje velikosti obravnavanega objekta.

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \begin{bmatrix} x \\ y \\ dz \end{bmatrix} \tag{2.6}$$

Kot lahko vidimo, vse transformacije matematično niso predstavljene z enakimi operacijami. Lahko rečemo tudi, da so operacije heterogene. Računanje je bistveno lažje, če so operacije homogene, kar pomeni, da za združevanje preslikav uporabljamo samo en matematični operator. V tem primeru je očitna izbira operacija množenja, saj sta dve od treh omenjenih transformacij predstavljeni z množenjem. To dosežemo z uvedbo homogenih koordinat. Homogene koordinate omogočajo, da z množenjem matrik poljubno združujemo transformacije. Postopek je preprost, saj samo uvedemo novo koordinato za zapis točk. Pri točkah ima ta koordinata vrednost 1, pri vektorjih pa 0. Funkcije transformacij tako dobijo novo matematično podobo. Poglejmo si novo enačbo za izraz transformacije:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.7}$$

Kljub homogenim operacijam moramo biti še vedno pozorni na njihov vrstni red. Množenje matrik ni komutativno. Vzemimo za primer predmet, katerega središče se nahaja v koordinatnem izhodišču. Če ta predmet rotiramo in nato transliramo, ne bomo dobili enakega rezultata, kot če predmet transliramo in nato rotiramo. Izid rotacije je vezan na koordinatno izhodišče, ki pa v obeh primerih ni na enakem mestu[4]. V vrsti transformacij je prva tista, ki se v enačbi nahaja skrajno desno.

Kot že omenjeno, je tudi preslikava točk iz tridimenzionalnega prostora na dvodimenzionalno ravnino slike transformacija. Kako ta preslikava poteka, določa projekcijska matrika P (Enačba 2.8).

$$P = K\,[R|t] \tag{2.8}$$

Sestavljena je iz tako imenovanih zunanjih in notranjih parametrov kamere. Zunanji parametri kamere so zapisani v sestavljeni matriki $[R|t]$. R imenujemo rotacijska matrika kamere, t pa translacijski vektor. Govorimo o dveh algebraičnih konstruktih, ki opisujeta gibanje kamere okoli negibnega prizorišča ali obratno. Koordinatna sistema kamere in predmeta zanimanja (ali okolice) sta drug glede na drugega translirana in zarotirana. Z uporabo zunanjih parametrov lahko opravljamo preslikave med obema. Obstaja tudi tretji primer, ko se položaja obeh elementov ne spreminjata. Takrat uporaba zunanjih parametrov ni potrebna.

Matrika K (Enačba 2.9) je matrika notranjih parametrov kamere.

$$K = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

Sestavljajo jo trije ključni elementi: goriščni razdalji kamere, označeni z f_x in f_y, osnovna točka, označena s c_x in c_y, in koeficient γ. Obe meri sta podani v številu slikovnih pik. Goriščna razdalja je izraz, ki v optiki poimenuje

[4]Glede na zorni kot predmeta.

pomembno lastnost leč. Gre za mero, ki nam pove, kako močno leča zbira oziroma razprši svetlobo. V fizikalnem pogledu gre za razdaljo, ki jo prepotuje snop kolinearnih[5] svetlobnih žarkov, preden se zberejo v skupni točki. Krajša kot je razdalja, večja je optična moč leče. To pomeni, da lahko svetlobne žarki ukrivi hitreje. Vsak optični sistem ima optično os, ki predstavlja središče rotacijske simetrije tega sistema. Glavna točka je točka, v kateri glavna ravnina seka optično os. Za vsako lečo lahko določimo dve glavni ravnini: sprednjo in zadnjo. Njuna poglavitna lastnost je ta, da se pri opazovanju žarka, ki potuje iz leče, zdi, da je žarek sekal sprednjo ravnino na enaki oddaljenosti od optične osi kot zadnjo. Tako lahko trdimo, da do lomljenja žarkov pride na glavnih ravninah. Glavna točka je pomembna informacija, ko določamo lastnosti optičnega sistema, saj je od nje odvisna stopnja povečevanja. Koeficent γ imenujemo tudi koeficient ukrivljenosti med osjo x in osjo y in mu v izračunih pogosto pripisujemo vrednost 0. Matrika notranjih parametrov kamere je kot del projekcijske matrike ključnega pomena v izračunu. Medtem ko se zunanji parametri stalno spreminjajo, so notranji vedno enaki. Ko jih enkrat določimo, jih lahko uporabimo večkrat. Upoštevati moramo samo to, da so ti parametri izračunani za določeno ločljivost slike, kakršna je bila uporabljena ob kalibraciji. Če spremenimo ločljivost slike, moramo postopek ponoviti, ker postanejo prejšnji izračuni neveljavni. Izračunamo jih s postopkom, ki ga imenujemo kalibracija kamere.

2.2.2 Kalibracija kamer

Obstaja več različnih načinov kalibracije, rezultati vseh pa so glavna točka, goriščni razdalji in nabor koeficientov k, ki opisujejo motnje, ki jih povzroča leča. Motnje so z oddaljenostjo od optične osi vedno izrazitejše. Kalibracija kamer Kinecta, uporabljenega pri pisanju te diplomske naloge, je bila opravljena s pomočjo programskega orodja Camera Calibration Toolbox for Matlab. Postopek obsega tri korake:

[5]Kolinearni žarki so vzporedni, zato se s potovanjem svetlobe njihova smer v idealnih pogojih ne spreminja, v realnem okolju pa so odstopanja zelo majhna.

1. Zajem slik: Uporabnik mora pred uporabo orodja s kamero, ki jo želi kalibrirati, posneti približno 20 slik šahovnice iz različnih zornih kotov. Če uporabljamo Kinect, moramo postopek izvesti za obe kameri, barvno in infrardečo. Čeprav infrardeča kamera zaznava le infrardečo svetlobo, lahko v tem primeru posnamemo tako imenovan neobdelan (*raw*) infrardeči posnetek, ki je za ta postopek bolj primeren. Za dosego tega moramo prekriti projektor žarkov in poskrbeti za dovolj močno količino infrardeče svetlobe v prostoru.

2. Označevanje kotov: Ko program zaženemo, se slike shranijo v računalnikov pomnilnik. Nato sledi ročno označevanje kotov šahovnice na vsaki sliki. Ko kote označimo, nam program na podlagi oznak poudari kote vsakega izmed črnih oziroma belih polj na šahovnici. Število polj lahko uporabnik vnese ročno, na voljo pa je tudi algoritem, ki to število predlaga. Če so koti označeni pravilno, lahko s to sliko zaključimo, lahko pa predlagamo koeficient motnje (*distortion coefficient*), na podlagi katerega program izriše nove oznake, če prejšnje niso bile ustrezne.

3. Izračun parametrov: V zadnjem koraku programsko orodje izračuna notranje parametre kamere in predlaga koeficiente motenj. Če menimo, da rezultati niso ustrezni, potem celoten postopek ponovimo.

Čeprav je z uporabnikovega vidika postopek zaključen v treh korakih, je celoten algoritem, ki teče v ozadju, nekoliko bolj obsežen. Če želimo njegovo delovanje razumeti, se moramo najprej poglobiti v teorijo homografske transformacije. To je transformacija, ki predstavlja projekcijo ene ravnine v prostoru na drugo ravnino v prostoru. Poglejmo si koordinatni sistem sveta, ki ga zajemamo s kamero. Predpostavimo lahko, da se ravnina ciljnega modela (*model plane*) nahaja v koordinatnem sistemu sveta, kjer je $z = 0$. Tako lahko točko na tej ravnini v homogenih kordinatah zapišemo kot $P = [x, y, 1]^T$, saj je koordinata z enaka nič. Točko v svetovnih koordinatah $P = [X, Y, Z]^T$ in točko na slikovni ravnini $p = [u, v]^T$ povezuje matrika

velikosti 3×3, ki jo imenujemo homografija H. Če velja, da $z = 0$, jo lahko opišemo s sledečo enačbo:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} r_1 r_2 r_3 t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = K \begin{bmatrix} r_1 r_2 t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = H \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{2.10}$$

Če homografijo H (Enačba 2.10) zapišemo v stolpcih, jo lahko izrazimo kot:

$$\begin{bmatrix} h_1 h_2 h_3 \end{bmatrix} = \lambda K \begin{bmatrix} r_1 r_2 t \end{bmatrix} \tag{2.11}$$

Z upoštevanjem, da sta stolpična vektorja r_1 in r_2 ortonormalna, izpeljemo dve omejitvi, ki veljata za homografijo:

$$h_1^T K^{-T} K^{-1} h_2 = 0 \tag{2.12}$$

$$h_1^T K^{-T} K^{-1} h_1 = h_2^T K^{-T} K^{-1} h_2 \tag{2.13}$$

Na podlagi teh ugotovitev lahko pričnemo z izračunom notranjih parametrov kamere. Ta se prične z določitvijo matrike B,

$$B = K - T K^{-1} \equiv \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix} \tag{2.14}$$

ki je simetrična in definirana s šestdimenzionalnim vektorjem b (Enačba 2.15).

$$b = \begin{bmatrix} B_{11} B_{12} B_{22} B_{13} B_{23} B_{33} \end{bmatrix} \tag{2.15}$$

Če posodobimo Enačbi 2.12 in 2.13, nas to pripelje do Enačbe 2.16, kjer je h_i i-ti stolpični vektor homografije, v pa je definiran kot $v_{ij} = [h_{i1} h_{j1}, h_{i1} h_{j2} + h_{i2} h_{j1}, h_{i2} h_{j2}, h_{i3} h_{j1} + h_{i1} h_{j3}, h_{i3} h_{j2}, h_{i2} h_{j3}, h_{i3} h_{j3}]$.

$$h_i^T B h_j = v_{ij}^T b \tag{2.16}$$

Omejitvi homografije iz Enačbe 2.12 lahko potem zapišemo kot homogeni enačbi

$$\begin{bmatrix} v_{12}T \\ (v_{11} - v_{22})^T \end{bmatrix} b = 0 \tag{2.17}$$

ali kar cel sistem enačb $Vb = 0$, če opazujemo več slik ravnine modela. Ko enkrat določimo vrednosti vektorja b, lahko potem določimo tudi notranje parametre kamere. Te se izračuna s pomočjo Enačb od 2.18 do 2.23.

$$c_y = \frac{(B_{12}B_{13} - B_{11}B_{23})}{(B_{11} - B_{22} - B_{12}^2)} \tag{2.18}$$

$$\lambda = \frac{B_{33} - [B_{13}^2 + c_y(B_{12}B_{13} - B_{11}B_{23})]}{B_{11}} \tag{2.19}$$

$$f_x = \frac{\sqrt{\lambda}}{B_{11}} \tag{2.20}$$

$$f_y = \frac{\sqrt{\lambda B_{11}}}{(B_{11}B_{22} - B_{12}^2)} \tag{2.21}$$

$$\gamma = \frac{-B_{12}f_x^2 f_y}{\lambda} \tag{2.22}$$

$$c_x = \frac{\gamma c_y}{f_x} - \frac{B_{13}f_x^2}{\lambda} \tag{2.23}$$

Celoten kalibracijski postopek tako obsega 6 korakov:

1. Natisnemo vzorec in ga nalepimo na ravno podlago.
2. Zajamemo več posnetkov vzorca iz različnih zornih kotov, s premikanjem ravnine ali kamere.
3. Označimo ali detektiramo značilne točke na posnetkih (tipično skrajne točke vzorca ali ravnine).
4. Ocenimo notranje parametre.

5. Ocenimo koeficiente motenj.

6. Izboljšamo kvaliteto izračunanih vrednosti.

Algoritem, uporabljen v programu Calibration Toolbox, je hiter, natančen in enostaven za uporabo. Premikamo lahko tako ravnino kot kamero, prav tako pa značilnosti premika niso pomembne.

Ogrodje, razvito v okviru tega diplomskega dela, predpostavlja, da sta tako okolica kot kamera negibni. Ta predpostavka v enačbi odpravi potrebo po zunanjih parametrih. Koordinate točke v tridimenzionalnem prostoru lahko potem izračunamo po sledeči enačbi:

$$v_i(u) = D_i(u)K^{-1}u \tag{2.24}$$

Enačbo 2.24 lahko drugače zapišemo tudi kot:

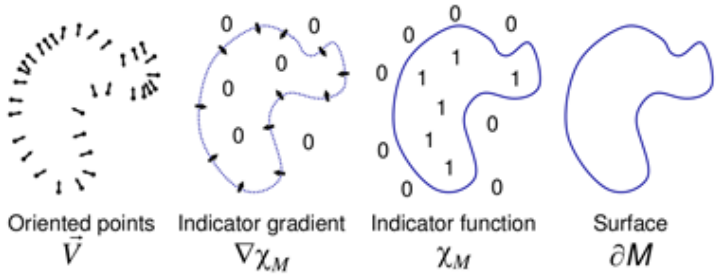
$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = D_i(u) \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.25}$$

$D_i(u)$ je vrednost, ki predstavlja globino, izmerjeno na območju slikovne pike u, K^{-1} pa je inverzna matrika matrike notranjih parametrov K. V enačbi v_i predstavlja posamezno vozlišče v množici. Ko za vsako vozlišče izračunamo njegove koordinate, jih lahko povežemo v celotno mrežo s pomočjo postopka, imenovanega Poissonova rekonstrukcija površja.

2.2.3 Poissonova rekonstrukcija površja

Poissonova rekonstrukcija površja [5] je algoritem, ki se uporablja za gradnjo mreže navideznega modela. Postopek se začne z zapisom tako imenovane tridimenzionalne implicitne pokazateljske funkcije (*indicator function*) χ. Ta je izražena tako, da ima na območju izven površja vrednost nič, na območju znotraj površja pa ena.

Med množico izračunanih točk (glej 2.2.1 Izračun točk) in implicitno funkcijo χ obstaja razmerje. Če ga želimo razložiti, moramo najprej razumeti kaj



Slika 2.4: Koncept Poissonove rekonstrukcije.

je gradient. Gradient funkcije je diferencialna operacija. Njen namen je za polje vektorjev ali skalarjev, določiti v kateri smeri se to polje najbolj spreminja. Gradient označimo z matematičnim simbolom ∇ (včasih tudi $\vec{\nabla}$), ki ga imenujemo nabla. Gradient implicitne funkcije χ je polje vektorjev, ki je skoraj povsod enako nič, saj se funkcija večinoma ne spreminja. Izjema so točke blizu površja, kjer je polje enako, navznoter obrnjeni normali površja. Množico izračunanih točk tako lahko obravnavamo, kot vzorec gradienta implicitne funkcije modela. Glavni problem tako postane poiskati inverzno operacijo gradientu, oziroma ustrezno skalarno funkcijo, ki najbolj natančno opiše polje vektorjev V, določeno z množico točk. Iščemo tako funkcijo χ, da bomo dobili rezultate, ki ustrezajo sledeči formuli:

$$min|\nabla\chi - V| \tag{2.26}$$

Z uporabo divergence lahko ta problem pretvorimo v standardni Poissonov problem. Izračunati moramo funkcijo χ, katere divergenca gradienta, je enaka divergenci vektorskega polja V, oziroma:

$$\Delta\chi \equiv \nabla \cdot \nabla\chi = \nabla \cdot \chi \tag{2.27}$$

Divergenca vektorskega polja je operator, ki vektorskemu polju na nek določen način priredi skalarno polje. Ko implicitno funkcijo določimo, ta opisuje potek površja, ki smo ga iskali.

Poglavje 3

Rešitve

3.1 Obstoječe rešitve

Skupnost razvijalcev aplikacij za Kinect je z vsakim dnem večja. Posledično lahko danes na medmrežju najdemo že ogromno implementacij in algoritmov, ki rešujejo problem tridimenzionalnega skeniranja. Po pregledu nekaj teh rešitev sem izbral štiri, po mojem mnenju, najboljše izvedbe:

1. LGDV 3D Facial Scan: Gre za implementacijo rekonstrukcije obraza, obraznih mimik in potez, ki vključuje tudi sprotno teksturiranje s pomočjo zajete barvne slike.

2. ReconstructMe: Preprosta aplikacija za splošno rekonstrukcijo okolice in predmetov.

3. Matherix 3Dify: Ogrodje, ki je rezultat dela dveh računalniških inženirjev, soustanoviteljev skupine Matherix Labs.

4. 3D Reconstruction DFKI: Rekonstrukcijski algoritem nemškega centra za razvoj umetne inteligence.

3.1.1 Ocene in opisi

Za lažje razumevanje in obrazložitev ocen in mnenj so v Tabeli 3.1 podani opisi kriterijev, v skladu s katerimi je določeno, kako dobro programi rešujejo različne dimenzije problema[1]. Kriterij je osnovan izključno na mojem mnenju.

Kriterij	Opis
Prijaznost do uporabnika	Sicer ena izmed manj pomembnih dimenzij problema, vendar je kljub temu ne gre zanemariti. Z vidika razvijalca prijazen uporabniški vmesnik sicer nima tako temeljnega pomena, vendar ne smemo pozabiti, da je cilj razvoja teh programov omogočiti uporabno 3D skeniranje prav na Kinectih, kar pa sčasoma neizogibno vodi tudi v implementiranje dobrega uporabniškega vmesnika.
Natančnost skeniranja	Ta člen kriterija ne potrebuje posebne razlage, saj na tem temelji celoten problem, ki ga programi rešujejo. Sestoji iz sledečih podkategorij: natančnost rekonstrukcije mreže, uspešnost izključevanja odvečnih elementov (ozadje in tla) in združevanje slik iz različnih kotov.
Manipulacija rezultata skeniranja	Od programa se pričakuje tudi ustrezen rezultat, ki omogoča nadaljnjo obdelavo, manipulacijo in urejanje. Okolico in predmete skeniramo zato, da bi rezultat lahko uporabili, bodisi v multimedijske ali kakšne druge namene.

Tabela 3.1: Kriteriji, na osnovi katerih so ocenjene rešitve.

Algoritmično ozadje posameznih rešitev na oceno ne vpliva, ker javno tudi

[1]Ker vse dimenzije niso enako ključne za uspešno delovanje algoritmov (in predvsem zadovoljive rezultate), je bilo to upoštevano pri ocenjevanju rešitev.

ni vedno dostopno. Pri večini lahko samo sklepamo, kako delujejo algoritmi v ozadju. Kjer pa so podrobnosti o implementaciji dostopne, je poleg opisa in ocene delovanja podan tudi osnutek glavnega algoritma. Če je bilo aplikacijo mogoče preizkusiti, so dodane tudi podrobnosti o uporabniški izkušnji. Ta obsega predvsem preprostost uporabe in praktičnost. Opisi rešitev se razlikujejo, saj so odvisni od podatkov, ki so na voljo.

LGDV 3D Facial scan: LGDV 3D Facial scan je aplikacija za računalniško rekonstrukcijo človeškega obraza. Njena implementacija loči zajem podatkov in rekonstrukcijo obraza. Prvi del deluje v realnem času, saj sproti zgradi grobo predstavitev vidnega polja kamere, kar močno izboljša kakovost uporabniške izkušnje. V drugem delu sledi izgradnja modela posnetega obraza, kar programu vzame nekaj časa. Delovanje programa je sledeče:

1. Zajem podatkov: V tem koraku s pomočjo Kinectove infrardeče kamere zajamemo neobdelane podatke o globini. Sledijo naslednji koraki, s katerimi program izboljša kakovost zajetih podatkov:

   (a) Časovno glajenje (*Temporal smoothing*): Uporabijo se povprečne vrednosti globine osmih zaporednih slik (*frames*). S povprečenjem vrednosti odstranimo šum in izboljšamo splošno kvaliteto podatkov.

   (b) Filtriranje z Gaussovim filtrom: Podatke se filtrira s pomočjo Gaussovega filtra. To pomeni, da vrednost vsake slikovne pike nadomestimo z obteženim povprečjem vrednosti okoliških pik. Filter ima definirano jedro (*kernel*), ki določa, kakšen vpliv imajo posamezni sosednji piksli. Tisti bližje središču jedra na novo vrednost vplivajo močneje kot tisti bližje robu. Nova vrednost se določi s pomočjo funkcije, ki določa obtežitev:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.1}$$

(c) Prepoznavanje ključnih točk: Sledi prepoznavanje ključnih točk obraza. Sem spadajo oči, konica nosu in brada.

(d) Ločevanje obraza in telesa: Za lažjo, hitrejšo in natančnejšo obdelavo se obraz loči od ostalega telesa in morebitnih drugih elementov, ki niso predmet zanimanja. Rezultat teh korakov je 3D ogrodje obraza, ki uporabniku služi kot prikaz trenutnega vidnega polja kamere.

2. Rekonstrukcija geometrije obraza: S pomočjo določenih ključnih točk in z uporabo predpripravljenega modela človeškega obraza sledi groba rekonstrukcija. Prepripravljen model (*template mesh*) se kar najbolj prilagodi ustvarjeni mreži, nato pa se z glajenjem odpravijo še nepotrebna odstopanja. Za konec se površino še teksturira s pomočjo slike, zajete s Kinectovo barvno kamero.

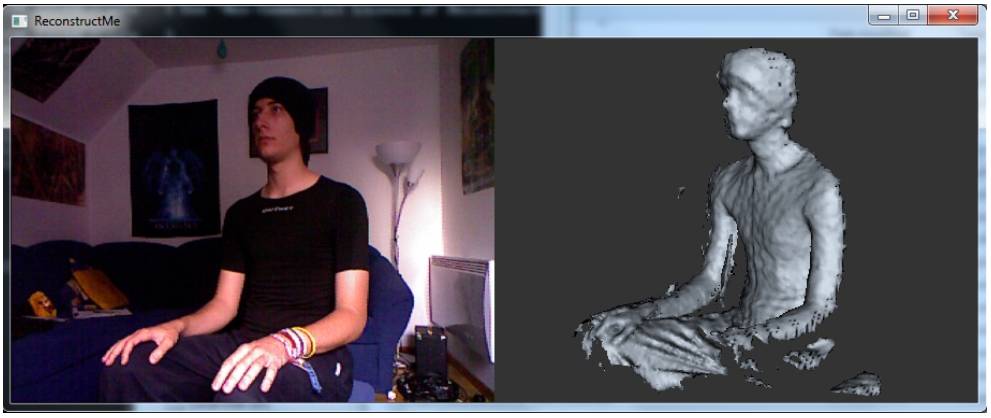
Zelo kvalitetna izvedba tridimenzionalnega skeniranja, ki me je še najbolj navdušila. Edina pomanjkljivost te rešitve je v tem, da na medmrežju nisem zasledil same aplikacije. Ocena prijaznosti uporabniškega vmesnika je tako posledično povzeta po predstavitvenih videoposnetkih na uradni strani razvijalcev. Kljub temu pa to rešitev odlikuje visoka natančnost skeniranja in zgledna stopnja manipulacije. Navidezni obraz lahko posnema preproste obrazne mimike, kot je na primer mimika govora.

ReconstructMe: Aplikacija je na voljo na uradni strani projekta in je še v razvoju, mogoča pa je prijava na testiranje. Kljub temu ima že dodan vmesnik za nameščanje. Uporabniški vmesnik aplikacije je zaenkrat narejen s pomočjo Windowsove ukazne vrstice (za zagon programa se izbere ustrezna `.bat` datoteka, odvisno, kakšen Kinect uporabljate in kaj želite z njim početi - snemati video/zvok, ustvariti mrežo). Nadaljnje ukaze se vnaša v konzolo (gre večinoma za črke na tipkovnici - p za začetek snemanja, Esc za zaključek). Program izdelane mreže shrani kot `.obj` datoteke, ki jih lahko tudi poljubno poimenujemo. V primeru, da aplikacija ne deluje pravilno

(takrat se konzola ponavadi odpre in takoj spet zapre), so nam na voljo tudi zapisi v dnevnik (*log*, shranjen v besedilnih datotekah), kjer lahko poiščemo vzrok za nepravilno delovanje.

Izgradnja se izvaja v vnaprej določenem prostoru. Privzete vrednosti tako sestavljajo kocko s stranico, dolgo en meter. S pričetkom snemanja se to območje določi relativno, glede na trenutni položaj kamere. Za kar najbolj učinkovito skeniranje moramo upoštevati nekaj omejitev, ki zadevajo spremenljive dejavnike, kot so najmanjša in največja oddaljenost od predmeta, nenadni hitri premiki in geometrija telesa. Paziti moramo, da na prizorišču ni sfer, valjev ali teles, ki jih sestavlja samo ena ravnina. Grafični model skeniranega predmeta je kvaliteten, saj moramo upoštevati tudi, da gre za rekonstrukcijo v realnem času. To uporabniku omogoča takojšnje nadaljnje delo, saj program mrežo gradi sproti. To je nedvomno ena izmed vodilnih dobrih lastnosti aplikacije. ReconstructMe je na voljo v dveh različicah: zastonj uporabniška različica in tako imenovana PRO različica, za katero je potrebno odšteti 360 €. Obe lahko uporabnik dobi na uradni strani, razlikujeta pa se tudi po naboru operacij, ki jih lahko izvajata. PRO različica v okviru te diplomske naloge ni bila preizkušena, lahko pa si na medmrežju ogledamo razne predstavitve njenih zmogljivosti. Plačljiva verzija programa omogoča tudi umestitev zgrajenega modela v virtualno okolje, ki si ga lahko ogledamo iz več zornih kotov. Kinect lahko na ukaz uporabnika tudi sledi premikom predmeta v prostoru in potem ustrezno translira in rotira model na zaslonu.

Algoritmičnega ozadja te aplikacije na medmrežju ni zaslediti, ogledamo pa si lahko več videoposnetkov, ki opisujejo njeno delovanje. Za program, ki deluje v realnem času, so rezultati več kot odlični, izvoz v datotečna formata `.obj` in `.ply` pa daje modelom veliko uporabno vrednost. Glede na to, da gre za projekt v razvoju, pa lahko v prihodnosti pričakujemo tudi nadgradnje uporabniškega vmesnika.



Slika 3.1: Končni rezultat zajema podatkov s programom ReconstructMe.

Matherix 3Dify: Matherix 3Dify uporablja podoben pristop k reševanju problema tridimenzionalne rekonstrukcije kot ostale aplikacije. Program ne deluje v realnem času. Uporabnik najprej s Kinectom posname video posnetek željenega predmeta iz vseh mogočih zornih kotov. Ta posnetek služi kot vhodni podatek rekonstrukcijskemu algoritmu, ki v kratkem času (v nekaj minutah) ustvari teksturirano mrežo. V postopku so uporabljeni običajni načini obdelave podatkov, ki ji srečamo v večini aplikacij: poravnava točk več zaporednih slik, povprečevanje globine in na koncu tudi združevanje v enoten oblak točk. Matherix 3Dify odlikujejo sledeče značilnosti:

- Natančnost: Razvijalca ogrodja obljubljata natančnost do enega milimetra.
- Preprosta uporaba: Uporabniški vmesnik je dovolj preprost za uporabo. Postopek izgradnje modela od uporabnika poleg snemanja prizorišča namreč ne zahteva veliko sodelovanja. Na voljo je začetek novega projekta, nadaljevanje obstoječega, lahko pa začnemo z grajenjem mreže iz že pripravljenega oblaka točk.
- Hitrost: Čeprav program ne deluje v realnem času, je nekaj minut še vedno zelo kratka čakalna doba za kvalitetno zgrajeno mrežo. Program naj bi imel v prihodnjih verzijah tudi način delovanja v realnem času.

Če aplikacijo želimo preizkusiti, je na uradni strani skupine Matherix Labs na voljo prijava na beta testiranje. Matherix 3Dify je ena izmed rešitev, ki si definitivno zasluži več pozornosti. Ima veliko potenciala in je v sprotnem postopku razvoja in izpopolnjevanja. Predstavlja dober primer vrste aplikacij, razvoj katerih je omogočil prihod Kinecta.

3D Reconstruction – DFKI: Ta rešitev je delo nemškega centra za razvoj umetne inteligence (*Deutsches Forschungszentrum für Künstliche Intelligenz*). Pri prejšnjih primerih je bila v ospredju aplikacija, medtem ko tu izjemoma ne govorimo o sprogramiranem uporabniškem vmesniku. Za razliko od ostalih rešitev, opisanih v tem poglavju, gre tu samo za algoritem reševanja problema. Ta sestoji iz dveh delov:

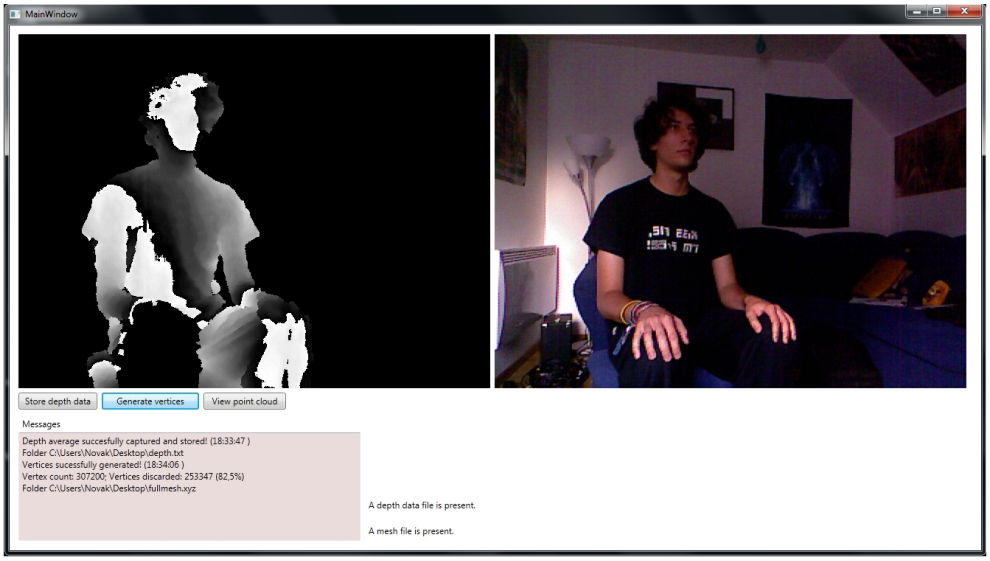
1. Zajem podatkov: Postopek se prične z zajemom globinskih podatkov. Nato se z uporabo več zaporednih slik zgradi sliko s super ločljivostjo (*super-resolution picture*). Informacijo o globini posameznih pikslov se uporabi pri računanju skupne globine.

2. Združevanje mrež (*stitching*): Mreže, generirane iz različnih zornih kotov, se združi v eno samo, z uporabo izboljšane tehnike *3D Shape Scanning with a Time-of-Flight Camera*, avtor katere je Yan Cui.

Kriteriji, navedeni v Tabeli 3.1, za opis te izvedbe sicer niso ustrezni. Ker tu ne govorimo o konkretni aplikaciji, tudi ne moremo oceniti uporabniškega vmesnika in hitrosti delovanja. Kljub temu pa si zaradi kakovosti končnih modelov, ki si jih lahko ogledamo v predstavitvenem videoposnetku, tudi ta algoritem zasluži omembo.

3.2 Lastna rešitev

Aplikacija se uporablja izključno za rekonstrukcijo predmetov in okolice. Čeprav ponuja Kinect še veliko več funkcionalnosti, s katerimi bi ogrodje lahko nadgradili, to ni cilj te naloge.



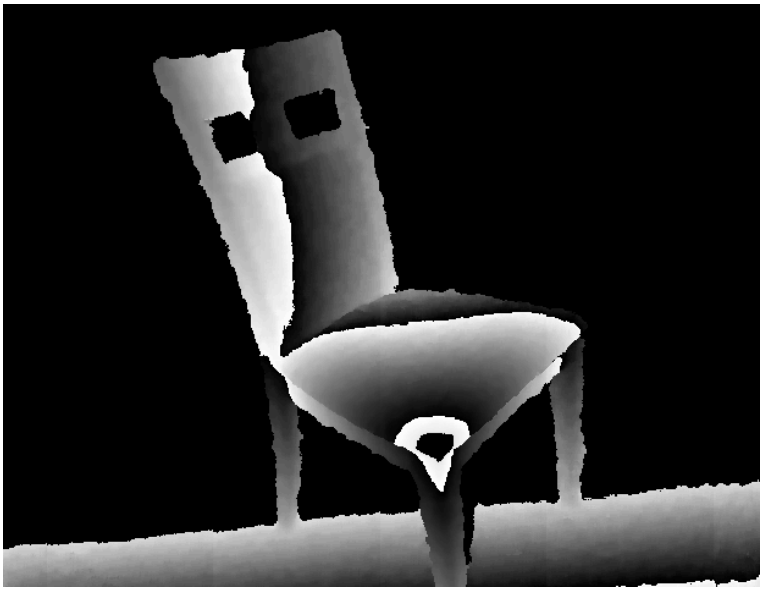
Slika 3.2: Uporabniški vmesnik ogrodja.

Največ pozornosti je bilo posvečeno zaznavanju okolice in predelavi podatkov o globini. Uporabniški vmesnik (Slika 3.2) aplikacije sestavljajo globinska slika, barvna slika, gumbi za proženje ukazov in okno z obvestili.

3.2.1 Globinska slika

V levem zgornjem kotu uporabniškega vmesnika se nahaja globinska slika (Slika 3.3). Gre za sliko, ki v širino meri 640, v višino pa 480 slikovnih pik, posodobi pa se tridesetkrat v sekundi. Njen namen je prikazati vidno polje infrardeče kamere. Za prikaz teh podatkov je bil uporabljena vrsta razreda *BitmapSource*, imenovana *WriteableBitmap*. *BitmapSource* predstavlja množico slikovnih pik z določeno velikostjo in ločljivostjo. *WriteableBitmap* je različica tega razreda, ki omogoča, da to množico lahko zapišemo in jo potem po potrebi tudi posodobimo, kar nam prihrani izdelavo tridesetih slik vsako sekundo. Preden slikovne pike lahko zapišemo, potrebujemo še podatke o globini posameznih pik ter željeni format. Format opisujeta dve značilnosti, barvni model in število bitov za zapis pike. Na voljo imamo

štiri glavne barvne modele: *RGB*(*Red, Green, Blue*), *BlackWhite*(črnobel), *CMYK*(*Cyan, Magenta, Yellow, blacK*) in *Gray*(siv). Med seboj se razlikujejo po številu kanalov, s katerimi določajo barvo posamezne slikovne pike (pri *RGB* imamo tako tri kanale, rdečega, zelenega in modrega). Po izbiri barvnega modela sledi še določitev števila bitov, s katerimi opisujemo barve. To je običajno vrednost z intervala od 2 do 32 bitov (kjer so vse vrednosti potence števila 2). V tem ogrodju je uporabljen format `Gray.8`, kar pomeni da imamo pri vsaki slikovni piki na razpolago 2^8 ali 256 odtenkov sive.



Slika 3.3: Globinska slika stola.

Podatke o globini posameznih pik dobimo s pomočjo razreda *DepthImageBuffer*, ki za vsako sliko (*frame*) shrani podatke o globini, ki jih posreduje infrardeča kamera. Kot že omenjeno v opisu delovanja Kinecta, so ti podatki shranjeni v binarni obliki, natančneje v šestnajstih bitih. S premikom vrednosti za 3 bite v desno dobimo razdaljo v milimetrih, ki se potem shrani v enodimenzionalno polje slikovnih pik. Da bi bila slika kar najbolj pregledna, je globinska slika razdeljena na tri dele. Prvi del predstavljajo slikovne pike z globino, manjšo od 900 milimetrov, drugi del pike z globino med 900 in

2000 milimetrov, tretji del pa vse pike z globino, večjo od 2000 milimetrov. Pri rekonstrukciji se namreč uporabljajo samo podatki iz drugega dela slike, zato je samo v tem delu uporabljen sivinski odtenek. V ostalih dveh delih je slika črna. S pritiskom na gumb ogrodje omogoča zapis podatkov o globini v besedilno datoteko.

3.2.2 Barvna slika

Slika meri 640 slikovnih pik v širino in 480 slikovnih pik v višino, posodablja pa se tridesetkrat na sekundo. V tej aplikaciji ni bistvenega pomena, služi bolj kot prikaz vidnega polja Kinectove barvne kamere. To uporabniku pomaga pri prepoznavanju predmetov v globinski sliki, saj iz te na prvi pogled ni vedno mogoče razbrati vseh predmetov. Kljub temu pa ima barvna slika določeno uporabno vrednost pri rekonstrukciji entitet. Slikovne pike, ki jih posname barvna kamera, vsebujejo podatke o teksturi predmetov. To nam omogoča, da poleg podatkov o zgradbi in obliki zajamemo tudi podatke o barvi predmetov. Rezultat tega je teksturirana mreža. Teksturiranje v ogrodju ni implementirano.

3.2.3 Obvestilno okno

Ob izvajanju ukazov nas ogrodje obvešča o trenutnem stanju in končnih rezultatih metod, ki izvajajo izračune algoritmov. Vnosi v besedilno okno niso mogoči, saj le-to služi izključno kot orodje za spremljanje napredka in iskanje napak v delovanju aplikacije.

3.2.4 Ukazi

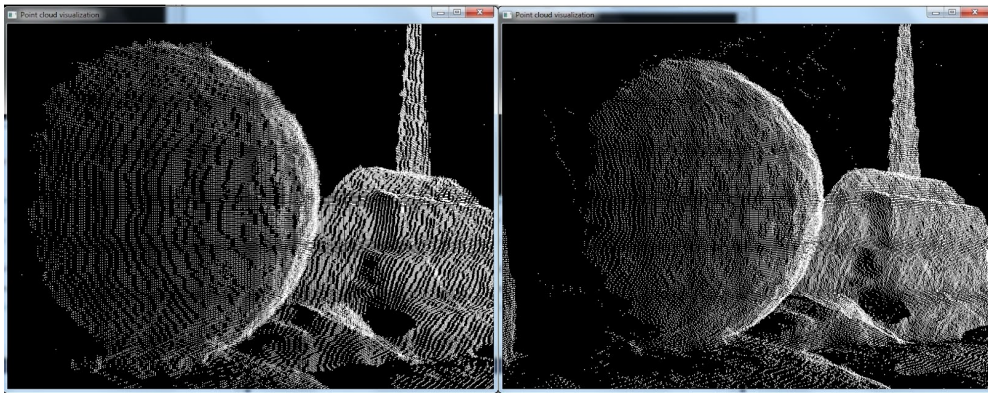
V aplikaciji imamo na voljo več ukazov, ki jih programu posredujemo s pritiskom na ustrezne gumbe:

- Capture depth: S pritiskom na ta gumb program ustvari besedilno datoteko, v katero shrani podatke o sliki, ki je trenutno prikazana na

zaslonu. V oknu z obvestili se nato pojavi poročilo o uspešnosti izvajanja tega ukaza. Za izboljšanje kvalitete podatkov program pred zapisom izračuna povprečno vrednost globin desetih zaporednih slik. Rezultat tega so bistveno boljši podatki, kot lahko vidimo na Sliki 3.4.

- Generate vertices: Ko smo globinske podatke zajeli, nam je na voljo nov ukaz, ki omogoča, da iz teh podatkov zgradimo mrežo. V primeru, da program še ni ustvaril ustrezne datoteke, nam to vmesnik sporoči z ustreznim besedilom, ki se izpiše poleg obvestilnega okna. Določitev vozlišč (*vertex* oziroma *vertices* v množini) in pripadajočih normal poteka v štirih korakih, opisanih v 3.2.5 Gradnja mreže.

- View point cloud: Po zapisu vozlišč in normal si lahko s pritiskom na gumb 'View point cloud' ogledamo tudi prikaz oblaka točk. Ta služi kot vmesni prikaz rezultatov, da uporabnik dobi vtis, kako bo izgledala končna mreža.



Slika 3.4: Prikaz oblaka točk za eno globinsko sliko (levo) in za povprečje desetih slik (desno).

3.2.5 Gradnja mreže

Branje in filtriranje globinskih podatkov: Iz datoteke, ki je bila ustvarjena s pritiskom na gumb za zajem globine, se preberejo vsi podatki. Prva prebrana vrednost predstavlja širino, druga pa višino slike, iz katere so zajeti ti podatki. Obe meri sta izraženi s številom slikovnih pik. Program nato ustvari prazno dvodimenzionalno polje, ki po velikosti ustreza meram slike. Po zapisu globinskih podatkov v polje sledi filtriranje. Namen tega je zmanjšati pretirane odklone v vrednostih, ki so posledica šuma in ostalih motečih dejavnikov. Za glajenje imamo na voljo več izbir, ki bi lahko bile primerne za rešitev tega problema, vendar sem se po premisleku odločil za uporabo filtra, osnovanega na mediani okoliških slikovnih pik. Mediana je statistična mera, ki označuje srednjo vrednost populacije. Bistvo filtra je v tem, da vsaki slikovni piki določi novo vrednost, ki je odvisna od vrednosti okoliških pik. Velikost okolice je lahko poljubna, določiti pa jo moramo pred računanjem. Algoritem, ki ga uporablja program, nove vrednosti določa glede na 8 okoliških pik. Vseh 9 števil (vključno z vrednostjo pike, ki jo spreminjamo) shrani v polje, ga uredi po velikosti in nato izbere novo vrednost, ki se nahaja na polovici polja (mediana). Rezultat tega je globinska slika, z bistveno manj šuma.

Izračun vozlišč: Po filtriranju globinskih podatkov sledi izračun vozlišč. Ta so izračunana po sledeči formuli:

$$v_i(u) = D_i(u)K^{-1}[u, 1] \tag{3.2}$$

$D_i(u)$ predstavlja posamezno vrednost v množici globin. Za vsako slikovno piko u imamo eno vrednost. K je matrika notranjih parametrov infrardeče kamere in ima 3 vrstice in 3 stolpce. Zadnji element v enačbi je slikovna pika (*pixel*), predstavljen s koordinatama osi x in osi y.

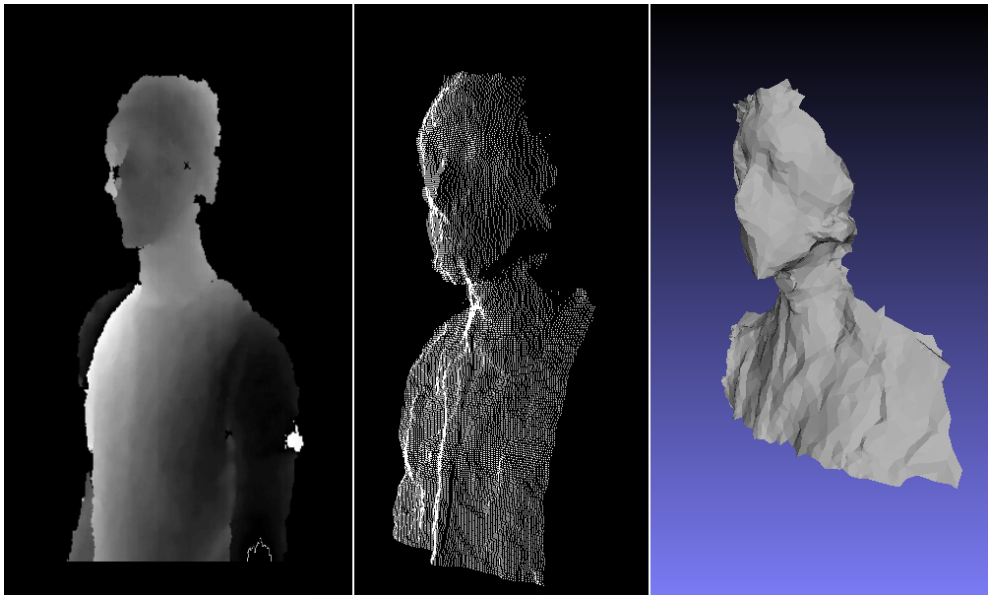
$$u = [x, y, 1] \tag{3.3}$$

Po izračunu vseh vozlišč sledi vmesni korak, v katerem se vrednosti zapišejo v besedilno datoteko.

Izračun normal: Premica ali vektor je nekemu objektu normala takrat, ko je nanj pravokoten/a. Pri grajenju mreže so normale vozlišč pomembne, če želimo predmet osvetliti. Hranijo namreč informacijo o površju, ki je pomembna za izračun odboja svetlobe. Brez normal predmeta ne moremo ustrezno osvetliti, oblika pa je tako bistveno manj izrazita. Program normale izračuna po naslednji enačbi:

$$n_i(u) = (v_i(x+1, y) - v_i(x, y)) \times (v_i(x, y+1) - v_i(x, y)) \tag{3.4}$$

Vse normale so normalizirane. To pomeni, da je vsaka njena komponenta deljena z velikostjo cele normale. Rezultat tega je vektor enotske velikosti. Po izračunu normal se te zapišejo v ustrezno besedilno datoteko.



Slika 3.5: Od globinske slike (levo) do oblaka točk (na sredini) in nazadnje do 3D mreže (desno).

Izvoz končne datoteke: v zadnjem koraku aplikacija izračunana vozlišča in normale uredi v pregleden seznam in jih shrani v skupno datoteko formata `.xyz`. Za izgradnjo mreže iz vozlišč in normal ogrodje uporablja najnovejšo različico knjižnice CGAL (*Computational Geometry Algorithms Library*) programskega jezika C++ [1]. Postopek, uporabljen pri rekonstrukciji, je algoritem, opisan v poglavju o teoretičnem ozadju, imenovan Poissonova rekonstrukcija površja (glej 2.2.3). Posebna ureditev končne datoteke in njena vrsta sta lastnosti vhodnih podatkov, ki jih zahteva knjižnica. Končni rezultat je datoteka tipa `.off`, v kateri so shranjeni podatki o rekonstruirani mreži.

3.3 Uporabljene tehnologije in programi

Pri implementiranju lastnega ogrodja za Kinect je bilo uporabljenih več programov in programskih knjižnic. Glavno razvojno okolje v projektu je Microsoft Visual Studio 2010 Express, koda pa je napisana v programskem jeziku C#. V Tabeli 3.2 so napisani vsi postopki in tehnologije, ki so bile uporabljene.

Postopek	Tehnologije
Zajem globinskih podatkov	Naprava Microsoft Kinect za XBOX 360 in razvojno okolje Microsoft Visual Studio 2010 Express – programski jezik C#
Obdelava podatkov: izračun vozlišč in normal, ureditev v izhodne datoteke	Razvojno okolje Microsoft Visual Studio 2010 Express – programski jezik C#
Kalibracija kamer Kinecta	Program Camera Calibration Toolbox for Matlab
Vizualizacija oblaka točk	Programska knjižnica OpenTK ali *The Open Toolkit Library* za programski jezik C#
Izgradnja končne mreže	Programska knjižnica CGAL ali *Computational Geometry Algorithms Library* za programski jezik C++
Prikaz in urejanje ustvarjenih mrež	Program MeshLab v1.3.1

Tabela 3.2: Postopki in tehnologije, ki so bile uporabljene..

Poglavje 4

Sklep

4.1 Izboljšave

Pri razvoju programov je vedno prostor za izboljšave. Tudi aplikacija, razvita za potrebe te diplomske naloge, ni nobena izjema. Na tem področju je namreč še zelo veliko možnosti, ki v tem strokovnem besedilu niso omenjene, ampak so kljub temu vredne nadaljnjega raziskovanja. Če bi želeli ustvariti seznam vseh podrobnosti, ki si zaslužijo več pozornosti, bi bil ta odločno predolg. Zato sem za konec ta seznam ustrezno priredil, da vsebuje le najbolj bistvene izboljšave, s katerimi bi lahko dopolnil aplikacijo.

1. Delovanje v realnem času: To je način delovanja programa, ki v času razvoja ogrodja ni bil deležen velike pozornosti. Ta odločitev morda deluje nenavadno, saj veliko kvalitetnih in obstoječih rešitev pogosto privzame tak način delovanja. Uporabniku namreč omogoča prijetnejšo uporabniško izkušnjo in hkrati navdušuje s sprotnim prikazovanjem rezultatov. Pogostejša uporaba tega načina je razumljiva, saj v današnjem času cenimo sisteme, ki delujejo hitro, čeprav je posledica tega ponavadi znaten padec kvalitete. Ta padec predstavlja glavno hibo delovanja v realnem času. Če se je želimo znebiti, lahko uporabimo sisteme, ki sproti samo zajemajo podatke, za prikaz rezultatov pa potrebujejo nekaj časa. Oba načina delovanja imata svoje prednosti in

slabosti, zato se zdi, da je še najboljša rešitev program, ki predstavlja kompromis med obema. Dobra aplikacija bi lahko uporabniku ponudila možnost, da se sam odloči, katera različica bolj ustreza težavi, ki jo rešuje. To bi razvijalcu povzročilo nekaj dodatnega dela, ampak bi mu s tem prihranilo težavno odločitev. Implementacija večih načinov delovanja nedovmno pozitivno vpliva na kvaliteto same aplikacije.

2. Teksturiranje s pomočjo barvne slike: Z uporabo barvne slike, zajete s pomočjo Kinectove barvne kamere, je mogoče za ustvarjene mreže narediti tudi ustrezne teksture. Vsaka slikovna pika barvne slike nosi podatke o barvi. Ker imata obe sliki, ki jih lahko zajamemo s Kinectom, enako ločljivost[1], lahko za vsak piksel v globinski sliki najdemo ustreznega v barvni. Sprotno teksturiranje uporabniku prihrani čas in delo, ki bi ju sicer moral vložiti v 'ročno' obdelavo ustvarjene mreže. Mogoč bi bil tudi prikaz vmesnih rezultatov s pomočjo oblaka točk, kjer bi vsaki točki v oblaku določili svojo barvo.

3. Upoštevanje gibanja v sistemu: Pri rekonstrukcijah, opisanih v tem delu, govorimo o negibni kameri in prizorišču. Čeprav to do neke mere postopek poenostavi, nudi vpeljava gibanja določene prednosti. Z gibanjem kamere okoli scene ali obratno zajamemo informacije iz več zornih kotov. To izboljša kakovost podatkov in dopolnjuje navidezni model, saj lahko posnamemo tudi površine, ki niso vidne iz ene same točke. Vendar pa moramo za dosego tega rešiti podproblem združevanja posameznih slik. Če posnamemo dovolj slik, te sčasoma začnejo sestavljati celoto. Da to dosežemo, jih moramo združiti. Algoritem, ki se v teh primerih pogosto uporablja, je tako imenovani algoritem šivanja (*stitching algorithm*).

[1]Ločljivost obeh slik znaša 640×480 slikovnih pik.

4.2 Zaključek

Po nekoliko bolj obširnem pregledu področja tridimenzionalne rekonstrukcije in zajema podatkov z optičnimi čitalci lahko nedvomno potrdimo, da ima ta veja računalniške grafike še zelo veliko potenciala. Odlikujeta jo hiter tehnološki razvoj in rastoča množica strokovnjakov ter ostalih privržencev. Rezultate njihovega dela lahko danes uporabimo na mnogih področjih in s tem izboljšamo kakovost človeškega življenja. Svet je Kinect sprejel z odprtimi rokami in nedvomno bo tako tudi z njegovimi nadgradnjami. Večja natančnost in zmogljivost bosta pozitivno vplivali na uporabniško izkušnjo pri igranju iger in na razvoj naprednejših ogrodij za zajem predmetov. Šele ob koncu tega pregleda človek dobi predstavo o velikosti te tematike, to diplomsko delo pa lahko obravnavamo kot uvod v tehnologije zajemanja tridimenzionalnih predmetov.

Literatura

[1] P. Alliez, L. Saboret, G. Günnebaud. "Surface Reconstruction from Point Sets", v priročniku: *CGAL User and Reference Manual*, CGAL Editorial Board, 4.0 Edition, 2012.

[2] M. Marolt. Predavanje, Tema: "Projekcije", Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, Ljubljana, Slovenija, Oktober 20, 2011.

[3] Z. Zhang, *Flexible Camera Calibration By Viewing a Plane From Unknown Orientations*, Redmond, WA: Microsoft Research, 1999.

[4] F. Bernardini, H. Rushmeier. "The 3D Model Acquisition Pipeline". *Computer Graphics Forum*, št. 2, zv. 21, str. 149-172, 2002.

[5] M. Kazhdan, M. Bolitho, H. Hoppe. "Poisson Surface Reconstruction", predstavljeno na četrtem Simpoziju o Geometričnem Procesiranju, Sardinija, Italija, 2006.

[6] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe. *KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera*, Santa Barbara, CA: ACM, Oktober 16-19, 2011.

[7] "How Kinect and Kinect Fusion (Kinfu) Works". Dostopno na: http://razorvision.tumblr.com/post/15039827747/how-kinect-and-kinect-fusion-kinfu-work, [Sept. 13, 2012]

[8] M. Bolitho, M. Kazhdan, R. Burns, H. Hoppe. *Parallel Poisson Surface Reconstruction*, Redmond, WA: Microsoft Research, 2009.

[9] C. Gotsman., M. Kazhdan. Prosojnice, Tema: "Poissonova rekonstrukcija površja", Technion, Izrael, 2011.

[10] J. Smíšek, M. Jančošek, T. Pajdla. *3D with Kinect*, Praga, Češka: Faculty of electrical engineering, Czech Technical University in Prague, 2011.

[11] "Kinect now able to do accurate 3d Facial Scan". Dostopno na: http://www.kinecthacks.com/kinect-now-able-to-do-accurate-3d-facial-scan/, [Sept. 13, 2012]

[12] "Matherix 3Dify uses Kinect for 3D reconstruction". Dostopno na: http://www.kinecthacks.com/matherix-3dify-uses-kinect-for-3d-reconstruction/, [Sept. 13, 2012]

[13] "ReconstructMe". Dostopno na: http://reconstructme.net/, [Sept. 13, 2012]

[14] "German Research Center for Artificial Intelligence". Dostopno na: http://www.dfki.de/web, [Sept. 13, 2012]

[15] "Kinect Hacking 103: Looking at Kinect IR Patterns". Dostopno na: http://www.futurepicture.org/?p=116, [Sept. 13, 2012]

[16] Čamera Calibration and 3D Reconstruction". Dostopno na: http://opencv.willowgarage.com/documentation/camera_calibration_and_3d_reconstruction.html, [Sept. 13, 2012]

[17] "Kinect for Windows". Dostopno na: http://www.microsoft.com/en-us/kinectforwindows/, [Sept. 13, 2012]

[18] Čamera Calibration Toolbox for Matlab". Dostopno na: http://www.vision.caltech.edu/bouguetj/calib_doc/, [Sept. 13, 2012]

[19] "Microsoft Kinect Teardown". Dostopno na:
http://www.ifixit.com/Teardown/Microsoft-Kinect-Teardown/4066,
[Sept. 13, 2012]

[20] "Matt's Webcorner". Dostopno na:
http://graphics.stanford.edu/ mdfisher/Kinect.html, [Sept. 13, 2012]